توری جستجوی دودوئی کامل :
یک ساختمان داده موازی برای کامپیوتر های موازی با حافظه توزیع شده

حمید بیگی محمدرضا میبدی

دانشکده مهندسی کامپیوتر
دانشگاه صنعتی امیرکبیر
تهران - ایران

چکیده

یک ماشین دیکشنری از پایگاه داده ای از زوج (کلید، رکورد) و مجموعه ای از عملگرها تشکیل میشود. عملگر های درج، حذف، حذف رکورد دارای کوچکترین کلید، حذف رکورد دارای بزرگترین کلید، جستجو، رکورد دارای نزدیکترین کلید، رکورد بعد، رکورد قبل ، رکورد دارای کوچکترین کلید، رکورد دارای بزرگترین کلید ، تعداد کلید های کوچکتر و تعداد رکوردهای موجود در پایگاه داده نمونه هایی از عملگرهای ماشین های دیکشنری هستند. پیاده سازی های مختلفی از ماشین های دیکشنری برای کامپیوتر های موازی با حافظه توزیع شده ارائه شده است که زنجیره مرتب [۲۷]، مکعب متوازن [۱۰] و هیپ بانین [۲۳] [۲۶] از آن جمله اند. توری جستجوی دودوئی نمونه ای از یک ساختمان داده موازی برای ماشین های دیکشنری بمنظور پیاده سازی در کامپیوترهای موازی با حافظه توزیع شده با ساختار توری یا ابر مکعب میباشد. پیاده سازی توری جستجوی دودوئی آنطور که در [۲۴] ارائه شده است دارای اشکالاتی میباشد که میتوان به ایجاد حفره در هنگام عمل درج ، از بین رفتن خاصیت توری جستجوی دودوئی پس از عمل حذف ، عدم توازن توری و پایین بودن توان عملیاتی اشاره نمود. اشکالاتی مانند عدم توازن در توری، تشکیل حفره در هنگام درج و ازبین رفتن خاصیت توری جستجوی دودوئی پس از عمل حذف ، توسط ساختمان داده توری جستجوی دودوئی متوازن برطرف شده است [۴]. در این مقاله توری جستجوی دودوئی کامل معرفی میگردد. این ساختمان داده بمنظور حل مشکلاتی که در توری جستجوی دودوئی و توری جستجوی دودوئی متوازن وجود دارد ارائه گردیده است. در این ساختمان داده، سطوح مختلف توری بترتیب پر میگردند و این خود باعث افزایش کارایی میگردد. در هنگام پیاده سازی این ساختمان داده روی توری یا ابر مکعب، ماشین از نظر منطقی از دو شبکه غیر مجزا پخش عملگرها و ارسال جواب ها تشکیل میشود. پخش عملگرها و ارسال جواب ها از طریق دو شبکه مختلف سبب کم شدن ترافیک روی شبکه میگردد. مقایسه توری جستجوی دودوئی کامل با طرح های دیگر گزارش شده برای کامپیوترهای موازی با حافظه توزیع شده و ساختار توری [۱۱][۱۲][۳۱][۳۴]، نشان میدهد که توری جستجوی دودوئی کامل دارای کارایی بمراتب بالاتری میباشد.

کلمات کلیدی : ساختمان داده موازی، ماشین دیکشنری، توری جستجوی دودوئی ، کامپیوترهای موازی

۱- مقدمه

یک ماشین دیکشنری از پایگاه داده ای از زوج (کلید، رکورد) و مجموعه ای از عملگرها تشکیل میشود. عملگر های درج[۱](Insert)، حذف[۲](Delete)، حذف رکورد دارای کوچکترین کلید[۳](XMin)، حذف رکورد دارای بزرگترین کلید[۴](XMax)، جستجو(Search)، نزدیکترین کلید(Near)، رکورد بعد (Next)، رکورد قبل (Prev) ، رکورد دارای کوچکترین کلید (Min)، رکورددارای بزرگترین کلید (Max) ، تعداد کلید های کوچکتر (CountLess) و تعداد رکوردهای موجود (Count) در پایگاه داده نمونه هایی از عملگرهای ماشین های دیکشنری هستند. فرض کنید F نشاندهنده مجموعه زوج های (k, r) با کلید k و رکورد r موجود در پایگاه داده باشد. اگر بتوان برای کلید k ، رکورد r را پیدا نمود که زوج (k, r) در پایگاه داده موجود باشد در اینصورت گفته میشود که کلید k در دیکشنری ذخیره شده است. فرض کنید $F(k) = \{(k, r) \mid (k, r) \in F\}$. بنابراین در صورت وجود کلید k در دیکشنری، F(k) یک مجموعه تک عضوی و در غیر اینصورت یک مجموعه تهی میباشد. شرح تعدادی از عملگرهای دیکشنری بصورت زیر است.

Insert (k, r) :
$F \leftarrow (F - F(k)) \cup \{(k,r)\}$

Delete (k) :
$F \leftarrow F - F(k)$

Min (k) :
return $F(k_{min})$

Max (k) :
return $F(k_{max})$

XMin (k) :
$F \leftarrow F - F(k_{min})$
return $F(k_{min})$

XMax (k) :
$F \leftarrow F - F(k_{max})$
return $F(k_{max})$

Search (k) :
if $k \in F$ **then return** F(k)

```
              else  return 'not found'

Next (k) :
     if Succ (k) ∈ F then return F(Succ (k))
                       else  return 'Last Key'
Prev (k) :
     if Pred (k) ∈ F then return F(Pred (k))
                       else  return 'First Key'
Near (k) :
     return F (X_near) where k_near is the stored key closest to k

CountLess (k) :
     return number of keys x where x < k

Count () :
     return number of keys k where k ∈ F
```

اگر کلید k قبل از عمل درج در دیکشنری موجود باشد ، عمل درج را زائد[۵] و اگر قبل از عمل حذف در دیکشنری موجود نباشد ، عمل حذف را زائد میگویند. برای سهولت ارائه در این مقاله ، یک زوج توسط کلید آن نمایش داده میشود همچنین فرض میشود که عملگرهای درج و حذف زائد نباشند.

عملگرهای ماشین دیکشنری را میتوان به دو دسته عملگرهای اصلاحی[۶] و عملگرهای پرس وجو[۷] تقسیم نمود. عملگرهای اصلاحی محتوای ماشین دیکشنری را تغییر میدهند در حالیکه عملگرهای پرس وجو محتوای ماشین را تغییر نمیدهند. عملگرهای درج، حذف و حذف رکورد دارای کوچکترین کلید نمونه هایی از عملگرهای اصلاحی و عملگرهای جستجو، رکورد بعد و رکورد دارای کوچکترین کلید نمونه هایی از عملگرهای پرس و جو میباشند.

برای پیاده سازی ماشین های دیکشنری سه روش گزارش شده ا ست که عبارتند از: الگوریتم های ترتیبی برای پیاده سازی روی کامپیوتر های تک پردازنده [۱][۳۶] ، الگوریتم های موازی برای پیاده سازی روی سخت افزار [۳] [۶] [۱۲] [۱۵][۲۱][۲۵][۳۱][۳۴]، و الگوریتم های موازی برای پیاده سازی روی کامپیوترهای موازی[۲][۷][۱۳][۱۴][۱۶][۱۷][۳۵]. الگوریتم های موجود برای پیاده سازی ماشین های دیکشنری روی کامپیوتر های موازی به دو دسته، الگوریتم های موازی برای پیاده سازی روی کامپیوترهای موازی با حافظه مشترک و الگوریتم های موازی برای پیاده سازی روی کامپیوترهای موازی با حافظه توزیع شده تقسیم میشوند. در طراحی الگوریتم های موازی برای پیاده سازی روی کامپیوترهای موازی با حافظه مشترک، توجه به تداخل فرایند های موازی در دسترسی به ساختمان داده مشترک میباشد در حالیکه در طراحی الگوریتم های موازی برای پیاده سازی روی کامپیوترهای با حافظه توزیع شده، توجه به استخراج توازی در ساختمان داده است. عملگرهای موازی برای درخت های دودوئی [۱۷]، درخت های متوازن [۱۹]، عملگرهای درج و جستجو در درخت های AVL و ۲-۳ [۱۳][۱۴]، عملگرهای موازی درج و حذف در هیپ [۲۹] نمونه هایی ازالگوریتم های موازی ارائه شده برای کامپیوترهای موازی با حافظه مشترک میباشند. مکعب متوازن[۸] [۱۰]، زنجیره مرتب[۹] [۲۷] وهیپ بانین[۱۰] [۲۳][۲۶] نمونه هایی از ساختمان داده موازی برای کامپیوترهای موازی با حافظه توزیع شده و ساختار ابر مکعب میباشند. کارایی یک ماشین دیکشنری با توجه به معیارهای زیر مورد بررسی قرار میگیرد.

فاصله شروع دو عملگر[۱۱]: حداقل فاصله زمانی که بین شروع اجرای دو عملگر متفاوت نیاز میباشد را فاصله شروع دو عملگر مینامند.

زمان پاسخ[۱۲]: زمان بین شروع و اتمام یک عملگر را زمان پاسخ عملگر مینامند.

توان عملیاتی[۱۳]: تعداد عملیات موازی انجام شده در واحد زمان را توان عملیاتی ماشین دیکشنری مینامند.

در ادامه سه ماشین دیکشنری که روی کامپیوترهای موازی با حافظه توزیع شده با ساختار توری[۱۴] پیاده سازی شده اند شرح داده میشود. در پایان این مقاله کارایی توری جستجوی دودوئی کامل که در این مقاله پیشنهاد شده است با این سه ماشین که در زیر بطور مختصر شرح داده شده اند مقایسه خواهد شد.

ماشین Schmeck و Schroder: ساختار این ماشین بصورت یک توری $n \times n$ است که در آن هر پردازنده به چهار پردازنده همسایه اش متصل شده است. کمانهای این ماشین به دو دسته تقسیم میشوند. یکدسته از کمانها برای انتشار[۱۵] عملگرها و جمع آوری نتایج و دسته دیگر برای ارتباط بین پردازنده های همسایه بکار میروند. در این ماشین عملگرها بطور همزمان به تمام پردازنده های سطر اول ارسال میشوند بطوریکه پس از n واحد زمان ، عملگرها به سطر آخر میرسند و از ترکیب نتایج تولید شده در سطر آخر نتیجه عملگر حاصل میشود. این ماشین دارای فاصله شروع دو عملگر $O(1)$ و زمان پاسخ $O(n)$ میباشد[۳۱].

ماشین Dehne و Santoro : ساختار این ماشین بصورت یک توری $n \times n$ است که در آن هر پردازنده به چهار یا هشت پردازنده همسایه اش متصل شده است. از نظر منطقی این ماشین از دو شبکه مارپیچی[۱۶] و شبکه انتشار تشکیل شده است که هر دو در توری جاداده میشوند و بطور همزمان عمل میکنند. شبکه مارپیچی بصورتی در توری جا داده میشود که هر پردازنده فقط یک بار در آن ظاهر میشود. رکوردها بصورت نزولی (از جهت پردازنده ورودی-خروجی[۱۷]) در این شبکه ذخیره میشوند. بطوریکه رکورد با بزرگترین کلید در پردازنده ورودی- خروجی قرار دارد. شبکه انتشار برای عملگر جستجو بکار میرود. این شبکه ها بصورت مجزا یا غیر مجزا روی توری جا داده میشوند و یک گراف بدون حلقه را تولید میکنند. این ماشین دارای فاصله شروع دو عملگر $O(1)$ و زمان پاسخ $O(n)$ میباشد[۱۱][۱۲].

ماشین Youn و Young: ساختار این ماشین بصورت یک توری $n \times n$ است که در آن هر پردازنده به شش پردازنده همسایه اش متصل میباشد. از نظر منطقی این ماشین از دو شبکه خطی[۱۸] و شبکه پوشا[۱۹] تشکیل شده است که هر دو در توری جاداده میشوند. شبکه خطی برای عملگر درج و شبکه پوشا برای عملگر جستجو بکار میرود. این ماشین دارای فاصله شروع دو عملگر $O(1)$ و زمان پاسخ $O(n)$ میباشد[۳۴].

توری جستجوی دودوئی[۲۰]، یک نمونه از ساختمان داده های موازی روی کامپیوترهای موازی با حافظه توزیع شده و ساختار توری یا ابر مکعب میباشد که برای اولین بار در [۲۴] پیشنهاد شده است. پیاده سازی این ساختمان داده روی توری یا ابر مکعب آنطور که در [۲۴] ارائه شده است دارای اشکالاتی میباشد که میتوان به ایجاد حفره در هنگام عمل درج ، از بین رفتن خاصیت توری جستجوی دودوئی پس از عمل حذف ، عدم توازن توری و پایین بودن توان عملیاتی آن اشاره نمود. عدم توازن توری سبب افزایش زمان پاسخ میگردد. برای کاهش زمان پاسخ ، ساختمان داده توری جستجوی دودوئی متوازن[۲۱] پیشنهاد شده است [۴] که در آن تعداد رکورد های ذخیره شده در تمام پردازنده های نیم توری تقریبا برابر میباشد. در توری جستجوی دودوئی $n \times n$ و توری جستجوی دودوئی متوازن $n \times n$ فاصله زمانی شروع دو عملگر مختلف $O(n)$ میباشد و بهمین دلیل دارای توان عملیاتی بالایی نیستند. در این مقاله یک ساختمان داده موازی جدید بنام توری جستجوی دودوئی کامل[۲۲] معرفی میگردد. این ساختمان داده که مشابه توری جستجوی دودوئی است دارای توان عملیاتی بالاتری میباشد. در هنگام پیاده سازی این ساختمان داده روی توری یا ابر مکعب، ماشین از نظر منطقی از دو شبکه غیر مجزا پخش عملگرها و ارسال جواب ها تشکیل میشود. شبکه پخش عملگرها مسیری یکتا بین ریشه و گره های دیگر توری جستجوی دودوئی کامل ایجاد میکند که باعث میگردد که عملگرهای ماشین دیکشنری بصورت متوالی وارد پردازنده‌ها گردند. این باعث میشود تا سازگاری در توری جستجوی دودوئی کامل بسادگی قابل پیاده سازی باشد. در توری جستجوی دودوئی کامل، عملگرها دارای زمان پاسخ از مرتبه $O(n)$ و فاصله زمانی بین شروع دو عملگر از مرتبه $O(1)$ میباشد.

بخش های بعدی مقاله بصورت زیر سازماندهی شده ا ست. ساختمان داده توری جستجوی دودوئی کامل در بخش ۲ معرفی شده است. پیاده سازی عملگرهای این ساختمان داده و سازگاری آن در بخش های ۳ و ۴ بررسی گردیده است. مقایسه ای بین توری جستجوی دودوئی کامل و چند ماشین دیکشنری گزارش شده [۱۱][۱۲][۳۱][۳۴] برای کامپیوترهای موازی با ساختار توری در نتیجه گیری آمده است.

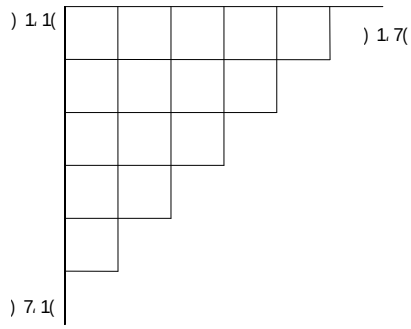
۲- توری جستجوس دودوئی کامل

در این قسمت در ابتدا تعاریف توری d- بعدی و نیم توری دو بعدی داده شده است و سپس توری جستجوی دودوئی و توری جستجوی دودوئی متوازن تعریف و اشکالات موجود در آنها بررسی گردیده است و نهایتا تعریف توری جستجوی دودوئی کامل که پیاده سازی آن در این مقاله آمده است داده شده است.

توری d- بعدی: یک توری d- بعدی از اجتماع گره هایی تشکیل میشود که در طول نقاط فضای d- بعدی قرار گرفته اند و کمانی بین هر گره و نزدیکترین همسایه هایش وجود دارد. گره های یک توری d- بعدی با n_i نقطه در طول بعد i ام بوسیله d- تایی $(x_1, x_2, ..., x_d)$ نمایش داده میشوند بطوریکه هر یک از مختصات x_i (برای $i = 1, 2, ..., d$) میتوانند مقادیر صحیح ۱ تا n_i را اختیار نمایند. دو گره $(x_1, x_2, ..., x_d)$ و $(y_1, y_2, ..., y_d)$ همسایه یکدیگرگفته میشود اگر یک i $(1 \le i \le d)$ وجود داشته باشد بطوریکه برای تمام i های مخالف j داشته باشیم $|x_i - y_i| = 1$ و $x_j = y_j$.

نیم توری دو بعدی: یک نیم توری دو بعدی $n \times n$ (که به اختصار نیم توری $n \times n$ نامیده میشود) از تقسیم توری دو بعدی $n \times n$ در راستای قطر {$(1,n),(2,n-1),...(n-$

{(n,1),(1,2)} بدست میاید. شکل ۱ یک نیم توری دو بعدی ۷ × ۷ را نشان میدهد. در این نیم توری، گره (۱،۱) ریشه، گره های روی قطر $N_L = \{ (i, j) \mid i + j = n + 1 \}$ برگ ، گره های روی اضلاع $N_{TB} = \{ (1, k) \mid 1 \le k < n \}$ و $N_{LB} = \{ (k, 1) \mid 1 < k < n \}$ گرههای مرزی (بترتیب مرز بالا و مرز چپ) و بقیه گره ها ، گره های میانی نامیده میشوند. نیم توری $n \times n$ دارای n سطح میباشد که بترتیب از سطح۱ الی n شماره گذاری میشوند. ریشه نیم توری در سطح ۱ قرار دارد. در سطح k ام نیم‌توری k گره $L_K = \{ (i, j) \mid i + j = K + 1 \}$ وجود دارند که برادر یکدیگر نامیده میشوند. گره $(k, 1)$ را برادر کوچکتر(گره مرزچپ) وگره $(1, k)$ برادر بزرگتر (گره مرز بالا) نامیده میشوند. برگ های نیم توری در سطح n ام نیم توری قرار دارند. گره میانی (I, J) دارای دو پدر راست $(I-1, J)$ و پدر چپ $(I, J-1)$ ، گره مرزبالا $(1, J)$ دارای پدر سمت چپ $(1, J-1)$، گره مرز چپ $(I, 1)$ دارای پدر سمت راست $(I-1, 1)$ و گره ریشه بدون پدر میباشند. گره غیر برگ(I, J) دارای دو فرزند سمت راست$(I, J+1)$ وفرزند سمت چپ $(I+1, J)$ میباشد. شماره سطح یک گره ، عمق آن گره و بیشترین عمق در بین گره های غیر تهی، عمق توری نامیده میشود.



شکل ۱: نیم توری دو بعدی ۷× ۷

توری جستجوی دودوئی: توری جستجوی دودوئی یک نیم توری دو بعدی است که دارای شرایط زیر میباشد.

۱- هر گره غیر برگ میتواند دارای صفر یا یک رکورد باشد.

۲- هر گره برگ میتواند دارای صفر یا بیشتررکورد باشد.

۳- کلید رکورد ذخیره شده در گره پدر از کلید رکورد ذخیره شده در گره فرزند سمت چپ بزرگتر و از کلید رکورد ذخیره شده در گره فرزند سمت راست کوچکتراست.

پیاده سازی توری جستجوی دودوئی برای کامپیوترهای موازی با حافظه توزیع شده و ساختار ابر مکعب در [۲۴] آمده است .الگوریتم های ارائه شده برای این ساختمان داده دارای اشکالاتی بوده است. یکی از این اشکالات بوجود آمدن حفره (یک گره میانی خالی که دارای دو پدر و فرزند سمت راست پر باشد) در عملگر درج میباشد که سبب از بین رفتن خواص توری جستجوی دودوئی میگردد. اشکال دیگرعدم برقراری خواص توری جستجوی دودوئی و تشکیل حفره پس از عمل حذف میباشد. اشکالات فوق بصورت زیر برطرف شده اند[۴]. برای از بین بردن حفره دوروش پیشنهاد شده است. در روش اول هنگام درج رکورد x، تشکیل یا عدم تشکیل حفره بررسی میشود. در صورت تشکیل حفره با جایگزینی رکورد پدر سمت راست حفره توسط رکورد x و درج رکورد پدر سمت راست حفره در توری جستجوی دودوئی از تشکیل حفره جلوگیری میشود. روش دوم برای از بین رفتن حفره استفاده از توری جستجوی دودوئی متوازن میباشد. توازن در توری جستجوی دودوئی سبب عدم تشکیل حفره میگردد.

عدم برقراری خواص توری جستجوی دودوئی و تشکیل حفره در هنگام حذف رکورد x بصورت زیر برطرف شده است. برای حذف رکورد x ، ابتدا گره حاوی x پیدا میشود. اگر گره پیدا شده برگ یا گره میانی بدون فرزند یا گره مرزی بدون فرزند باشد، رکورد x حذف میشود. اگر گره پیدا شده یک گره مرزی باشد از برگ تا گره پیدا شده شیفت داده میشوند تا حفره ایجاد شده پر گردد. اگر گره پیدا شده دارای یک فرزند باشد ورکورد فرزند موجود شرایط انتقال به گره پدر را داشته باشد، رکورد فرزند جایگزین رکورد پدر میشود ودر غیر اینصورت رکورد یکی از والدین که شرایط جایگزینی را داشته باشد به این گره منتقل میشود و عملگر حذف به گرهی منتقل میگردد که رکورد آن انتقال یافته است. در صورتیکه گره پیدا شده دارای دو فرزند باشد در صورت برقرار بودن شرایط جایگزینی، حفره ایجاد شده توسط رکورد یکی از فرزندان و در غیر اینصورت توسط رکورد یکی از والدین پر میشود و عملگر حذف به گره واجد شرایط منتقل میشود. جایگزینی و انتقال تا زمانی ادامه پیدا میکند که به گره برگ ، گره مرزی یا گره میا نی بدون فرزند برسیم.

در توری جستجوی دودوئی میتوان حداکثر یک رکورد در گره های غیر برگ و چندین رکورد را در گره های برگ ذخیره نمود. بهمین دلیل بار روی گره های برگ بیشتر میبا شد که منجر به پایین آمدن توان عملیاتی ساختمان داده میگردد. مشکل دیگری که در توری جستجوی دودوئی وجود دارد رشد یک بعدی توری (مانند آنچه در درخت جستجوی دودوئی وجود دارد) میباشد. اگر داده های درج شونده بترتیب صعودی (نزولی) وارد توری شوند تنها در گره های مرزبالا (مرز چپ) توری قرار میگیرند و بقیه گره ها خالی می ما ند. رشد توری در یک بعد و عدم توازن تعداد رکورد ها در گره های توری نیز سبب کاهش توان عملیاتی این ساختمان داده میگردد. برای رفع این اشکالات، توری جستجوی دودوئی متوازن که تعریف آن در زیر آمده است پیشنهاد شده است[۴]. در این ساختمان داده رکورد ها بین گره های توری بصورت متوازن توزیع میشوند تا از این طریق توان عملیاتی ساختمان داده افزایش یابد.

توری جستجوی دودوئی متوازن: توری جستجوی دودوئی متوازن یک نیم توری دوبعدی است که دارای شرایط زیر باشد.

۱- هر گره میتواند دارای صفر یا بیشتررکورد باشد.

۲- کلید رکورد های ذخیره شده در گره پدر از کلید رکورد های ذخیره شده در گره فرزند سمت چپ بزرگتر و از کلید رکورد های ذخیره شده در گره فرزند سمت راست کوچکتراست.

۳- قدر مطلق تفاوت تعداد رکورد های ذخیره شده در هر دو گره موجود در توری حداکثر یک است.

۴- یک گره میانی خالی نمیتواند فرزند پر داشته باشد.

۵- یک گره مرز چپ خالی میتواند فرزند سمت راست پر داشته باشد ولی نمیتواند فرزند سمت چپ پر داشته باشد.

۶- یک گره مرز بالا خالی میتواند فرزند سمت چپ پر داشته باشد ولی نمیتواند فرزند سمت راست پر داشته باشد.

در توری جستجوی دودوئی و توری جستجوی دودوئی متوازن $n \times n$ فاصله زمانی شروع دو عملگر مختلف از مرتبه $O(n)$ میباشد[۴] که این خود باعث پایین آمدن توان عملیاتی این دو ساختمان داده میگردد. هدف از معرفی توری جستجوی دودوئی کامل حل مشکل پایین بودن توان عملیاتی در توری جستجوی دودوئی و توری جستجوی دودوئی متوازن میباشد.

توری جستجوی دودوئی کامل $n \times n$: توری جستجوی دودوئی کامل $n \times n$ (که به اختصار توری جستجوی دودوئی کامل نامیده میشود) یک نیم توری دوبعدی است که شرایط زیر را دارا میباشد.

۱- هر گره غیر برگ میتواند دارای صفر(گره خالی)، یک(گره نیمه پر) یا دو(گره پر) رکورد باشد.

۲- هر گره برگ میتواند دارای صفر یا بیشتر رکورد باشد.

۳- بزرگترین رکورد در هرگره از رکورد های ذخیره شده در فرزند سمت چپ بزرگتر و از رکورد های ذخیره شده در فرزند راست کوچکتر است.

۴- کوچکترین رکورد در هرگره از رکورد های ذخیره شده در فرزندان سمت چپ و سمت راست آن گره کوچکتر است.

۵- همه گره های نیمه پر در یک سطح قرار دارند یا بعبارتی دیگر در نیم توری تنها یک سطح وجود دارد که دارای گره های نیمه پر است.

۶- هر گره نیمه پر و یا پر دارای والدین پر میباشد.

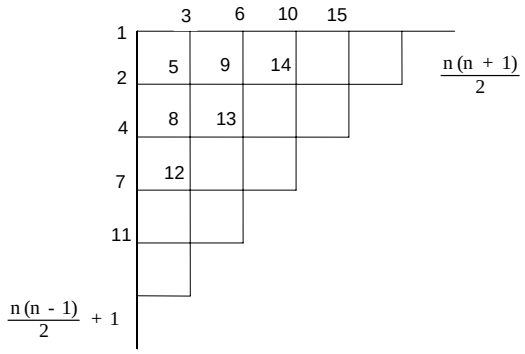
۷- یک سطح بطور همزمان نمیتواند دارای گره های خالی و پر باشد.

۸- یک گره نیمه پر نمیتواند دارای برادر بزرگتر پر باشد.

۹- یک گره خالی نمیتواند دارای برادر بزرگتر نیمه پر باشد.

۱۰- برای دو گره برگ a و b بطوریکه گره a برادر کوچکتر گره b باشد تعداد رکوردهای ذخیره شده در گره a مساوی یا بزرگتر از تعداد رکوردهای ذخیره شده در گره b می باشد.

فرض کنید که گره های نیم توری دوبعدی $n \times n$ با عمق m که در آن N رکورد ذخیره شده باشد را بصورت شکل ۲ شماره گذاری نماییم در اینصورت برای یک توری جستجوی کامل قضایای زیر صادق میباشد.



شکل ۲: شماره گذاری نیم توری دو بعدی

قضیه ۱: اگر $N \le n(n-1)$ و $N \le m^2$ باشد گرههای باشماره $1,\ldots,\frac{m(m-1)}{2}$ پر و گره های با شماره $\frac{m(m-1)}{2}+1,\ldots,N-\frac{m(m-1)}{2}$ نیمه پر هستند.

قضیه ۲: اگر $m^2 \le N \le n(n-1)$ باشد گرههای با شماره $1,\ldots,N-\frac{m(m+1)}{2}$ پر و گره های با شماره $\frac{m(m+1)}{2}+1,\ldots,N-\frac{m(m+1)}{2}$ نیمه پر هستند.

قضیه ۳: اگرشرط $N > n(n-1)$ برقراراشدگرههای برگ باشماره $1,\ldots,\frac{n(n-1)}{2}$ پر، برگهای باشماره $K+\frac{n(n-1)}{2}+1,\ldots,$ (برای $K=\text{mod}(N-n(n-1),n)$) دارای $\left\lfloor\frac{N-n(n-1)}{n}\right\rfloor$ رکورد و برگهای با شماره $K+1+\frac{n(n-1)}{2},\ldots,\frac{n(n+1)}{2}$ دارای $\left\lfloor\frac{N-n(n-1)}{n}\right\rfloor$ رکورد هستند.

اثبات قضایای فوق در [۵] آورده شده است.

فرض‌کنیدکه برای $1 \le I \le n-k$ مجموعه‌های $R_k(I)=\{(I,J) \mid k<J \le n\}$،

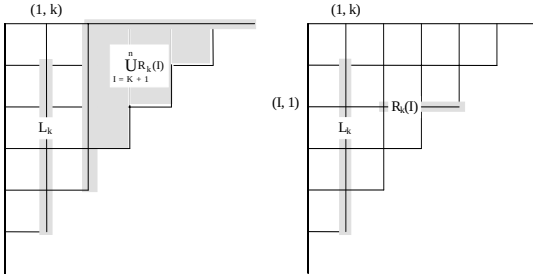
$L_k=\{(I,k) \mid k<I \le n\}$ و $\bigcup_{I=k+1}^{n} R_k(I)$ بصورت شکل ۳ تعریف شده باشند.

گره مرز بالا $(1,k)$ را کامل میگوییم اگر

- ۱اگره $(1,k)$ خالی یا نیمه پر باشد یا
- ۲تفاوت رکوردهای موجود در مجموعه های $R_k(1)=\{(1,J) \mid k<J \le n\}$ و $L_k=\{(I,k) \mid k<I \le n\}$ برابرصفریا۱- باشد وگره مرزبالا$(1,k+1)$ کامل باشد.

نیم توری دو بعدی $n \times n$ با ریشه q را کامل میگوییم اگر

- ۱ریشه کامل باشد و
- ۲نیم توری دو بعدی $(n-1)\times(n-1)$ باریشه فرزندسمت‌راست q کامل باشد.



شکل ۳

قضیه ۴: اگر در یک توری جستجوی دودوئی کامل N رکورد ذخیره شده باشد عمق توری $\min\left(n, \left\lfloor\frac{\sqrt{1+4N}-1}{2}\right\rfloor\right)$ میباشد.

اثبات : اثبات در دو قسمت انجام میگیرد

۱- تعداد رکوردهای ذخیره شده در برگها کمتر و یا مساوی دو باشد:

فرض میکنیم عمق توری جستجوی دودوئی کامل k باشد. با توجه به تعریف توری جستجوی دودوئی کامل، اگرتعداد رکوردهای‌ذخیره‌شده در توری بزرگتر از $k(k-1)$ و کوچکتر یا مساوی$k(k+1)$ باشد. با استفاده از نامساویهای $k(k+1) \ge N$ و $k(k-1) < N$ و حل آنها جوابهای مثبت زیر بدست میایند.

$$\frac{\sqrt{1+4N}-1}{2} \le k < \frac{\sqrt{1+4N}+1}{2}$$

بدلیل اینکه عمق توری عددی صحیح است، در نتیجه عمق برابر است با

$$k = \left\lfloor\frac{\sqrt{1+4N}-1}{2}\right\rfloor$$

۲- تعداد رکوردهای خیره شده در حداقل یک برگ بیش از دو رکورد باشد. دراینحالت عمق توری برابر n است. درنتیجه عمق توری دارای N رکورد، برابر است با

$$\min\left(n, \left\lfloor\frac{\sqrt{1+4N}-1}{2}\right\rfloor\right)$$

و بدین ترتیب قضیه اثبات می‌گردد■.

قضیه ۵ : زمان پاسخ عملگر جستجو در توری جستجوی دودوئی کامل که در آن N رکورد ذخیره شده باشد برابر است با:

$$\min\left(n-1, \left\lfloor\frac{\sqrt{1+4N}-1}{2}\right\rfloor\right)$$

اثبات: با توجه به قضیه ۴، در توری جستجوی دودوئی کامل با N رکورد، تعداد مقایسه های لازم برای پیدا کردن یک رکورد برابر است با

$$= \min\left(n, \left\lfloor\frac{\sqrt{1+4N}-1}{2}\right\rfloor\right) - 1$$

$$= \min\left(n-1, \left\lfloor\frac{\sqrt{1+4N}-1}{2}\right\rfloor - 1\right)$$

$$= \min\left(n-1, \left\lfloor\frac{\sqrt{1+4N}-1}{2}\right\rfloor\right)$$

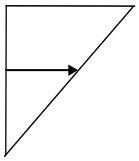
و قضیه اثبات می‌گردد■.

کوچکترین و بزرگترین رکورد هر گره بترتیب توسط دو متغیر SmallItem و LargeItem ذخیره میشوند. برای گرههای دارای یک رکورد مقدار SmallItem و LargeItem یکسان میباشند.

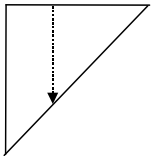
با توجه به تعریف توری جستجوی دودوئی کامل میتوان نتایج زیر را گرفت. برای سهولت نمایش، رکورد های SmallItem (S) ، LargeItem(L) و زوج (SmallItem, LargeItem)(I) بترتیب توسط خطوط نقطه چین، خط چین و پر نشان داده میشوند.

نتیجه ۱: رشته زیر یک رشته صعودی است.

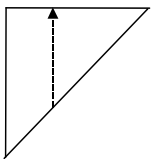
S (k, 1), L (k, 1), S (k, 2), L (k, 2), . . , S (k, n- k +1), L (k, n-I+1) }}



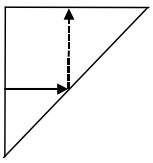
نتیجه ۲: رشته {{S(1,J), S(2,J),..., S(n-J+1, J)} یک رشته صعودی است.



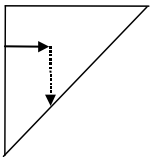
نتیجه ۳: رشته {{L (n-J+1, J),...,L (2,J), L (1,J)} یک رشته صعودی است.



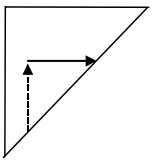
نتیجه ۴: رشته {{I (k,1),...,I(k,J-1), S(k,J), L(k,J),..., L(1,J)} **یک‌رشته** صعودی است.



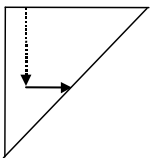
نتیجه ۵: رشته {I(k,1),...,I(k,J-1),S(k,J),..., S(J,n-J+1)} **یک‌رشته** صعودی است.



نتیجه ۶: رشته {L(n-J+1,J),...,L(k,J), I (k,J),..., I (k,n-I+1)} **یک‌رشته** صعودی‌است.

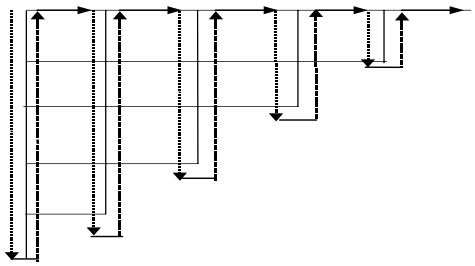


نتیجه ۷: رشته {S(1,J),..., S(k,J), I(k, J),..., I(k, n-k+1)} **یک‌رشته**صعودی است.



قضیه ۶: زنجیره زیر یک‌رشته صعودی است

{S(1,1),...,S(n,1),L(n,1),...,L(1,1),S(1,2),...,S(n-1,2),...,L(1,2),...,S(1,n-1) , S(2,n-), L(2,n-1), L(1,n-), S(1,n), L(1,n)}



اثبات: با استفاده از نتایج ۱، ۲و ۳.

۳- عملگرها

دراین‌بخش‌چگونگی‌کارکردعملگرهای‌درج،حذف،حذف‌رکورددارای‌کوچکترین کلید، جستجو، رکورددارای‌کلیدبعد، رکورددارای‌کوچکترین‌کلیدورکورددارای بزرگترین‌کلیدشرح‌داده‌میشود. برای‌سادگی‌ارائه، درادامه‌بحث‌رکوردهای موجود دریک‌گره‌غیربرگ توسط‌زوج (SmallItem, LargeItem) نشان‌داده میشود.

توازن در گره مرز بالا (1, k) توسط متغیر Balance نشان‌داده‌میشود. اگرمقدار این متغیرمساوی‌صفرباشددراینصورت این گره متوازن است و نیم توریی که ریشه آن گره مرز بالا (1, k) است یک توری جستجوی دودوئی کاملا پر میباشد. در صورتی که مقدار این متغیر بزرگتر از صفر باشد تعداد رکوردهای موجود در مجموعه $R_k(1)$ بیشتر ازتعداد رکورد های موجود در مجموعه L_k و در غیر اینصورت تعداد رکوردها ی موجود در مجموعه L_k بیشتر از تعداد رکورد های موجود در مجموعه $R_k(1)$ میباشد. بدلیل اینکه رکوردهایی که از طریق فرزند سمت راست گره مرزبالا (1,k) درج میشوند مشخص نیستند که در چه گره ای درج میشوند بنابراین مقدار متغیر Balance قابل محاسبه نیست. قضیه زیر بر اساس تعداد رکوردهایی که از طریق فرزند سمت راست و فرزند سمت چپ درج میشوند چگونگی محاسبه مقدار متغیر Balance را بیان میکند.

قضیه ۷: فرض کنید R و L بترتیب نشان دهنده تعداد رکورد های موجود در مجموعه های $\bigcup_{I=k+1}^{n} R_k(I)$ و L_k باشد. اگر گره مرز بالا، (1, k) کامل باشد در اینصورت رابطه زیر برقرار است.

$$R = \begin{cases} \left\lceil \frac{L}{2} \right\rceil \left(\left\lfloor \frac{L}{2} \right\rfloor + 1 \right) & \text{if } L \le 2(n-k) \\ (n-k)\left[L - (n-k+1) \right] & \text{if } L > 2(n-k) \end{cases}$$

اثبات: اثبات قضیه در دو قسمت انجام میگیرد.

۱- تعداد رکورد های ذخیره شده در برگ هاکمتر یا مساوی دو باشد:

با توجه به تعریف توازن سطحی و اینکه حداکثر دو رکورد در هر گره غیر برگ میتواند ذخیره شودتعداد گره های نیمه پر و پر در مجموعه های $R_k(1)$ وL_k برابر است با $n_L = \left\lceil \frac{L}{2} \right\rceil$. با توجه به تعریف توری جستجوی کامل، R برابر است با

$$R = \sum_{m=0}^{n_L - 1} (L - 2m) = \left\lceil \frac{L}{2} \right\rceil \left(\left\lfloor \frac{L}{2} \right\rfloor + 1 \right)$$

۲- تعداد رکورد های ذخیره شده در برگ ها بیش از دو باشد:

با توجه به تعریف نیم توری‌دو بعدی$n\times n$، (n-k) گره برگ در مجموعه $\bigcup_{I=k+1}^{n} R_k(I)$ و(n- k-1) گره غیر برگ در مجموعه L_k وجود دارند. با توجه به اینکه (n-k-1) گره غیر برگ موجود در مجموعه L_k پر (دارای دو رکورد) هستند با استفاده از قسمت ۱ همین قضیه، تعداد رکورد های موجود در گر های غیر برگ مجموعه R برابر است با

$R_1 = (n-k-1)(n-k)$

و(L-2(n-k-1) رکورد در برگ (n-k+1, k) ذخیره شده است. با توجه به تعریف توازن سطحی، تعداد رکورد های موجود در گره های برگ مجموعه $\bigcup_{I=k+1}^{n} R_k(I)$ ، (n-k) برابر تعداد رکوردهای موجود در برگ (n-k+1,k) میباشد. در نتیجه تعداد رکوردهای موجود در برگهای مجموعه برابر است با

$R_2 = [L-2(n-k-1)](n-k)$

بنابراین تعداد رکورد های موجود در مجموعه $\bigcup_{I=k+1}^{n} R_k(I)$ برابر است با

$R = R_1 + R_2 = [L - (n-k-1)](n-k)$

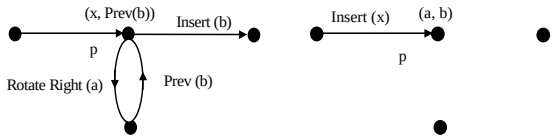
و بدین ترتیب قضیه اثبات میگردد.■

عملگر درج : این عملگر علاوه بر درج یک رکورد وظیفه حفظ خاصیت توری جستجوی دودوئی کامل را بر عهده دارد. این عملگر در حین درج یک رکورد در توری با ایجاد چرخش های لازم توازن در توری جستجوی دودوئی را برقرار مینماید. عملکرد این عملگر برای گره های مرز بالا و گره های دیگر متفاوت است. نحوه اجرای این عملگر با توجه به نوع گره بصورت زیر میباشد.

الف- اگرگره q یک گره مرزبالا باشد، عملگر درج رکورد x را از پدر سمت چپ خود دریافت میکند. در صورتیکه گره q خالی یا نیمه پر باشد رکورد x در این گره قرار میگیرد. در صورتیکه گره q دارای دو رکورد باشد متغیر Balance مشخص میکند که رکورد x ، باید از طریق کدام فرزند درج شود. اگر مقدارمتغیر Balance کوچکتر از صفر باشد رکورد x از طریق فرزند سمت راست درج میشود. در فرایند درج رکورد x، از طریق فرزند سمت راست امکان برهم خوردن خاصیت توری جستجوی دودوئی کامل وجود دارد. برای برقرار نمودن خاصیت توری جستجوی دودوئی کامل یکی از روشهای زیر میتواند مورد استفاده قرار گیرد.

۱- اگر رکورد x ، از رکورد LargeItem بزرگتر باشد، رکورد x ، از طریق فرزند سمت راست درج میشود.

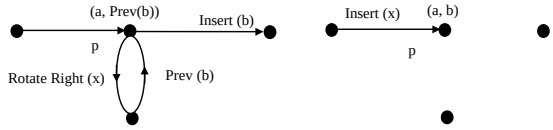
۲- اگررکوردx، ازرکورد SmallItem کوچکترباشد، رکوردLargeItem، ازطریق فرزند سمت‌راست‌درج میشود ورکورد SmallItem برای‌چرخش‌ازطریق فرزند سمت‌چپ‌استفاده میگردد و سپس‌رکوردهای x و(Prev(LargeItem (رکورد LargeItem در فرزند سمت چپ این گره) بترتیب جای خالی رکورد های SmallItem و LargeItem را پر میکنند. نحوه اجرای این عمل در شکل زیر نشان داده شده است.



پس از اجرا عملگر درج قبل از اجرا عملگر درج

شکل۴

۳-اگـر رکـورد x از رکـورد SmallItem بزرگتـر و ازرکـورد LargeItem کـوچکتر باشـد، رکوردهای x و LargeItem بترتیب از طریق فرزندان سـمت چـپ و سـمت راسـت راسـت درج میشوند و سپس رکورد Prev (LargeItem) جـای خـالی LargeItem را پـر میکنـد. نحـوه اجرای این عمل در شکل زیر نشان داده شده است.



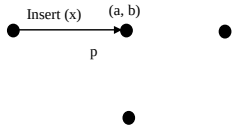
پس از اجرا عملگر درج قبل از اجرا عملگر درج

شکل۵

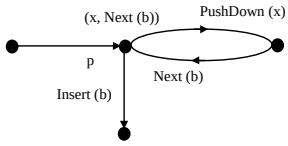
اگر مقدار Balance بزرگتر یا مساوی صـفر باشـد، رکـورد x از طریـق فرزنـد سـمت چـپ درج میشود که یکی از حالتهای زیر اتفاق میافتد.

۱- اگر رکورد x از رکورد SmallItem بزرگتر و از رکورد LargeItem کوچکتر باشد رکـورد x از طریق فرزند سمت چپ درج میشود.

۲- اگر رکورد x از رکـورد LargeItem بزرگتـر باشـد، رکـورد LargeItem از طریـق فرزنـد سمت چپ درج میشود ورکورد x برای چرخش از طریق فرزند سمت راست اسـتفاده میگـردد و سـپس جـای خـالی رکـورد LargeItem توسـط رکـورد Next (LargeItem) (رکـورد SmallItem در فرزند سمت راست این گره) پر میگردد. نحوه اجرای ایـن عمـل در شـکل زیـر نشان داده شده است.



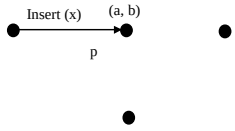
قبل از اجرا عملگر درج



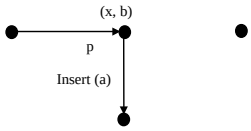
پس از اجرا عملگر درج

شکل ۶

۳- اگر رکورد x از رکـورد SmallItem کـوچکتر باشـد، رکـورد SmallItem از طریـق فرزنـد سمت چپ درج میشود و سپس جای خالی آن توسط رکورد x پر میگردد. نحوه اجرای این عمل در شکل زیر نشان داده شده است.



قبل از اجرا عملگر درج



پس از اجرا عملگر درج

شکل ۷

ب- اگر گره q یک گره میانی یا گره مرزچپ یا گره برگ باشـد، عملگـر درج رکـورد x را از پـدر سمت راست خود دریافت میکند. در صورتیکه گره q ، گره میانی خالی یا نیمه پـر یـا گـره بـرگ باشد رکورد x در گره q قرار میگیرد. در غیر اینصورت یکی از سه حالت زیر پیش میاید.

۱- اگر رکورد x از رکورد LargeItem بزرگتر باشد رکورد LargeItem از طریق فرزنـد سـمت چپ درج میشود وسپس جای خالی آن توسط رکورد x پر میگردد.

۲- اگر رکورد x از رکورد SmallItem کوچکتر باشدرکورد SmallItem از طریق فرزند سمت چپ درج میشود و سپس رکورد x جای خالی آنرا پر میکند.

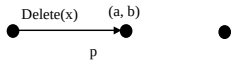
۳-اگر رکورد x ازرکورد SmallItem بزرگتر و ازرکورد LargeItem کوچکتر باشد رکورد x از طریق فرزند سمت چپ درج میشود.

عملگر حذف : نحوه اجرای این عملگر مشابه عملگر حذف در درخت جـستجوی دودوئـی اسـت یعنی با حذف رکورد x، رکورد Next (x) (کوچکترین رکورد بزرگتر از x) یـا رکـورد Prev (x) (بزرگترین رکورد کوچکتر از x) جای خالی x را پر میکنند. دراین عملگـر، در حـین حـذف یـک رکورد با ایجاد چرخش های لازم خواص توری جستجوی دودوئی کامل برقرار میگـردد. عملکـرد این عملگر برای گره های مرز بالا و گره های دیگر متفاوت است. نحوه اجرای این عملگر بـا توجـه به نوع گره بصورت زیر میباشد.

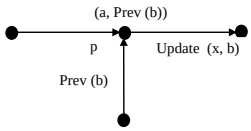
الف - اگر گره q یک گره مرز بالا باشد، عملگر حذف رکورد x را از پدر سمت چپ خـود دریافـت میکند. اگر گره q فرزندی نداشته باشد رکورد x از این گره حذف میگردد. در غیر اینصورت یکـی از چهار حالت زیر پیش آید.

۱- اگر رکورد x از رکورد LargeItem و متغیـر Balance از صـفر بزرگتـر باشـند رکـورد x از طریق فرزند سمت راست حذف میشود.

۲- اگر رکورد x از رکورد LargeItem بزرگتـر و متغیـر Balance کـوچکتر یـا مـساوی صـفر باشند، از طریق فرزند سمت راست رکورد های x و LargeItem بترتیب حذف ودر ج میـشوند و سـپس رکـورد Prev (LargeItem) جـای خـالی LargeItem راپرمیکننـد. نحوهاجرایاینعملدرشکل زیر نشان داده شده است.



قبل از اجرا عملگر درج

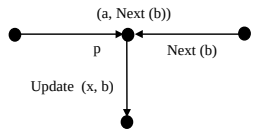


پس از اجرا عملگر درج

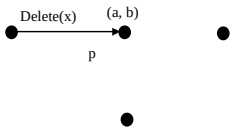
شکل ۸

۳- اگرکـورد x بزرگترازرکـورد SmallItem وکـوچکتر از رکـورد LargeItem و متغیـر Balance کوچکتر یا مساوی صفر باشد، رکورد x از طریق فرزند سمت چپ حذف میشود.

۴- اگرکورد x بزرگترازرکورد SmallItem وکوچکترازرکورد LargeItem و متغیـر Balance بزرگتراز صفر باشد، رکورد x از طریق فرزند سمت چـپ حـذف ورکـورد LargeItem از طریـق فرزند سمت چپ درج میگردد. جای خالی LargeItem توسـط رکـورد Next (LargeItem) پر میشود. نحوه اجرای اینعمل شکل زیر نشان داده شده است.



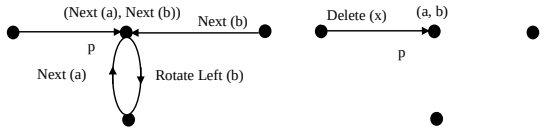
قبل از اجرا عملگر درج



پس از اجرا عملگر درج

شکل ۹

۵- اگر رکورد x مساوی رکورد SmallItem باشد، رکورد SmallItem حذف و سپس جای خالی آن توسط رکورد Next (SmallItem) پر میشود. اگر متغیر Balance بزرگتر از صفر باشد رکورد LargeItem برای چرخش ازطریق فرزند سمت چپ استفاده میشود وسپس جای خالی رکورد LargeItem توسط رکورد Next (LargeItem) پر میشود. نحوه اجرای این چرخش در شکل زیر نشان داده شده است.



پس از اجرا عملگر درج قبل از اجرا عملگر درج

شکل ۱۰

۶- اگر رکورد x مساوی رکورد LargelItem باشد، رکورد LargeItem حذف میشود. اگر متغیر Balance بزرگتر از صفر باشد جای خالی LargeItem توسط رکورد Next (LargeItem) و اگر متغیر Balance مساوی یا کوچکتراز صفر باشد جای خالی LargeItem توسط رکورد Prev (LargeItem) پر میشود.

ب- اگر گره q یک گره میانی، یا مرز چپ و یا برگ باشد، عملگرحذف رکورد x را از پدر سمت راست خود دریافت میکند. اگر گره q یک گره برگ، یک گره میانی نیمه پر و یا پر بدون فرزند باشد رکورد x از این گره حذف میشود. اگر گره q یک گره میانی دارای فرزند باشد یکی از سه حالت زیر پیش میاید.

۱- اگر رکورد x ازرکورد SmallItem بزرگتر و ازرکورد LargeItem کوچکتر باشد، رکورد x ازطریق فرزند سمت چپ حذف میشود.

۲- اگر رکورد x مساوی رکورد LargeItem باشد، رکورد LargeItem حذف و سپس جای خالی آن توسط رکورد Prev (LargeItem) پر میگردد.

۳- اگر رکورد x مساوی SmallItem باشد رکوردSmallItem حذف میشود و جای خالی آنرا رکورد Next (SmallItem) پر میکند.

عملگر جستجو : اگر رکورد با کلید x ، در توری ذخیره شده باشد عملگر جستجو رکورد دارای کلید x و در غیر اینصورت search-fail را بعنوان جواب تولید میکند. عملکرد این عملگر برای گره های مرز بالا و گره های دیگر متفاوت است. نحوه اجرای این عملگر با توجه به نوع گره بصورت زیر میباشد.

الف - اگر گره q یک گره مرز بالا باشد، عملگر جستجو را از پدر سمت چپ خود دریافت میکند. و یکی از چهار حالت زیر پیش میآید.

۱- اگر کلید x مساوی کلید یکی از رکوردهای ذخیره شده در گره q باشد، کلید x در توری وجود دارد و جستجو پایان میپذیرد.

۲- اگر کلید x بزرگتر از کلید رکورد LargeItem باشد، جستجو از طریق فرزند سمت راست انجام میشود.

۳- اگر کلید x کوچکتراز کلید رکورد SmallItem و کوچکتر از کلید رکورد LargeItem باشد، جستجو از طریق فرزند سمت چپ انجام میشود.

۴- اگر کلید x کوچکتراز کلید رکورد SmallItem یا گره q خالی باشد، کلید x در توری موجود نمیباشد و جستجو پایان میپذیرد.

ب- اگر گره q یک گره میانی یا مرز چپ یا برگ باشد، عملگر جستجو را از پدر سمت راست خود دریافت میکند و یکی از سه حالت زیر پیش میاید.

۱- اگر کلید x مساوی کلید یکی از رکوردهای ذخیره شده در گره q باشد، کلید x در توری وجود دارد و جستجو پایان میپذیرد.

۲- اگر کلید x بزرگتر از کلید رکورد SmallItem و کوچکتر از کلید رکورد LargeItem باشد، جستجو از طریق فرزند سمت چپ انجام میشود.

۳- اگر کلید x کوچکتراز کلید رکورد SmallItem یا بزرگتر از کلید رکورد LargeItem یا گره q خالی باشد، کلید x در توری موجود نمیباشد و جستجو پایان میپذیرد.

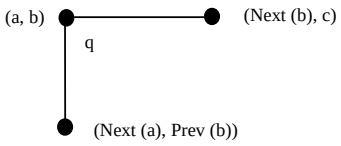
عملگرجستجوی رکورد دارای کوچکترین کلید: وظیفه این عملگر پیدا نمودن رکورد دارای کوچکترین کلید در توری میباشد. رکورد دارای کوچکترین کلید در متغیر SmallItem گره ریشه قرار دارد.

عملگرجستجوی رکورد دارای بزرگترین کلید: وظیفه این عملگر پیدا نمودن رکورد دارای بزرگترین کلید در توری میباشد. رکورد دارای بزرگترین کلید در سمت راست ترین گره مرز بالای غیر تهی قرار داردو با حرکت از ریشه به فرزند سمت راست ، این رکورد پیدا میشود.

عملگر حذف رکورد دارای کوچکترین کلید: وظیفه این عملگر پیدا نمودن رکورد دارای کوچکترین کلید و حذف آن از توری میباشد. رکورد دارای کوچکترین کلید در متغیر SmallItem گره ریشه قرار دارد. پس از حذف این رکورد، اگر ریشه دارای فرزندی باشد از چرخش های مشابه چرخش های عملگر حذف (شکل های ۸ الی ۱۰) برای پر نمودن جای خالی رکورد SmallItem استفاده میشود.

عملگرجستجوی رکورد بعد : وظیفه این عملگر در صورتیکه کلید x بزرگترین کلید توری نباشد پیدا نمودن کوچکترین رکورد بزرگتر از رکورد x (رکورد Next (x)) میباشد و در غیر اینصورت جواب LastKey را تولید میکند. عملکرد این عملگر برای گره های مرز بالا و گره های دیگر متفاوت است. این عملگر ابتدا گره ای را پیدا میکند که رکورد با کلید x در آن ذخیره شده باشد و سپس براساس نوع گره رکورد بعد را تعیین میکند. نحوه اجرای این عملگر با توجه به نوع گره بصورت زیر میباشد.

الف - اگر گره q یک گره مرز بالا باشد، عملگر رکورد بعد را از پدر سمت چپ خود دریافت میکند. ترتیب رکورد های ذخیره شده در این گره و فرزندانش بصورت زیر میباشد.



شکل ۱۱

برای اجرای این عملگر یکی از چهار حالت زیر پیش میآید.

۱- اگرگره q یک‌گره نیمه پر باشد، این رکورد دارای بزرگترین‌کلید میباشد.

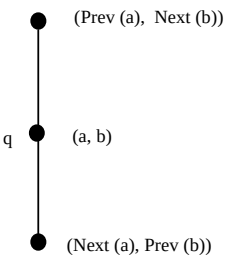
۲- اگرگره q یک‌گره پر بدون فرزند و کلید x مساوی کلیدرکورد SmallItem باشد، رکورد بعد، رکورد LargeItem گره q است.

۳- اگر گره q یک گره پر بدون فرزندسمت راست و کلید x مساوی کلید رکورد LargeItem گره q باشداین رکورد دارای بزرگترین کلید میباشد.

۴- اگرگره q یک‌گره‌پردارای فرزندسمت‌راست وکلید x مساوی کلید رکورد LargeItem باشد، رکورد بعد، رکورد SmallItem فرزندسمت‌راست‌این‌گره است.

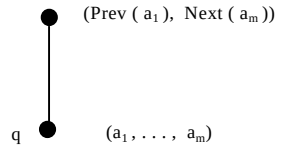
۵- اگرگره q یک‌گره پر دارای فرزند سمت چپ و کلید x مساوی‌کلیدرکورد SmallItem باشد، رکوردبعد، رکورد SmallItem فرزندسمت‌چپ این گره است.

ب - اگر گره q یک گره مرزمیانی یا مرز چپ باشد، عملگر رکورد بعد را از پدر سمت راست خود دریافت میکند. ترتیب رکورد های ذخیره شده در این گره و فرزند سمت چپ و پدر سمت راستش در شکل ۱۲ نشان داده شده است.



شکل ۱۲

برای اجرای این عملگر یکی از سه حالت زیر پیش میآید.
۱- اگرگره q یک‌گره نیمه پرباشد، رکورد بعد، رکورد LargeItem پدرسمت راست این‌گره می باشد.
۲- اگرکلید x مساوی‌کلیدرکورد LargeItem باشد، رکورد بعد، رکورد LargeItem پدر سمت راست این گره است.
۳- اگرگره q یک گره دارای فرزند سمت چپ و کلید x مساوی کلید رکورد SmallItem باشد، رکوردبعد، رکورد SmallItem فرزندسمت چپ این‌گره میباشد.
ج - اگر گره q یک برگ باشد، عملگر رکورد بعد را از پدر سمت راست خود دریافت میکند. ترتیب رکورد های ذخیره شده در این گره و پدر سمت راستش بصورت زیر میباشد.

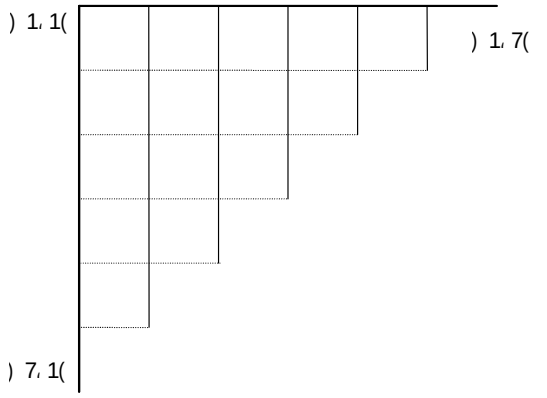


شکل ۱۳

فرض کنید که m رکورد $(a_1,a_2,...,a_m)$ در برگ q ذخیره شده باشد بطوریکه برای $k = 1, ..., m-1$ شرط $a_k < a_{k+1}$ برقرار باشد. برای اجرای این عملگر یکی ازدوحالت زیر پیش میاید.
۱- اگرکلید x مساوی کلید رکورد a_m باشد، رکورد بعد، رکورد LargeItem پدر سمت راست گره q است.
۲- اگرکلید x مساوی‌کلیدرکورد $a_k (k < m)$ باشد، رکورد بعد، رکورد a_{k+1} گره q است.

۳- پیاده سازی

دراین‌قسمت‌چگونگی‌پیاده‌سازی‌توری‌جستجوی‌دودوئی‌کامل روی یک کامپیوتر موازی با حافظه توزیع شده و ساختار ابر مکعب یا توری شرح داده میشود.
نیم توری بگونه ای در یک ابر مکعب نگاشت میشود که خاصیت همسایگی گره ها در آن حفظ شده باشد. یک روش شناخته شده برای برقراری خاصیت همسایگی در این نگاشت استفاده از کد گری انعکاسی است. این روش را میتوان بصورت زیر شرح داد: گره (x_1, x_2) در یک توری دو بعدی به گره S_1S_2 در ابر مکعب نگاشت داده میشود بطوریکه S_i ، x_i امین کد گری d_i بیتی میباشد [۸][۹][۳۰].
گره هایی از ابر مکعب که عضو نیم توری نگاشت شده نیستند تشکیل یک نیم توری $(n-1) \times (n-1)$ را میدهند که نیم توری آینه ای نامیده میشود. نیم توری آینه ای را میتوان برای ذخیره نمودن ساختمان داده دیگری و یا ترکیب نتایج تولید شده توسط برگ ها استفاده نمود. شکل زیر یک توری جستجوی دودوئی کامل ۷ × ۷ را نشان میدهد.



شکل ۱۴ : توری جستجوی دودوئی کامل ۷× ۷

مطابق شکل فوق کمانهای نیم توری به دو دسته زیر تقسیم میشوند.

کمانهای ارسال جواب: کمان متصل کننده هر گره به پدر سمت چپ خود مخصوص ارسال جواب میباشد. در شکل فوق این کمانها توسط خطوط نقطه چین نشان داده شده اند.
کمانهای دریافت عملگر: برای گره های مرزی بالا کمان متصل کننده هر گره با پدر سمت چپ خود و برای گره های دیگر کمان متصل کننده هر گره با پدر سمت راست خود مخصوص دریافت عملگرهای دیکشنری میباشد. در شکل فوق این کمانها توسط خطوط پر نشان داده شده است

هر گره در توری جستجوی دودوئی کامل دارای اطلاعات زیر میباشند.

Small Item	Left Count	Right Count	Large Item

الف) اطلاعات موجود در گره های مرز بالا

Small Item	Large Item

ب) اطلاعات موجود در گره های میانی و مرز چپ

شکل ۱۵: اطلاعات موجود در گره های توری جستجوی دودوئی کامل

SmallItem کوچکترین رکورد در هر گره و LargeItem بزرگترین رکورد در هر گره است. درصورتیکه پردازنده برگ باشد بدلیل اینکه بیشتر از یک رکورد دربرگ ها ذخیره میشوند رکوردهای موجود در گره های برگ بصورت مرتب شده ذخیره میشوند. RightCount و LeftCount بترتیب تعداد رکورد های درج شده از طریق فرزند سمت راست و فرزند سمت چپ میباشد. طبق قضیه ۸ مقدار متغیر Balance در پردازنده (1, k) بصورت زیر میباشد.

$$\text{Balance} = \begin{cases} \text{RightCount} - \left\lceil \frac{\text{LeftCount}}{2} \right\rceil \left(\left\lfloor \frac{\text{LeftCount}}{2} \right\rfloor + 1 \right) & \text{LeftCount} \le 2(n-k) \\ \text{RightCount} - (n-k)\left[\text{LeftCount} - (n-k+1)\right] & \text{LeftCount} > 2(n-k) \end{cases}$$

فرایندی که در هر یک از گره های ابر مکعب اجرا میشود از دوقسمت مجزا تشکیل شده است. قسمت اول فرایند متعلق به برنامه کاربردی است که روی همه گره های ابر مکعب اجرا میشود. قسمت دوم فرایند که اجرا کننده نامیده میشود عملگرهای ماشین دیکشنری را اجرا میکند. برای سادگی فرض میشود یک اجرا کننده میتواند دریافت و پردازش تعدادی پیغام را بطور همزمان انجام دهد. پیغام برای این عملگرها بوسیله فرآیند کاربردی که روی گره های ابر مکعب اجرا میشوند راه اندازی میگردد. این پیغام ها پس از دریافت به اجرا کننده ارسال میشود سپس اجرا کننده این پیغام ها را برای اجرا به گره ریشه ارسال میکند. عملگرهایی که بوسیله ریشه دریافت میشوند بوسیله پردازش لوله ای اجرا میگردند. در صورت نیاز نتیجه هر عمل به پردازنده درخواست کننده ارسال میشود. هر اجرا کننده میتواند یک عملگر را از یکی همسایگان خود یا قسمت برنامه کاربردی فرایند خود دریافت نماید.

ما از گرامر و معنای زیر برای دستورات SEND و RECEIVE استفاده میکنیم. دستور SEND (<processor>, <instruction>) ، دستور العمل <instruction> را برای‌اجرا به‌پردازنده <processor> ارسال میکند. پردازنده‌ای که دستورالعمل RECEIVE (<processor>, <information>) را اجرا میکند سبب دریافت اطلاعات <information> ازپردازنده <processor> و ارسال آن به فرایند درخواست کننده میگردد. در این مقاله فرض میشود که دستور RECEIVE دستور Blocking باشد (رویه ای که دستور RECEIVE را اجرا میکند منتظر میماند تا اطلاعات مورد نیاز دریافت شود). اگر مقدار<processor> برابر anyprocessor باشد دراینصورت‌دستورالعمل RECEIVEزمانی‌کامل‌میگردد‌که‌اطلاعات از یکی‌ازپردازنده‌ها‌دریافت شود. برنامه‌کاربردی‌که‌روی‌پردازنده p اجرا میشودعملگردیکشنری را توسط دستور SEND (root, <instruction>) به ریشه ارسال میکند که <instruction> یکی ازعملگرهای دیکشنری است. سپس‌اگردستورالعمل‌صادرشده نیاز به جواب داشته باشد دستور العمل RECEIVE (anyprocessor, <instruction-done>) را اجرا میکند و منتظر اتمام دستور میگردد. مقدار <instruction-done> میتواند برابر یکی از مقادیر search-success، search-fail، max-found، min-found، min-deleted، NextKey و LastKey باشد و anyprocessorشماره یک پردازنده در ابر مکعب یا توری میباشد.

برای اجرای عملگرهای دیکشنری از دستورات زیر استفاده شده است. برای درج رکورد با کلید x از دستور Insert (x)، برای جستجوی رکورد با کلید x از دستور Search (p,x)، برای حذف رکورد با کلید x از دستور Delete(x)، برای پیدا کردن رکورد دارای کوچکترین کلید ازدستور Min (p)، برای پیدا کردن رکورد دارای بزرگترین کلید از دستور Max (p)، برای حذف رکورد داری کوچکترین کلیدازدستور XMin(p) و برای رکورد بعد از دستور Next(p,x) استفاده میشود. p پردازنده ای است که عملگر را شروع میکند. حال‌به چگونگی پیاده سازی عملگرهای ماشین دیکشنری میپردازیم.

عملگر درج : عملکرداین‌عملگربرای‌پردازنده های مرز بالاو پردازنده های‌دیگر متفاوت‌است. نحوه‌اجرای‌این‌عملگربا توجه به نوع پردازنده بصورت‌زیر میباشد.

الف- اگرq یک‌پردازنده مرز بالا باشدعملگر Insert (x) را از پدرسمت‌چپ‌خود دریافت میکند. در اجـرای ایـن عملگـر، اگررکـورد x ، از طریـق فرزنـد سـمت راسـت درج شـود مقـدار متغیـر RightCount و در صورتیکه از طریق فرزند سمت چپ درج شـود مقـدار متغیـر LeftCount افزایش پیدا میکند. برای اجرای این عملگر پردازنده مرز بالا q قطعه کد زیر را اجرا میکند.

```
if  inst = Insert (x) then
   if  empty (q)  or q has one record then
      insert x into node q
   else  {q has two records}
      if Balance (q) < 0 then
         if x > LargeItem (q) then
            SEND (RC (q), Insert (x))
         else
            SEND (RC (q), Insert (LargeItem (q)))
            if x > SmallItem (q) then
               SEND (LC (q), RotateRight (x))
            else
               SEND (LC (q), RotateRight (SmallItem (q)))
               SmallItem (q) = x
            end if
         end if
         inc (RightCount (q))
      else
         if x < SmallItem (q) then
            SEND (LC (q), Insert (SmallItem (q)))
            SmallItem (q) = x
         else
            if x < LargeItem (q) then
               SEND (LC (q), Insert (x))
            else
               SEND (RC (q), PushDown (x))
               SEND (LC (q), Insert (LargeItem (q)))
            end if
         end if
         inc (LeftCount (q))
      end if
   end if
end if
```

ب- اگر پردازنده q یک پردازنده میانی، مرز چپ و یا بـرگ باشـد عملگـر Insert (x) را از پدر سمت راست خود دریافت و قطعه کد زیر را اجرا میکند.

```
if inst = Insert (x) then
   if leaf (q) or empty (q) or q has one record then
      insert x into node q
   else {q has two records}
      if x < SmallItem (q) then
         SEND (LC (q), Insert (SmallItem (q)))
         SmallItem (q) = x
      else
         if x < LargeItem (q) then
            SEND (LC (q), Insert (x))
         else
            SEND (LC (q), Insert (LargeItem (q)))
            LargeItem (q) = x
         end if
      end if
   end if
end if
```

عملگر حذف : عملکرد این عملگر برای پردازنده های مرز بـالا و پردازنـده هـای دیگـر متفـاوت است. نحوه اجرای این عملگر با توجه به نوع پردازنده بصورت زیر میباشد.

الف- اگر پردازنده q یک پردازنده مرز بالا باشـد عملگـر Delete (x) رااز‌پدرسـمت چـپ خـود دریافت میکند. در اجرای این عملگر، اگررکورد x ازطریق فرزند سمت راست حـذف شـود مقـدار متغیـر RightCount ودر صـورتیکه از طریـق فرزنـد سـمت چـپ حـذف شـود مقـدار متغیـر LeftCount کاهش پیدا میکند. برای اجرای این عملگر پردازنده q قطعه کد زیر را اجرا میکند.

```
if inst = Delete (x) then
   if LC (q) = null then
      delete x
   else
      if Balance (q) ≥ 0 then
         if x > LargeItem (q) then
            SEND (RC (q), Delete (x))
         else
            SEND (RC (q), PopupSmall)
            if SmallItem < x < LargeItem (q) then
               SEND (RC (q), Update (x, LargeItem (q)))
            else
               if x = SmallItem (q) then
                  SEND (LC (q), RotateLeft (LargeItem (q)))
               end if
            end if
         end if
         dec (RightCount (q))
      else { Balance (p) ≤ 0 }
         if x ≥ LargeItem (q) then
            SEND (LC (q), PopupLarge)
            if x > LargeItem (q) then
               SEND (RC (q), Update (x, LargeItem (q)))
            end if
         else
            if x > SmallItem then
               SEND (LC (q), Delete (x))
            else
               SEND (LC (q), PopupSmall)
            end if
         end if
         dec (LeftCount (q))
      end if
   end if
end if
```

ب- اگرپردازنده q یک‌پردازنده میانی، مرزچپ‌ویا برگ‌باشد عملگر Delete (x) را از پـدر سـمت راست خود دریافت وقطعه کد زیر را اجرا میکند.

```
if inst = Delete (x) then
   if  leaf (q) or LC (q) = null then
      delete x
   else
      if x = LargeItem (q) then
         SEND (LC (q), PopupLarge)
      else
         if x = SmallItem (p) then
            SEND (LC (p), PopupSmall)
         else
            SEND (LC (p), Delete (x))
         end if
      end if
   end if
end if
```

عملگر جستجو : عملکرد این عملگر برای پردازنده های مرز بالا و پردازنـدههای دیگـر متفـاوت است. نحوه اجرای این عملگر با توجه به نوع پردازنده بصورت زیر میباشد.

الف- اگر q پردازنده مرز بالا باشدعملگر Search(p, x) را از پـدر سـمت چـپ خـود دریافـت و قطعه کد زیر را اجرا میکند.

```
if inst = Search (p, x) then
   if  empty (q) or x < SmallItem (q) then
      SEND (p, search-fails)
   else
      if  x > LargeItem (q) then
         SEND (RC (q), Search (p, x))
      else
         if x  > SmallItem (q) then
            SEND (LC (q), Search (p, x))
         else
            SEND (p, search-success)
         end if
      end if
   end if
end if
```

اگر q یک پردازنده میانی یا مرز چپ یا برگ باشد عملگر Search(p,x) را از پدر سمت راسـت خود دریافت و قطعه کد زیر را اجرا میکند.

```
if inst = Search (p, x) then
   if leaf (q) then
      if  x is in q  then
         SEND (p, search-success)
      else
         SEND (p, search-fails)
      end if
   else
```

```
    if empty (q) or x < SmallItem (q) or x > LargeItem (q) then
      SEND (p, search-fails)
    else
      if SmallItem (q) < x < LargeItem (q) then
        SEND (LC (q), Search (p, x))
      else
        SEND (p, search-success)
      end if
    end if
  end if
end if
```

عملگر جستجوی رکورد دارای کوچکترین کلید: بدلیل اینکه رکورددارای کوچکترین کلیددرپردازندهریشه قرار دارد عملگر Min(p) تنها در پردازنده ریشه اجرا میشود. برای اجرای این عملگر پردازنده ریشه قطعه کد زیر را اجرا میکند.

```
if inst = Min (p) then
  SEND (p, min-found (SmallItem (root))
end if
```

عملگرجستجوی رکورد بعد : عملکرد این عملگر برای پردازنده های مرز بالا و پردازنده های دیگر متفاوت است. نحوه اجرای این عملگر با توجه به نوع پردازنده بصورت زیر میباشد.

الف- اگر q پردازنده مرز بالا باشد عملگرNext (p, x) را از پدر سمت چپ خود دریافت و قطعه کد زیر را اجرا میکند.

```
if inst = Next (p, x) then
  if q has one record then
    SEND (p, lastKey)
  else
    if q has no childs then
      if x = SmallItem (q) then
        SEND (p, NextKey (LargeItem (q)))
      else
        SEND (p, lastKey)
      end if
    else
      if x = SmallItem (q) then
        SEND (LC (q), SendSmall (p))
      else
        if x < LargeItem (q) then
          SEND (LC (q), Next (p, x))
        else
          if x = LargeItem (q) then
            if RC (q) ≠ null then
              SEND (RC (q), SendSmall (p))
            else
              SEND (p, last-record)
            end if
          else
            SEND (RC (q), Next (p, x))
          end if
        end if
      end if
    end if
  end if
end if
```

ب- اگر یک پردازنده میانی ، مرز چپ و یا برگ باشد، عملگر Next(p,x) را از پدر سمت راست خود دریافت و قطعه کد زیر را اجرا میکند.

```
if inst = Next (p, x) then
  if leaf (q) then
    if x is largest record of node q then
      SEND (RP (q), SendLarge (p))
    else
      SEND (p, NextKey (next record of x))
    end if
  else
    if x = LargeItem (q) then
      SEND (RP (q), SendLarge (p) )
    else { x = SmallItem (q) }
      if LC (q) = null then
        if q has one record then
          SEND (RP (q), SendLarge (p) )
        else
          SEND (p, NextKey (LargeItem (q)))
        end if
```

```
      else
        SEND (LC (q), SendSmall (p))
      end if
    end if
  end if
end if
```

عملگرجستجوی رکورد دارای بزرگترین کلید: بدلیل اینکه رکورد دارای بزرگترین کلید در پردازنده های مرز بالا ذخیره میشود، عملگر Max(p) تنها در این پردازنده ها اجرا میشود. برای اجرای این عملگر پردازنده های مرز بالا q قطعه کد زیر را اجرا میکنند.

```
if inst = Max (p) then
  if empty (RC (q)) then
    SEND (p, min-found (LargeItem (q))
  else
    SEND (RC (q), Max (p))
  end if
end if
```

عملگرحذف رکورد دارای کوچکترین کلید: بدلیل اینکه رکورددارای کوچکترین کلیددرپردازندهریشه قراردارد، عملگرXMin(p) تنهادرپردازندهریشه اجرا میشود. برای اجرای این عملگر پردازنده ریشه قطعه کد زیر را اجرا میکند.

```
if inst = XMin (p) then
  if LC (root) ≠ null or RC (root) ≠ null then
    if Balance (p) ≥ 0 then
      SEND (RC (root), PopupSmall)
      SEND (LC (root), RotateLeft (LargeItem (root)))
      dec (RightCount (root))
    else
      SEND (LC (root), PopupSmall)
      dec (LeftCount (root))
    end if
  end if
  SEND (p, min-deleted (SmallItem (root)))
end if
```

عملگر PopupSmall: این عملگر توسط عملگرDelete (x) استفاده میشود و نحوه عملکرد آن برای پردازنده های مرز بالا و پردازنده های دیگر متفاوت است. نحوه اجرای این عملگر با توجه به نوع پردازنده بصورت زیر میباشد.

الف- اگر q یک پردازنده مرز بالا باشد، عملگر PopupSmallرا از پدر سمت چپ خود دریافت میکند وسبب میگردد تا رکورد SmallItem از پردازنده q حذف و برای پر کردن جای خالی، رکورد LargeItem مربوط به پدر سمت چپ پردازنده q به آن پردازنده ارسال شود و سپس جای خالی رکورد SmallItem توسط رکورد Next (SmallItem) پر میگردد. اگر مقدار متغیر Balance بزرگتر ازصفرباشد، برایحفظ توازندرتوری، رکورد LargeItem برایچرخش به چپ بهفرزندسمتچپ ارسال میگردد و سپس جای خالی رکوردLargeItem توسط رکورد Next (LargeItem) (که توسط عملگر PopUpSmall مشخص میگردد) پر میشود. برای اجرای این عملگر پردازنده q قطعه کد زیر را اجرا میکند.

```
if inst = PopupSmall then
  SEND (LP (q), SetLarge (SmallItem (q)))
  if LC (q) = null then
    delete SmallItem (q)
  else
    if Balance (q) ≤ 0 null then
      SEND (LC (q), PopupSmall)
      dec (LeftCount (q))
    else
      SEND (RC (q), PopupSmall)
      SEND (LC (q), RotateLeft (LargeItem (q)))
      dec (RightCount (q))
    end if
  end if
end if
```

ب- اگر q یک پردازنده میانی، مرز چپ و یا برگ با شد عملگرPopupSmall را از پدر سمت راست خود دریافت میکند و باعث میشود تا رکورد SmallItem از این گره حذف شود و برای جایگزینی در رکورد SmallItem پردازنده پدر سمت راست q به آن پردازنده ارسال گردد. اگر پردازنده q، یک پردازنده میانی یا مرز چپ دارای فرزند باشد عملگر PopupSmall برای پر نمودن جای خای رکورد حذف شده SmallItem به فرزند سمت چپ ارسال میشود. برای اجرای این عملگر پردازنده میانی، مرز چپ و یا برگ q قطعه کد زیر را اجرا میکند.

```
if inst = PopupSmall then
  if leaf (q) then
```

```
    find the smallest value y in q and delete it
    SEND (RP (q), SetSmall (y))
  else
    SEND (RP (q), SetSmall (SmallItem (q)))
    if LC (q) = null  then
      delete SmallItem
    else
      SEND (LC (q), PopupSmall)
    end if
  end if
end if
```

عملگر PopupLarge: این عملگر توسط عملگر Delete (x) استفاده میشود و تنها در پردازنده های میانی، مرزچپ ویا برگ اجرا و باعث میشود تا رکورد LargeItem از این گره حذف و برای جایگزینی در رکورد LargeItem پدر سمت راست ارسال شود. اگر پردازنده q، یک پردازنده میانی و یا مرز چپ دارای فرزند باشد عملگر PopupLarge برای پر نمودن جای خای رکورد حذف شده LargeItem به فرزند سمت چپ ارسال میشود. برای اجرای این عملگر پردازنده میانی و یا مرز چپ یا برگ q قطعه کد زیر را اجرا میکند.

```
if inst = PopupLarge then
  if leaf (q) then
    find the largest value y in q and delete it
    SEND (RP (q), SetLarge (y))
  else
    SEND (RP (q), SetLarge (LargeItem (q)))
    if LC (q) = null then
      delete LargeItem (q)
    else
      SEND (LC (q), PopupLargel)
    end if
  end if
end if
```

عملگر PushDown (x) : عملگرPushDown (x)، توسط عملگر Insert (x) راه اندازی و تنها در پردازنده مرز بالا q اجرا میشود و سبب میگردد تا رکورد x از طریق این پردازنده درج و سپس کوچکترین رکورد در این پردازنده و فرزند های آن حذف و به پدر سمت چپ ارسال گردد. با توجه به مقدار رکورد x یکی از حالات زیر اتفاق میافتد.

۱- اگر پردازنده q خالی یا رکورد x از رکورد SmallItem کوچکتر باشد، رکورد x به پدر سمت چپ ارسال میشود.

۲- اگر پردازنده q پر یا نیمه پر و فرزندی نداشته باشد، رکورد SmallItem حذف و به پدر سمت چپ ارسال میشود و سپس رکورد x در این پردازنده درج میشود.

۳- اگر پردازنده q دارای فرزند سمت چپ ورکورد x بزرگترازرکورد LargeItem باشد رکورد SmallItem حذف و به پدر سمت چپ ارسال میگردد و رکورد LargeItem برای چرخش به چپ به فرزند سمت چپ و رکورد x توسط عملگر PushDown (x) به فرزند سمت راست ارسال میگردند و سپس جای خالی رکوردهای SmallItem و LargeItem بترتیب توسط رکوردهای Next (SmallItem) و Next (LargeItem) پر میشوند.

۳- اگرپردازنده q دارایفرزندسمت چپ ورکورد x کوچکترازرکورد LargeItem و بزرگتراز رکورد SmallItem باشدرکورد SmallItem حذفوبهپدرسمتچپ ارسال میگردد ورکورد x برایچرخش به چپ به فرزند سمت چپ ارسال میگردد. جایخالیرکورد mallItemتوسط رکورد Next (SmallItem) پر میشود.

برای اجرای این عملگر پردازنده q قطعه کد زیر را اجرا میکند.

```
if inst = PushDown (x) then
  if empty (q) or x <  SmallItem (q)  then
    SEND (LP (q), SetLarge (x) )
  else
    SEND (LP (q), SetLarge (SmallItem (q)))
    if LC (q) = null then
      insert x into q
    else
      if x  >  LargeItem (q) then
        SEND (LC (q), RotateLeft (LargeItem (q)))
        SEND (RC (q), PushDown (x))
      else
        SEND (LC (q), RotateLeft (x))
      end if
    end if
  end if
end if
```

عملگر Update (x, y): این عملگر توسط عملگرDelete (x) استفاده میشود و باعث میشود تا رکورد x از طریق این پردازنده حذف و رکورد y در جای مناسب خود درج شود. نحوه عملکرد آن برای پردازنده های مرز بالا و پردازنده های دیگر متفاوت است. نحوه اجرای این عملگر با توجه به نوع پردازنده بصورت زیر میباشد.

الف- اگر q یکپردازنده مرز بالا باشد، عملگرUpdate(x,y) را از پدرسمتچپ خود دریافت میکند. با توجه به طریقه استفاده عملگرUpdate (x, y)، رکورد y همواره کوچکتر از رکورد SmallItem پردازنده q است. بنابراین محل درج رکورد y در متغیر SmallItem پردازنده q میباشد. با توجه به مقدار رکورد x یکی از حالات زیر اتفاق میافتد.

۱- اگر رکورد x مساوی رکورد SmallItem باشد، رکورد y جایگزین رکورد SmallItem میشود.

۲- اگر رکورد x مساوی رکورد LargeItem باشد رکورد SmallItem برای چرخش به راست به فرزند سمت چپ ارسال میشود و سپس رکوردهای y و Prev (LargeItem) بترتیب جای خای رکورد های SmallItem و LargeItem را پر میکنند.

۳- اگر رکورد x بزرگتر از رکورد SmallItem و کوچکتر از رکورد LargeItem باشد رکوردهای x و SmallItem توسط عملگر Update (x, SmallItem) برای حذف رکورد x و درج رکورد SmallItem به فرزند سمت چپ ارسال میشوند و سپس رکورد y جای خای رکورد SmallItem را پر میکنند.

برای اجرای این عملگر پردازنده q قطعه کد زیر را اجرا میکند.

```
if inst = Update (x, y) then
  if SmallItem (q) < x  <  LargeItem (q)then
    SEND (LC (q), Update (x, SmallItem (q)))
  else
    if x ≥ LargeItem (q) then
      SEND (LC (q), RotateRight (SmallItem (q)))
      if x > LargeItem (q) then
        SEND (RC (q), Update (x, LargeItem (q)))
      end if
    end if
  end if
  SmallItem (q) = y
end if
```

ب- اگر پردازنده q یک پردازنده میانی ، مرز چپ و یا برگ با شد عملگر Update (x, y) را از پدر سمت راست خود دریافت میکند. با توجه به نوع استفاده عملگر Update (x, y)، رکورد y همواره کوچکتر از رکورد SmallItem پردازنده q است. بنابراین محل درج رکورد y در متغیر SmallItem پردازنده q میباشد. با توجه به مقدار رکورد x یکی از حالات زیر اتفاق میافتد.

۱- اگر رکورد x مساوی رکورد SmallItem باشد، رکورد y جایگزین رکورد SmallItem میشود.

۲- اگر x رکورد مساوی رکورد LargeItem باشد، رکورد SmallItem برای چرخش به راست به فرزند سمت چپ ارسال میشود و سپس رکوردهای y و Prev (LargeItem) بترتیب جای خای رکورد های SmallItem و LargeItem را پر میکنند.

۳- اگر رکورد x بزرگتر از رکورد LargeItem باشد، رکورد SmallItem برای چرخشبهراستبهفرزندسمتچپورکوردهای x وLargeItem توسط عملگر Update (x, LargeItem) برای حذف رکورد x و درج رکورد LargeItem به فرزند سمت راست ارسال میشوند و سپس رکوردهای y و Prev(LargeItem) بترتیب جای خای رکورد های SmallItem و LargeItem را پر میکنند.

۴- اگر رکورد x بزرگتر از رکورد SmallItem و کوچکتر از رکورد LargeItem باشد، رکوردهای x و SmallItem توسط عملگر Update (x, SmallItem) برای حذف رکورد x و درج رکورد SmallItem به فرزند سمت چپ ارسال میشوند و سپس رکورد y جای خای رکورد SmallItem را پر میکنند.

برای اجرای این عملگر پردازنده q قطعه کد زیر را اجرا میکند.

```
if inst = Update (x, y) then
  if leaf (q) or LC (q) = null then
    delete x from q and insert y into node q
  else
    if x  =  LargeItem (q) then
      SEND (LC (q), RotateRight (SmallItem (q)))
    else
      if x > SmallItem (q) then
        SEND (LC (q), Update (x, SmallItem (q)))
      end if
    end if
    SmallItem (q) = y
```

```
    end if
end if
```

عملگر چرخش به راست: این عملگر فقط در پردازنده میانی، مرز چپ ویا برگ q اجرا میشود و سبب میشود تا رکورد x از طریق این پردازنده درج شود و سپس رکورد LargeItem ازپردازنده q حذف و به پدر سمت راست ارسال گردد. اگر پردازنده q، یک پردازنده برگ یا یک پردازنده غیر برگ بدون فرزند باشد پس از درج رکورد x ، بزرگترین رکورد این پردازنده به پدر سمت راست ارسال میشود. با توجه به مقدار رکورد x یکی از حالات زیر اتفاق میافتد.

۱- اگر پردازنده q خالی یا رکورد x از رکورد LargeItem بزرگتر باشد، رکورد x به پدر سمت راست ارسال میشود.

۲- اگررکورد x کوچکتر از رکورد SmallItem باشد رکورد LargeItem حذف و به پدر سمت راست و رکورد SmallItem برای چرخش به راست به فرزند سمت چپ ارسال میشوند و سپس جای خالی رکوردهای SmallItem و LargeItem بترتیب توسط رکوردهای x و Prev (LargeItem) پر میشوند.

۳- اگر رکورد x کوچکتر از رکورد LargeItem و بزرگتر از رکورد SmallItem باشد رکورد LargeItem حذف و به پدر سمت راست و رکورد x برای چرخش به چپ به فرزند سمت چپ ارسال میگردند و سپس جای خالی رکورد LargeItem توسط رکورد Prev (LargeItem) پر میشود.

برای اجرای این عملگر پردازنده q قطعه کد زیر را اجرا میکند.

```
if inst = RotateRight (x) then
   if leaf (q) or LC (q) = null then
      insert x into node q
      find the largest value y and delete it from node q
      SEND (RP (q), SetLarge (y))
   else
      if x > LargeItem (q) then
         SEND (RP (q), SetLarge (x))
      else
         SEND (RP (q), SetLarge (LargeItem (q)))
         if  x  < SmallItem (q) then
            SEND (LC (q), RotateRight (SmallItem (q)))
            SmallItem (q) = x
         else
            SEND (LC (q), RotateRight (x))
         end if
      end if
   end if
end if
```

عملگر چرخش به چپ: این عملگر فقط در پردازنده میانی، مرز چپ ویا برگ q اجرا میشود و سبب میشود تا رکورد x از طریق پردازنده درج شود و سپس رکورد SmallItem پردازنده q حذف و به پدر سمت راست ارسال گردد. اگر پردازنده q، یک پردازنده برگ و یا یک پردازنده غیر برگ بدون فرزند باشد پس از درج رکورد x ، بزرگترین رکورد این پردازنده به پدر سمت راست ارسال میشود. با توجه به مقدار رکورد x یکی از حالات زیر اتفاق میافتد.

۱- اگر پردازنده q خالی و یا رکورد x کوچکتر از رکورد SmallItem باشد، رکورد x به پدر سمت راست ارسال میشود.

۲- اگررکورد x بزرگتر از رکورد LargeItem باشد، رکورد SmallItem حذف و به پدر سمت راست و رکورد LargeItem برای چرخش به چپ به فرزند سمت چپ ارسال میشوند و سپس جای خالی رکوردهای SmallItem و LargeItem بترتیب توسط رکوردهای x و Prev (LargeItem) پر میشوند.

۳- اگر رکورد x کوچکتر از رکورد LargeItem و بزرگترازرکورد SmallItem باشد، رکورد SmallItem حذف و به پدر سمت راست و رکورد x برای چرخش به راست به فرزند سمت چپ ارسال میگردند و سپس جای خالی رکورد SmallItem توسط رکورد Next (SmallItem) پر میشوند.

برای اجرای این عملگر پردازنده q قطعه کد زیر را اجرا میکند.

```
if inst = RotateLeft (x) then
   if leaf (q) or LC (q) = null then
      insert x into node q
      find the largest value y and delete it from node q
      SEND (RP (q), SetSmall (y))
   else
      if x  <  SmallItem (q) then
         SEND (RP (q), SetSmall (x))
      else
         SEND (RP (q), SetSmall (SmallItem (q)))
         if x > LargeItem (q) then
            SEND (LC (q), RotateLeft (LargeItem (q)))
            LargeItem (q) = x
         else
            SEND (LC (q), RotateLeft (x))
         end if
      end if
   end if
end if
```

عملگرهای SetSmall، SetLarge، SendSmall و SendLarge که در زیر آمده اند توسط همه پردازنده ها اجرا میشوند.

```
if inst = SetSmall (x) then
   SmallItem = x
end if

if inst = SetLarge (x) then
   LargeItem = x
end if

if inst = SendSmall (p) then
   SEND (p, NextKey (SmallItem (q))
end if

if inst = SendLarge (p) then
   SEND (p, NextKey (LargeItem (q))
end if
```

قضیه ۸: عملگر درج(درج رکورد x) خواص توری جستجوی دودوئی را حفظ میکند.

اثبات : اثبات توسط استقرا روی تعداد رکوردهای نیم توری(N) انجام میگیرد.

پایه استقرا: اگر N = 0 یا N = 1 باشد درج رکورد x در نیم توری خصوصیات توری جستجوی دودوئی را حفظ میکند زیرا ریشه نیم توری دارای صفر یا یک رکورد است و رکورد x در ریشه قرار میگیرد.

فرض استقرا: اگر N = K (K > 1) باشد فرض میکنیم که خصوصیات توری جستجوی دودوئی کامل برقرار باشد.

حکم استقرا: اگر نیم توری دارای N = K (K > 1) رکورد و خصوصیات توری جستجوی دودوئی کامل برقرار باشد در اینصورت درج رکورد x خصوصیات توری جستجوی دودوئی کامل را حفظ میکند.

با توجه به تعریف توری جستجوی دودوئی کامل برای درج رکورد x یکی از دو حالت زیر پیش میاید.

الف- اگر Balance < 0 باشد یکی از چهارحالت زیر پیش میاید.

۱- اگر نیم توری سمت چپ یک توری جستجوی دودوئی کاملا پر با گرههای سطح آخر پر وهمچنین نیم توری سمت راست توری جستجوی دودوئی کامل باشد. با توجه به رویهم پوشانی دو نیم توری سمت چپ و سمت راست، تنها یک گره نیمه پر در نیم توری سمت راست وجود دارد و یک رکورد باید درآن (در نیم توری سمت راست درج گردد) قرار گیرد تا نیم توری سمت راست به یک توری جستجوی دودوئی کاملا پر تبدیل گردد. با توجه به مقدار رکورد x یکی ازسه حالت زیر پیش میاید.

۱-۱- اگر x > LargeItem باشد، رکورد x ، باید در نیم توری سمت راست درج شود.

۱-۲- اگر x < SmallItem باشد، رکورد LargeItem ، باید نیم توری سمت راست ورکورد SmallItem در نیم توری سمت چپ درج شود و سپس رکورد های x و Prev (LargeItem) بترتیب جای خالی رکورد های SmallItem و LargeItem را پر کنند.

۱-۳- اگر SmallItem < x < LargeItem باشد، رکوردهای x و LargeItem باید بترتیب در نیم توری های سمت چپ و سمت راست راست درج شوند و سپس رکورد Prev (LargeItem) جای خالی LargeItem را پر کند.

۲- اگر نیم توری سمت چپ ، یک توری جستجوی دودوئی کاملا پر با گره های سطح آخر نیمه پر و نیم توری سمت راست ، توری جستجوی دودوئی کامل باشد. با توجه به رویهم پوشانی دو نیم توری سمت چپ و سمت راست، تنها یک گره خالی در نیم توری سمت راست وجود دارد و یک رکورد باید درآن (در نیم توری سمت راست درج گردد) قرار گیرد تا نیم توری سمت راست به یک توری جستجوی دودوئی کاملا پر تبدیل گردد. با توجه به مقدار رکورد x یکی ازسه حالت زیر پیش میاید.

۲-۱- اگر x > LargeItem باشد، رکورد x ، باید در نیم توری سمت راست درج شود.

۲-۲- اگر x < SmallItem باشد، رکورد LargeItem ، باید نیم توری سمت راست ورکورد SmallItem در نیم توری سمت چپ درج شود و سپس رکورد های x و Prev (LargeItem) بترتیب جای خالی رکورد های SmallItem و LargeItem را پر کنند.

۲-۳- اگر SmallItem < x < LargeItem باشد، رکوردهای x و LargeItem باید بترتیب در نیم توری های سمت چپ و سمت راست راست درج شوند و سپس رکورد Prev (LargeItem) جای خالی LargeItem را پر کند.

۳- اگر نیم توری سمت چپ، یک توری جستجوی دودوئی کامل با عمق m-1 و نیم توری سمت راست ، توری جستجوی دودوئی کاملا پر با عمق m-2 و گره های سطح آخر پر باشد. برای حفظ خواص توری جستجوی دودوئی کامل، رکورد باید در نیم توری سمت چپ درج گردد. با توجه به رویهم پوشانی دو نیم توری و با توجه به اینکه گره های مرزبالا، عملگر را تنها از پدر سمت چپ خود دریافت میکنند یک رکورد باید در نیم توری سمت راست درج شود که یکی ازسه حالت زیر پیش میاید.

۳-۱- اگر x > LargeItem باشد، رکور x، بایددر نیم توری‌سمت‌راست‌درج شود.

۳-۲- اگر x < SmallItem باشد، رکورد LargeItem، باید نیم توری سمت راست ورکورد SmallItem در نیم توری سمت چپ درج شود و سپس رکورد های x و Prev (LargeItem) بترتیب جای خالی رکورد های SmallItem و LargeItem را پر کنند.

۳-۳- اگر SmallItem < x < LargeItem باشد، رکوردهای x و LargeItem باید بترتیب در نیم توری های سمت چپ و سمت راست راست درج شوند و سپس رکورد Prev (LargeItem) جای خالی LargeItem را پر کند.

۴- اگر نیم توری سمت چپ، یک توری جستجوی دودوئی کامل با عمق m-1 (که گره های سطح آخر آن نیمه پر هستند) و نیم توری سمت راست ، توری جستجوی دودوئی کاملا پر با عمق m-1 و گره های سطح آخر نیمه پر باشد. برای حفظ خواص توری جستجوی دودوئی کامل رکورد باید در نیم توری سمت چپ درج گردد. با توجه به رویهم پوشانی دو نیم توری و با توجه به اینکه گره های مرزبالا عملگر را تنها از پدر سمت چپ خود دریافت میکنند یک رکورد باید در نیم توری سمت راست درج شود که یکی ازسه حالت زیر پیش میاید.

۴-۱- اگر x > LargeItem باشد، رکورد x ، باید در نیم توری سمت راست درج شود.

۴-۲- اگر x < SmallItem باشد، رکورد LargeItem ، باید نیم توری سمت راست ورکورد SmallItem در نیم توری سمت چپ درج شود و سپس رکورد های x و Prev (LargeItem) بترتیب جای خالی رکورد های SmallItem و LargeItem را پر کنند.

۴-۳- اگر SmallItem < x < LargeItem باشد، رکوردهای x و LargeItem باید بترتیب در نیم توری های سمت چپ و سمت راست راست درج شوند و سپس رکورد Prev (LargeItem) جای خالی LargeItem را پر کند.

ب- اگر Balance = 0 باشد نیم توری های سمت چپ و راست، توری جستجوی دودوئی کاملا پر هستند و رکورد جدید باید در نیم توری سمت چپ درج شود که یکی از سه حالت زیر پیش میاید.

۱- اگر x > LargeItem باشد، باید رکورد LargeItem در نیم توری سمت چپ ورکورد x در نیم توری سمت راست درج گردد و سپس جای خالی رکورد LargeItem توسط رکورد Next (LargeItem) پر گردد.

۲- اگر x < SmallItem باشد، باید رکورد SmallItem در نیم توری سمت چپ درج و سپس جای خالی آن توسط رکورد x پر گردد

۳-اگر SmallItem < x < LargeItem باشد، رکورد x ، باید در نیم توری سمت چپ درج شود.

با توجه به اینکه رویه فوق توازن در ریشه و همچنین مرتب بودن رکوردها را حفظ میکند بنابراین عملگر درج خواص توری جستجوی دودوئی کامل را در ریشه حفظ میکند. حال باید ثابت نماییم که درج در نیم توری های سمت چپ و سمت راست خواص توری جستجوی دودوئی کامل را حفظ میکنند. درج در نیم توری سمت راست طبق رویه فوق انجام میپذیرد بنابراین خواص حفظ میگردد. بدلیل اینکه دو نیم توری سمت چپ و سمت راست در یک ستون نیم توری رویهم پوشانی ندارند و کامل بودن توری تنها گره های مرز بالا بررسی میگردد بنابراین کافی است اثبات نماییم که درج در این ستون خواص توری (ترتیب رکورد ها)را حفظ میکند. در صورتیکه ریشه نیم توری گره خالی یا نیمه پر یا برگ باشد رکورد x باید در ریشه قرار گیرد. در غیر اینصورت یکی از سه حالت زیر پیش میاید.

۱- اگر x > LargeItem باشد، باید رکورد LargeItem در نیم توری سمت چپ درج وسپس جای خالی آن توسط رکورد x پر گردد.

۲- اگر x < SmallItem باشد، باید رکورد SmallItem در نیم توری سمت چپ درج و سپس جای خالی آن توسط رکورد x پر گردد.

رکورد SmallItem در نیم توری سمت چپ درج و سپس جای خالی آن توسط رکورد x پر گردد

۳-اگر SmallItem < x < LargeItem باشد، رکورد x ، باید در نیم توری سمت چپ درج شود.■

قضیه ۹: عملگر حذف خواص توری جستجوی دودوئی را حفظ میکند.

اثبات قضیه فوق در [۵] آمده است.

قضیه ۱۰: عملگر حذف رکورد دارای کوچکترین کلید خواص توری جستجوی دودوئی را حفظ میکند.

اثبات: فرض کنید که K_{min} کوچکترین کلید موجود در توری جستجوی دودوئی کامل باشد. با توجه به توضیح و قطعه کد داده شده برای این عملگر، روشن است که این عملگر فقط از قسمت کوچکی از عملگر حذف که موبوط به حالت Delete (K_{min}) میباشد استفاده میکند و با توجه به قضیه ۷ این عملگر خصوصیات توری جستجوی دودوئی کامل را حفظ میکند.

قضیه ۱۱: زمان پاسخ عملگرهای رکورد بعد و رکورد دارای بزگترین کلید O(n) است.

اثبات: عملگر های فوق از دومرحله جستجو و تولید جواب تشکیل میشوند و زمان پاسخ آنها از مجموع زمان اجرای این دو مرحله تشکیل میگردد. با توجه به قضیه ۵ زمان پاسخ عملگر جستجو O(n) است و بدلیل اینکه تولید جواب نیاز به یک ارتباط دارد بنابراین زمان لازم برای تولید جواب O (1) است در نتیجه زمان پاسخ این عملگرها O(n) است.

قضیه ۱۲: زمان پاسخ عملگرهای رکورد دارای کوچکترین کلید و حذف رکورد دارای کوچکترین رکورد O (1) میباشد.

اثبات: با توجه به اینکه رکورد دارای کوچکترین کلید در پردازنده ریشه قرار دارد زمان پاسخ این دو عملگر هرکدام O (1) است.

۴- سازگاری درساختمان داده موازی

در ادامه بحث سازگاری در ساختمان داده های موازی تعریف میشود و سپس سازگاری در توری جستجوی کامل مورد بررسی قرار خواهد گرفت.

ساختمان داده موازی سازگار: یک ساختمان داده موازی را سازگار میگویند اگر نتیجه تولید شده توسط اجرای موازی هر ترتیب از عملگرها روی ساختمان داده با نتیجه تولید شده توسط اجرای متوالی آن ترتیب روی ساختمان داده یکسان باشد.

قضیه ۱۳: پیاده سازی توری جستجوی دودوئی کامل روی توری یا ابر مکعب سازگار نیست.

اثبات: فرض کنید عملگر درج و سپس عملگر جستجو بترتیب وارد توری شوند. اگر عملگر درج سبب چرخش در توری شود نتیجه عملگر جستجو قبل و بعد از اتمام عملگر درج میتواند متفاوت باشد که منجر به ناسازگاری ساختمان داده خواهد شد.

در ادامه بحث رویه ای بیان میگردد که با اجرای آن سازگاری پیاده سازی توری جستجوی دودوئی کامل روی توری یا ابر مکعب تضمین میگردد. این رویه بصورت زیر میباشد. عملگر های دیکشنری برای اجرا به ریشه توری فرستاده میشوند و در پردازنده ریشه در یک صف قرار میگیرند. پردازنده ریشه علاوه بر قرار دادن عملگر در انتهای صف ، تاخیر مورد نیاز برای شروع عملگر بعدی را محاسبه و با این عملگر در صف حفظ میکند. تاخیر مورد نیاز برای عملگر بعد بصورت زیر خواهد بود.

۱- در صورتیکه عملگر بعد از عملگر Next (p, x) یکی از عملگرهای پرس و جو باشددر اینصورت این عملگر میتواند بدون تاخیراجرا شود.

۲- عملگربعد از عملگر درج بایستی با دو تاخیر آغاز شود. یک واحد از این تاخیر بخاطر عمل چرخشی است که در بعضی موارد در موقع انجام عملگر درج به آن نیاز میباشد.

۳- عملگر بعد از عملگر های حذف و حذف رکورد دارای کوچکترین رکورد بایستی با دو واحد تاخیر آغاز شود. یک واحد از این تاخیر بخاطر عمل چرخشی است که در بعضی موارد در موقع انجام عملگر درج به آن نیاز میباشد.

۴- عملگر بعد از عملگر رکورد بعد باید با دو واحد تاخیر آغاز شود. یک واحد از این تاخیر مربوط به حالت زیر میباشد. اگرکلید x مساوی کلید رکورد LargeItem یک پردازنده میانی یا مرز چپ (با توجه به شکل ۱۲) یا مساوی کلید بزرگترین رکورد یک پردازنده برگ (با توجه به شکل ۱۳) باشدرکورد بعد دررکورد LargeItem پدر سمت راست این پردازنده قرار دارد و بهمین دلیل نیاز به یک ارتباط در جهت عکس حرکت عملگرهای ساختمان داده دارد در نتیجه یک تاخیر اضافه مورد نیاز می باشد.

۵- بخاطر اینکه عملگرهای جستجو، کوچکترین کلید و بزرگترین کلید نیاز به عمل چرخش و ارتباط اضافی ندارند به بیش از یک واحد تاخیر بین این عملگرها و عملگر بعدی نیاز نمیباشد.

قضیه ۱۴: توان عملیاتی توری جستجوی دودوئی کامل از مرتبه O (1) است.

اثبات: بدلیل اینکه تمامی عملگر ها برای اجرا به ریشه ارسال میگردند و ریشه در هر زمان قادر به راه اندازی بیش از یک عملگر نمیباشد، توان عملیاتی این ماشین دیکشنری O (1) است.

۵- نتیجه گیری

در این مقاله یک ساختمان داده موازی بنام توری جستجوی دودوئی کامل برای یک ماشین دیکشنری بمنظور پیاده سازی در کامپیوترهای موازی با حافظه توزیع شده پیشنهاد شد ه است. در این ساختمان داده زمان پاسخ عملگرهای جستجو، بزرگترین کلید و کلید بعد از مرتبه O (n) و زمان پاسخ عملگرهای کوچکترین کلید و حذف کوچکترین کلید از مرتبه O (1) میباشد. پس از شروع عملگرهای درج، حذف، حذف کوچکترین کلید و رکورد بعد نیاز به دو واحد تاخیر برای شروع عملگر بعدی میباشد در حالیکه عملگرهای کوچکترین کلید، بزرگترین کلید و جستجو نیاز به یک واحد تاخیر دارند.

توری جستجوی دودوئی کامل با سه ماشین Youn و Young، ماشین Schmeck و Schroder و ماشین Dehne و Santoro مقایسه گردیده است که نتایج آن در جدول ۱ آمده

است. با توجه این جدول و مباحث انجام گرفته مشخص میگردد که ماشین پیشنهادی در این مقاله بهتر از سه ماشین دیگر عمل میکند. زیرا

۱- زمان پاسخ توری جستجوی دودوئی کامل حدود نصف زمان پاسخ توری هشت و شش ضلعی است.

۲- با توجه به اینکه شبکه پخش عملگرها یک مسیر یکتا بین ریشه و گره های دیگرتوری جستجوی دودوئی کامل ایجاد میکند، عملگرهای ماشین دیکشنری بصورت متوالی وارد گره ها میگردند و هیچ حالتی اتفاق نمی افتد که یک عملگر از دو مسیر مختلف و همزمان به یک پردازنده برسد و این باعث میگردد تا سازگاری در توری جستجوی دودوئی کامل بسادگی قابل پیاده سازی باشد

۳- دریافت عملگرها و ارسال جوابها در توری جستجوی دودوئی کامل از طریق دو شبکه غیر مجزا انجام میشود که باعث کم شدن ترافیک و برخورد پیغام ها میگردد.

۴- توری جستجوی دودوئی کامل قابل جا دادن در ابر مکعب میباشد. روشی برای جای دادن توری هشت و شش ضلعی در ابر مکعب تا آنجایی که نویسندگان این مقاله اطلاع دارند هنوز گزارش نشده است.

۵- تعداد اتصالات موجود درتوری جستجوی دودوئی کامل کمتر از توری هشت و شش ضلعی میباشد که پیاده سازی VLSI آنرا ساده تر مینماید.

جدول ۱:مقایسه بین فاصله شروع دو عملگردر توری جستجوی دودوئی کامل و ماشینهای دیگر

عملگر	Insert	Max	XMin	Next	Min	Delete	Search
توری جستجوی دودوئی کامل	O(1)	O(1)	O(1)	O(1)	O(1)	O(1)	O(1)
ماشین Youn و Young	O(1)	*	*	*	O(1)	O(1)	O(1)
ماشین Schmeck و Schroder	O(1)	*	*	*	O(1)	O(1)	O(1)
ماشین Dehne و Santoro	O(1)	*	*	*	O(1)	O(1)	O(1)

جدول ۲: مقایسه بین زمان پاسخ عملگرها در توری جستجوی دودوئی کامل و ماشین های دیگر

عملگر	Search	Max	XMin	Next	Min
توری جستجوی دودوئی کامل	O(n)	O(n)	O(1)	O(n)	O(1)
ماشین Youn و Young	O(n)	*	*	*	O(1)
ماشین Schmeck و Schroder	O(n)	*	*	*	O(1)
ماشین Dehne و Santoro	O(n)	*	*	*	O(1)

معیارهایی که توسط * نشانداده شده اند گزارش نشده اند

مراجع

[1] A. V. Aho, J. E. Hopcraft, and J. D. Ullman, The Design and Analysis of Computer Algorithm, Addison Weley, 1974.

[2] M. J. Atallah, F. Dehne, R. Millers, and A. Rau-Chaplin, "Multisearch Techniques: Parallel Data Structures on Mesh-Connected Computers", Journal of Parallel and Distributed Computing, Vol. 20, pp. 1-13, 1994.

[3] M. J. Atallah and S. R. Kosaraju, "A Generalized Dictionary Machine for VLSI", IEEE Trans. on Computers, Vol. 34, No. 2, pp. 151-155, 1985.

[4] H. Beigy and M. R. Meybodi, "Notes on Binary Search Mesh: A Concurrent Data Structure for Hypercube Computer", Technical Report 3CE98, Computer Eng. Dept., Amirkabir University, Tehran, Iran, 1998.

[5] H. Beigy and M. R. Meybodi, "Complete Binary Search Mesh: A Concurrent Data Structure for Distributed Memory Parallel Computer", Technical Report 4CE98, Computer Eng. Dept., Amirkabir University, Tehran, Iran, 1998.

[6] J. L. Bentley and H. T. Kung, "A Tree Machine for Searching Problems", Proc. of Int. Conf. on Parallel Processing, Augest 1979.

[7] J. Biswas and J. C. Browne, "Simultaneous Update of Priority Structures", Proc. of Int. Conf. on Parallel Processing, pp. 124-131, 1987.

[8] J. E. Brandenburg and D. S. Scott, "Embedding of Comunication Tree and Grid into Hypercube", IPSC Technical Reports, 1986.

[9] D. P. Breteskas and J. N. Tsitsiklis, Parallel and Distributed Computation, Printice-Hall, 1989.

[10] W. J. Dally, A VLSI Architecture for Concurrent Data Structures, Kluwer Academic Publishers, 1987.

[11] F. Dehne and N. Santoro, "An Improved New Embedding for VLSI Dictionary Machines on Meshes", Proc. of Int. Sympos. Computer Applications in Design, Simulation, and Analysis, pp. 113-116, 1989.

[12] F. Dehne and N. Santoro, "Optimal VLSI Dictionary Machines on Meshes", Proc. of Int. Conf. on Parallel Processing, pp. 832-840, August 1987.

[13] C. S. Ellis, "Concurrent Search and Insertion in 2-3 Trees", Acta Information, Vol. 14, pp. 63-86, 1980.

[14] C. S. Ellis, " Concurrent Search and Insertion in AVL Trees ", IEEE Trans. on Computers, Vol. 29, No. 9, pp. 811-817, 1980.

[15] A. L. Fisher, "Dictionary Machine With Small Number of Processors", Proc. of Int. Symp. on Computer Architecture, University of Michigan, Ann arbor, Michigan, pp. 151-156, 1984.

[16] J. Ghosh, S. K. Das, and A. John, "Concurrent Processing of Lineary Ordered Data Structures on Hypercube Multicomputers", IEEE Trans. on Parallel and Distributed Systems, Vol. 5, No. 9, pp. 898-911, 1994.

[17] H. T. Kung and P. L. Lehman, "Concurrent Manipulation of Binary Search Trees", ACM Trans. on Database Systems, Vol. 5, No. 3, pp. 354-382, 1980.

[18] S. K. Lee and H. Choi, "Embedding of Complete Binary Trees into Meshs with Row-Column Routing", IEEE Trans. on Parallel and Distributed Systems, Vol. 7, No. 5, pp. 493-497, 1996.

[19] P. L. Lehman and S. B. Yao, "Efficient Locking for Concurrent Operations on B-Trees", ACM Trans. on Database System, Vol. 6, No. 4, pp. 650-670, 1981.

[20] H. O. Li and D. K. Probst, "Optimal VLSI Dictionary Machines Without Compress Instructions", IEEE Trans. on Computers, Vol. 39, No. 5, pp. 676-, 1990.

[21] M. R. Meybodi, "Implementating Priority Queue on Hypercube Machine", Proc. of Fourth Parallel Processing Symp., University of Callifornia, Irvine, pp. 85-111, April 1990.

[22] M. R. Meybodi, "New Designs for Priority Queue Machine", Proc. of PARABASE-90, Miami, Florida, pp. 123-121, Mayl 1990.

[23] M. R. Meybodi, "Concurrent Data Structures for Hypercube Machine", Lecture Notes in Computer Science: Parallel Architecture and Languages, Springer Verlage, pp. 559-577, 1992.

[24] M. R. Meybodi, "Binary Search Mesh: A Concurrent Data Structure for Hypercube Computer", Proc. of ICEE-93, Amirkabir University, Tehran, Iran, pp. 601-607, May 1993.

[25] M. R. Meybodi, "Tree Structured Dictionary Machines for VLSI", Proc. of 23th Annual Modeling and Simulation Conf., School of Eng., University of Pittsburg, Pittsburg, Pennsylvania, pp. 703-724, May 1992.

[26] M. R. Meybodi, "Banyan Heap Machine", Proc. of IEEE 6th Int. Parallel Processing Sympos., Los Almitos, CA, pp. 224-231, 1992.

[27] A. R. Omondi and J. D. Brock, "Implementing a Dictionary on Hypercube Machine", Proc. of Int. Conf. on Parallel Processing, pp. 707-709, 1987.

[28] T. A. Ottman, A. L. Rosenberge, and L. J. Stockmeyer, "A Dictionary Machine (for VLSI)", IEEE Trans. on Computers, Vol. 31, No. 9, pp. 892-897, 1982.

[29] V. N. Rao and V. Kumar, "Concurrent Access to Priority Queue", IEEE Trans. on Computers, Vol. 37, No. 12, pp. 1657-1665, 1988.

[30] Y. Saad and M. Schultz, "Topological Properties of Hypercubes", IEEE Trans. on Computers, Vol. 37, No. 7, pp. 867-872, 1988.

[31] H. Schemeck and H. Schroder, "Dictionary Machines for Different Models of VLSI", IEEE Trans. on Computers, Vol. 34, No. 5, pp. 472-475, 1985.

[32] A. K. Somani and V. K. Agarwal, "An Unsorted VLSI Dictionary Machine", Proc. of 1983 Canadian VLSI Conf., University of Waterloo, Waterloo, 1983.

[33] A. K. Somani and V. K. Agarwal, "An Efficient VLSI Dictionary Machine", Proc. of 11th Annual Int. Symp. on Computer Architecture, University of Mishigan, Ann arbor, Mishigan, pp. 142-150.

[34] H. Y. Youn and J. Y. Lee, "An Efficient Dictionary Machine Using Hexagonal Processor Arrays", IEEE Trans. on Parallel and Distributed Systems, Vol. 7, No. 3, pp. 226-273, 1996.

[35] W. Zhang and R. E. Korf, "Parallel Heap Operations on EREW PRAM", Journal of Parallel and Distributed Computing, Vol. 20, pp. 248-255, 1994.

[36] D. E. Knuth, The Art of Computer Programming,_Volume 3, Addison Wesley, 1975.

1- Insert
2- Delete

3- Extract minimum
4- Extract maximum
5- Redundant
6- Modify operators
7- Query operations
8- Balance cube
9- Sorted chain
10- Banyan heap
11- Initiation interval
12- Response time
13- Throughput

١٤ - Mesh

15- Broadcast net
16- Snak net
17- IO processor
18- Linear net
19- Spanning net
20- Binary search mesh (BSM)
21- Balanced binary search mesh (BBSM)
22- Complete binary search mesh (CBSM)