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

حمید بیگی محمد رضا 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 زیجره در [۲۷]، مکعب متعادل [۱۰] و هیپ بایس [۲۳][۲۶] از آن جمله اند. مش 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 مش یا ابر مکعب میباشد. پیاده سازی مش جستجوی دودوئی انطور که در [۲۴] ارائه شده است دارای اشکالاتی میباشد که میتوان به ایجاد حفره در هنگام عمل درج، از بین رفتن خاصیت مش جستجوی دودوئی پس از عمل حذف، عدم تعادل مش و بایین بودن توان عملیاتی اشاره نمود. اشکالاتی مانند عدم تعادل در مش، تشکیل حفره در هنگام درج و ازبین رفتن خصیت مش جستجوی دودوئی پس از عمل حذف، توسط ساختمان داده مش جستجوی دودوئی متعادل برطرف شده است [۲]. در این مقاله مش جستجوی دودوئی کامل معرفی میگردد. این ساختمان داده بمنظور حل مشکلاتی که در مش جستجوی دودوئی و مش جستجوی دودوئی متعادل وجود دارد ارائه گردیده است. در این ساختمان داده، سطوح مختلف مش بترتیب پر میگردند و این خود باعث افزایش کارایی میگردد. در هنگام پیاده سازی این ساختمان داده روی مش 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 مش جستجوی دودوئی کامل با طرح های دیگر گزارش شده برای کامپیوترهای موازی با حافظه توزیع شده و ساختار مش [۱۱][۱۲][۳۱][۳۴]، نشان میدهد که مش جستجوی دودوئی کامل دارای کارایی بمراتب بالاتری میباشد.

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

۱- مقدمه

یک ماشین دیکشنری از پایگاه داده ای از زوج (کلید، رکورد) و مجموعه ای از عملگرهـا تشـکیل میشـود. عملگـر هـای درج[1](Insert)، حذف[2](Delete)، حذف رکورد دارای کوچکترین کلید[3](XMin)، حذف رکورد دارای بزرگترین کلید[4](XMax)، جستجو(Search)، نزدیکترین کلید(Near)، رکورد بعد (Next)، رکورد قبل (Prev) ، رکورد دارای کوچکترین کلید (Min)، رکورد

1- Insert

2- Delete
3- Extract minimum
4- Extract maximum

دارای 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 $F(k) = \{(k, r) | (k, r) \in F\}$. بنابراین در صورت وجود کلید k در دیکشنری، F(k) یک مجموعه تک عضوی و در غیر اینصورت یک مجموعه تهی میباشد. اثر اعمال 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 (p, k) :
$F \leftarrow F - F(k_{min})$
return $F(k_{min})$

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

Search (p, k) :
if $k \in F$ **then return** F(k)
else return 'not found'

Next (p, k) :
if Succ (k) $\in$ F **then return** F(Succ (k))
else return 'Last Key'

Prev (p, k) :
if Pred (k) $\in$ F **then return** F(Pred (k))
else return 'First Key'

Near (p, k) :
return $F(X_{near})$ where k_{near} is the stored key closest to k

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

Count (p) :
return number of keys k where $k \in F$

اگر کلید k قبل از عمل درج در دیکشنری موجود باشد، عمل درج را زائد[1] 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[2] و عملگرهای پرس وجو[3] 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 است 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 پروسس های موازی در دسترسی به ساختمان داده مشترک میباشد در حالیکه در طراحی الگوریتم های موازی برای پیاده سازی روی کامپیوترهای موازی با حافظه توزیع شده، توجه به استخراج توازی در ساختمان داده است. عملگرهای موازی برای درخت های دودوئی [۱۷]، درخت های متعادل [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 متعادل[4] [۱۰]، زنجیره مرتب[5] [۲۷] وهیپ بانین[6] [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[7]: حداقل فاصله زمانی که بین شروع اجرای دو عملگر متفاوت نیاز میباشد را فاصله شروع دو عملگر مینامند.

زمان پاسخ[8]: زمان بین شروع و اتمام یک عملگر را زمان پاسخ عملگر مینامند.

توان عملیاتی[9]: 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 مش پیاده سازی شده اند شرح داده میشود. در پایان این مقاله کارایی مش جستجوی دودوئی کامل که در این مقاله پیشنهاد شده است با این سه ماشین مقایسه خواهد شد.

ماشین Schmeck و Schroder ساختار این ماشین بصورت یک مش $n \times n$ است که در آن هر پردازنده به چهار پردازنده همسایه اش متصل شده است. کمانهای این ماشین به دو دسته تقسیم میشوند. یکدسته از

1- Redundant
2- Modify operators

3- Query operations
4- Balance cube
5- Sorted chain
6- Banyan heap
7- Initiation interval
8- Response time
9- Throughput

کمانها برای انتشار[1] 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 مش $n \times n$ 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[2] و شبکه انتشار تشکیل شده است که هر دو در مش جاداده میشوند و بطور همزمان عمل میکنند. شبکه مارپیچی بصورتی در مش جا داده میشود که هر پردازنده فقط یک بار در آن ظاهر میشود. رکوردها بصورت نزولی (از طرف پردازنده ورودی-خروجی[3]) 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 مش 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 مش $n \times n$ است که در آن هر پردازنده به شش پردازنده همسایه اش متصل میباشد. از نظر منطقی این ماشین از دو شبکه خطی[4] و شبکه پوشا[5] تشکیل شده است که هر دو در مش 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].

مش جستجوی دودوئی[6]، یک نمونه از ساختمان داده های موازی روی کامپیوترهای موازی با حافظه توزیع شده و ساختار مش یا ابر مکعب میباشد که برای اولین بار در [۲۴] پیشنهاد شده است. پیاده سازی این ساختمان داده روی مش 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 مش جستجوی دودوئی پس از عمل حذف ، عدم تعادل مش و پایین بودن توان عملیاتی آن اشاره نمود. عدم تعادل مش سبب افزایش زمان پاسخ میگردد. برای کاهش زمان پاسخ ، ساختمان داده مش جستجوی دودوئی متعادل[7] پیشنهاد شده است [۴] که در آن تعداد رکورد های ذخیره شده در تمام پردازنده های نیم مش تقریبا برابر میباشد. در مش جستجوی دودوئی $n \times n$ و مش جستجوی دودوئی متعادل $n \times n$ 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 مش جستجوی دودوئی کامل[8] معرفی میگردد. این ساختمان داده که مشابه مش جستجوی دودوئی است دارای توان عملیاتی بالاتری میباشد. در هنگام پیاده سازی این ساختمان داده روی مش 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 مش جستجوی دودوئی کامل ایجاد میکند که باعث میگردد که عملگرهای ماشین دیکشنری بصورت متوالی وارد پردازنده ها گردند. این عامل باعث میشود تا سازگاری در مش جستجوی دودوئی کامل بسادگی قابل پیاده سازی باشد. در مش 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 است. ساختمان داده مش 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 مش جستجوی دودوئی کامل و چند ماشین دیکشنری گزارش شده [۱۱][۱۲][۳۱][۳۴] برای کامپیوترهای موازی با ساختار مش در نتیجه گیری آمده است.

۲- مش جستجوس دودوئی کامل

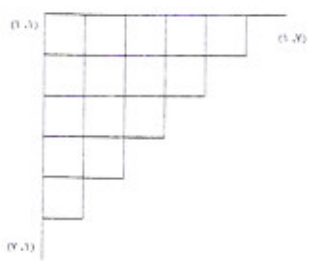
در این قسمت در ابتدا تعاریف مش d- بعدی و نیم مش دو بعدی داده شده است و سپس مش جستجوی دودوئی و مش جستجوی دودوئی متعادل تعریف و اشکالات موجود در آنها بررسی گردیده است و نهایتا تعریف مش جستجوی دودوئی کامل که پیاده سازی آن در این مقاله به تفضیل آمده است داده شده است.

مش d- بعدی: یک مش 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 مش 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 گفته میشود اگر یک i $(1 \le i \le d)$ وجود داشته باشد بطوریکه $|x_i - y_i| = 1$ و $x_j = y_j$ برای تمام $j \neq i$

نیم مش دو بعدی: یک نیم مش دو بعدی $n \times n$ (که به اختصار نیم مش $n \times n$ نامیده میشود) از تقسیم مش دو بعدی $n \times n$ در راستای قطر $\{(1,n),(2,n-1),... (n-1,2),(n,1)\}$ بدست میاید. شکل ۱ یک نیم مش دو بعدی ۷×۷ را نشان میدهد. در این نیم مش، گره (۱،۱) ریشه، گره های روی قطر $N_L = \{(i, j) | i + j = n + 1\}$ برگ ، گره های روی اضلاع $N_{TB} = \{(1, k) | 1 \le k < n\}$ و $N_{LB} = \{(k, 1) | 1 < k < n\}$ گرههای مرزی (بترتیب مرز بالا و مرز چپ) و بقیه گره ها ، گره های میانی نامیده میشوند. نیم مش $n \times n$ دارای n سطح میباشد که بترتیب از سطح ۱ الی n شماره گذاری میشوند و ریشه نیم مش در سطح ۱ قرار دارد. در سطح k ام نیم مش k گره $L_K = \{(i, j) | i + j = K + 1\}$ وجود دارند که برادر یکدیگر میباشند. گره (k, 1) را برادر کوچکتر (گره مرزچپ) وگره (1, k) را برادر بزرگتر (گره مرز بالا) مینامند. برگ های نیم مش در سطح n ام نیم مش 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)

1- Broadcast net
2- Snak net
3- IO processor
4- Linear net
5- Spanning net
6- Binary search mesh (BSM)
7- Balanced binary search mesh (BBSM)
8- Complete binary search mesh (CBSM)

و گره ریشه بدون پدر میباشند. گره غیر برگ (I, J) دارای دو فرزند سمت راست (I, J+1) و فرزند سمت چپ (I+1, J) میباشد.
شماره سطح یک گره ، عمق آن گره و بیشترین عمق در بین گره های غیر تهی، عمق مش نامیده میشود.



شکل ۱: نیم مش دو بعدی ۷×۷

مش جستجوی دودوئی: مش جستجوی دودوئی یک نیم مش 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 مش 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 مش جستجوی دودوئی میگردد. اشکال دیگرعدم برقراری خواص مش 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 مش جستجوی دودوئی از تشکیل حفره جلوگیری میشود. روش دوم از بین رفتن حفره با استفاده از مش جستجوی دودوئی متعادل میباشد. تعادل در مش جستجوی دودوئی سبب عدم تشکیل حفره میگردد.
عدم برقراری خواص مش جستجوی دودوئی و تشکیل حفره در هنگام حذف رکورد x بصورت زیر برطرف شده است. برای حذف وکورد 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 باشدرکورد 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 صورتی که 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 میانی بدون فرزند برسیم.
در مش 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 ونتیجتا توان عملیاتی ساختمان داده کاهش میابد. مشکل دیگری که در مش جستجوی دودوئی وجود دارد رشد یک بعدی مش (ما نند آنچه در درخت جستجوی دودوئی وجود دارد) میباشد. در اینصورت اگر داده های درج شونده بترتیب صعودی (نزولی) وارد مش شوند تنها در گره های مرزبالا (مرز چپ) مش قرار میگیرند و بقیه گره ها خالی میما نند. رشد مش در یک بعدو عدم تعادل تعداد رکورد ها در گره های مش سبب کاهش توان عملیاتی این ساختمان داده میگردد. برای رفع این اشکالات، مش جستجوی دودوئی متعادل که تعریف آن در زیر آمده است پیشنهاد شده است[۴]. در این ساختمان داده رکورد ها بین گره های مش بصورت متعادل توزیع میشوند که این خود باعث افزایش توان عملیاتی ساختمان داده میگردد.
مش جستجوی دودوئی متعادل: مش جستجوی دودوئی متعادل یک نیم مش دوبعدی است که شرایط زیر را دارا میباشد.
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 مش 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 مش جستجوی دودوئی و مش جستجوی دودوئی متعادل $n \times n$ 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 مش جستجوی دودوئی کامل حل مشکل پایین بودن توان عملیاتی در مش جستجوی دودوئی و مش جستجوی دودوئی متعادل میباشد.
مش جستجوی دودوئی کامل $n \times n$: مش جستجوی دودوئی کامل $n \times n$ (که به اختصار مش جستجوی دودوئی کامل نامیده میشود) یک نیم مش 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 مش تنها یک سطح وجود دارد که دارای گره های نیمه پر است.

۶- هر گره نیمه پر دارای والدین پر میباشد.

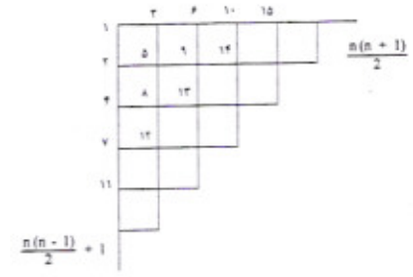
۷- یک سطح بطور همزمان نمیتواند دارای گره های خالی و پر باشد.

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 دراینصورت 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 مش دوبعدی $n \times n$ با عمق m که در آن N رکورد ذخیره شده باشد را بصورت زیر شماره گذاری نماییم در اینصورت اگر نیم مش ، یک مش جستجوی کامل باشد قضایای زیر صادق هستند.



شکل ۲: شماره گذاری نیم مش دو بعدی

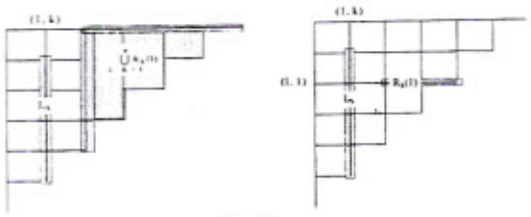
قضیه ۱: اگر $n(n-1) < N \le m^2$ باشد گرههای با شماره $1, \ldots, \frac{m(m-1)}{2}$ پر و گره های با شماره $\frac{m(m-1)}{2}+1, \ldots, N-\frac{m(m-1)}{2}$ نیمه پر هستند.

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

قضیه ۳: اگر شرط $N > n(n-1)$ برقرار باشد گره های برگ با شماره $1, \ldots, \frac{n(n-1)}{2}$ پر هستند و برگهای با شماره $\frac{n(n-1)}{2}+1, \ldots, \frac{n(n-1)}{2}+K$ دارای $\left\lceil \frac{N-n(n-1)}{n} \right\rceil$ رکورد و برگهای با شماره $\frac{n(n+1)}{2}, \ldots, \frac{n(n-1)}{2}+K+1, \ldots$ دارای $\left\lfloor \frac{N-n(n-1)}{n} \right\rfloor$ رکورد هستند. (K=mod (N- n (n - 1). n))

اثبات قضایای فوق در [۵] آورده شده است.

فرض کنید که مجموعه های $R_k(l)=\{(l,J) \mid k< J \le n\}$، $(1 \le l \le n-k)$، $L_k=\{(l,k) \mid k< l \le n\}$ و $\bigcup_{l=k+1} R_k(l)$ بصورت شکل زیر تعریف شده باشند.



شکل ۳

گره مرز بالا $(1, k)$ را کامل میگوییم

۱- اگر گره $(1, k)$ خالی یا نیمه پر باشد یا

۲- اگر تفاوت رکوردهای موجود در مجموعه $R_k(1)=\{(1, J) \mid k< J \le n\}$ و مجموعه $L_k=\{(l, k) \mid k< l \le n\}$ برابر با صفر یا ۱- باشد و گره مرزبالا $(1, k+1)$ کامل باشد.

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

۱- اگر ریشه کامل باشد و

۲- اگر نیم مش دو بعدی $(n-1) \times (n-1)$ با ریشه فرزند سمت راست q کامل باشد.

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

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

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

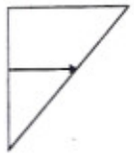
اثبات قضایای فوق در [۵] آورده شده است.

کوچکترین و بزرگترین رکورد هر گره بترتیب توسط دو متغیر Smallltem و Largeltem ذخیره میشوند. در صورتیکه گره ای دارای یک رکورد باشد مقدار Smallltem و Largeltem یکسان میباشند.

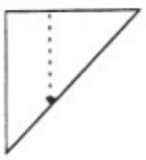
با توجه به تعریف مش جستجوی دودوئی کامل میتوان نتایج زیر را گرفت. برای سهولت نمایش، رکورد های Smallltem (S) ، Largeltem (L) و زوج (Smallltem, Largeltem)(l) بترتیب توسط خطوط نقطه چین، خط چین و پر نشان داده میشوند.

نتیجه ۱: رشته زیر یک رشته صعودی است.

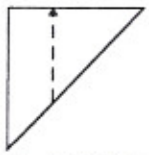
{S (l, 1), L (l, 1), S (l, 2), L (l, 2), . . . S (l, n- l +1), L (l, n-l+1) }



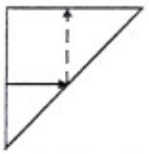
نتیجه ۲: رشته {S (1, J), S (2, J),. . . , S (n- J +1, J)} یک رشته صعودی است.



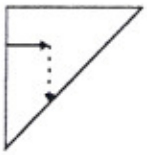
نتیجه ۳: رشته { L (n- J +1, J), . . , L (2, J), L (1, J)} یک رشته صعودی است.



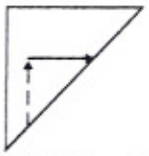
نتیجه ۴: رشته {I (I, 1),. . ,I(I, J-1), S (I, J), L(I, J) . . , L (1, J)} صعودی است.



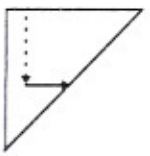
نتیجه ۵: رشته {I (I, 1),. . ,I(I, J-1), S (I, J), . . , S (J, n-J+1)} صعودی است.



نتیجه ۶: رشته {L (n-J+1, J),. . , L (I, J), I (I, J), . . , I (I, n-I+1)} صعودی است.

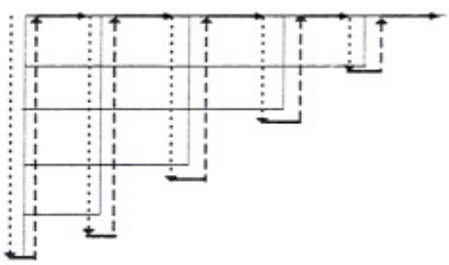


نتیجه ۷: رشته {S (1, J),. . , S (I, J), I (I, J), . . , I (I, n-I+1)} صعودی است.



قضیه ۶: زنجیره زیر یک رشته 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 -1), L(2, n -1), L(1, n -1), S(1, n), L (1,n)}



اثبات: با استفاده از نتایج ۱، ۲و ۳.

۳- عملگرها

در این بخش چگونگی کارکرد عملگر هـای درج، حذف، حـذف رکـورد دارای کوچکترین کلیـد، جـستجو، رکـورد دارای کلیدبعـد ، رکـورد دارای کوچکترین کلید و رکورد دارای بزرگترین کلید شرح داده میشـود. بـرای سادگی ارائه ، در ادامه بحث رکورد های موجـود در یـک گـره غـیر بـرگ توسط زوج (SmallItem, LargeItem) نشان داده میشود.

تعادل در گره مرز بالا (1, k) توسط متغیر Balance نشـان داده میشـود. اگر مقدار این متغیر مساوی صفر باشد در اینصورت این گره متعادل است و نیم مشی که ریشه آن گره مرز بالا (1, k) اسـت یـک مـش جـستجوی دودوئی کاملا پر میباشد. در صورتی که مقدار این متغـیر بزرگـتر از صفـر باشد تعداد رکوردهای موجود در مجموعه $R_k(1)$ بیشـتر ازتعـداد رکـورد های موجود در مجموعه L_k و در غیر اینصورت تعداد رکوردهـا ی موجـود در مجموعه L_k بیشـتر از تعـداد رکـورد هـای موجـود در مجموعـه $R_k(1)$ میباشد. بدلیل اینکه رکوردهایی کـه از طریـق فرزنـد سـمت راسـت گـره مرزبـالا (1, k) درج میشـوند مشـخص نیسـتند کـه در چـه گـره ای درج میشوند بنابراین مقدار متغیر Balance قابل محاسبه نیست. قضیه زیر بـر اساس تعداد رکوردهایی که از طریق فرزند سـمت راسـت و فرزنـد سـمت چپ درج میشوند چگونگی محاسبه مقدار متغیر Balance را بیان میکند.

قضیه ۷: فرض کنیـد R و L بـترتیب نشـان دهنـده تعـداد رکـورد هـای 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.

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

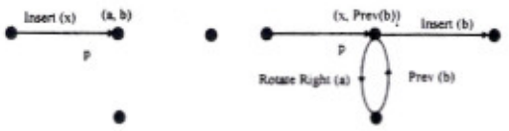
ثبات قضیه فوق در [۵] آورده شده است.

عملگر درج : این عملگر علاوه بر درج یک رکورد وظیفه حفظ خـاصیت مش جستجوی دودوئی کامل را بر عهده دارد. ایـن عملگـر در حیـن درج یک رکورد در مش با ایجاد چرخش های لازم تعـادل در مـش جـستجوی دودوئی را برقرار مینماید. عملکرد این عملگر برای گره های مرز بالا و گره های دیگر متفاوت است. نحوه اجرای ایـن عملگـر بـا توجـه بـه نـوع گـره بصورت زیر میباشد.

الف- اگرگره q یک گره مرزبالا باشد، عملگر درج رکورد x را از پدر سـمت چپ خود دریافت میکند. در صورتیکه گره q خالی یا نمیه پر باشد رکورد x در این گره قـرار میگـیرد. در صورتیکـه گـره q دارای دو رکـورد باشـد متغیر Balance مشخص میکند که رکورد x ، باید از طریـق کـدام فرزنـد درج شود. اگر مقدارمتغیر Balance کوچکـتر از صفـر باشـد رکـورد x از طریق فرزند سمت راست درج میشود. در فرایند درج رکـورد x، از طریـق فرزند سمت راست امکان برهم خوردن خاصیت مش جـستجوی دودوئـی کامل وجود دارد. برای برقرار نمودن خـاصیت مـش جـستجوی دودوئـی کامل یکی از روشهای زیر میتواند استفاده شود.

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

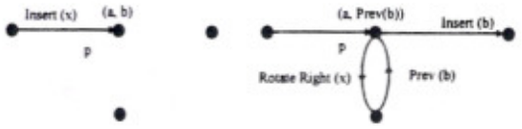
۲- اگـر رکـورد x ، از رکـورد SmallItem کوچکـتر باشـد، رکـورد LargeItem ، از طریـق فرزنـد سـمت راسـت درج میشـود ورکـورد SmallItem برای چرخش از طریق فرزنـد سـمت چپ اسـتفاده میگـردد و سپس رکورد های x و Prev (LargeItem) (رکورد LargeItem در فرزند سـمت چپ ایـن گـره) بـترتیب جـای خـالی رکـورد هـای SmallItem و LargeItem را پر میکنند. نحوه اجرای این عمل در شـکل زیـر نشـان داده شده است.



قبل از اجرا عملگر درج پس از اجرا عملگر درج

شکل۴

۳-اگـر رکـورد x از رکـورد SmallItem بزرگـتر و ازرکـورد LargeItem کوچکتر باشد، رکوردهای x و LargeItem بترتیب از طریق فرزندان سـمت چپ و سمت راست راست درج میشوند و سپس رکورد Prev (LargeItem) جای خالی LargeItem را پر میکند. نحوه اجرای این عمـل در شـکل زیـر نشان داده شده است.



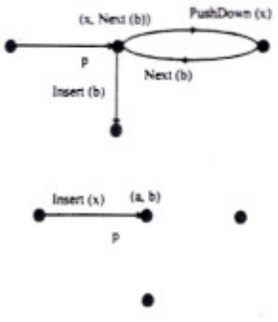
قبل از اجرا عملگر درج پس از اجرا عملگر درج

شکل۵

اگر مقدار Balance بزرگتر یا مساوی صفر باشد، رکورد x از طریـق فرزنـد سمت چپ درج میشود. در اینصورت یکی از سه حالات زیر اتفاق میافتد.

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

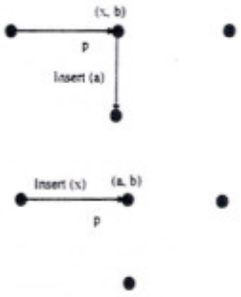
۲- اگر رکورد x از رکورد LargeItem بزرگتر باشـد رکـورد LargeItem از طریق فرزند سمت چپ درج میشود ورکـورد x بـرای چرخـش از طریـق فرزند سمت راست استفاده میگردد و سپس جای خالی رکورد LargeItem توسط رکورد (LargeItem) Next (رکـورد SmallItem در فرزنـد سـمت راست این گره) پر میگردد. نحوه اجرای این عمل در شکل زیر نشان داده شده است.



پس از اجرا عملگر درج قبل از اجرا عملگر درج

شکل ۶

۳- اگر رکورد x از رکورد SmallItem کوچکتر باشـد رکـورد SmallItem از طریق فرزند 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.



پس از اجرا عملگر درج قبل از اجرا عملگر درج

شکل ۷

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

۱- اگر رکورد x از رکورد LargeItem بزرگـتر باشـد رکـورد LargeItem از طریق فرزند سمت چپ درج میشود وسپس جای خالی آن توسط رکـورد x پر میگردد.

۲- اگر رکورد x از رکورد SmallItem کوچکتر باشد رکـورد SmallItem از طریق فرزند سمت چپ درج میشود و سپس رکورد x ، جای خالی آنرا پـر میکند.

۳-اگـر رکـورد x ازرکـورد SmallItem بزرگـتر و ازرکـورد LargeItem کوچکتر باشد رکورد x از طریق فرزند سمت چپ درج میشود.

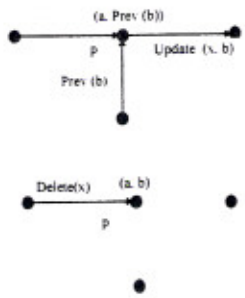
عملگر حذف : نحوه اجرای این عملگر مشـابه عملگـر حـذف در درخـت جسـتجوی دودوئـی اسـت یعنـی بـا حـذف رکـورد x، رکـورد Next (x) (کوچکترین رکورد بزرگـتر از x) یـا رکـورد Prev (x) (بزرگـترین رکـورد کوچکتر از x) جای خالی x را پر میکنند. دراین عملگـر، در حیـن حـذف یک رکورد با ایجاد چرخش های لازم خـواص مـش جسـتجوی دودوئـی کامل برقرار میگردد. عملکرد این عملگر برای گره های مرز بالا و گره های دیگر متفاوت است. نحوه اجرای این عملگر با توجه به نـوع گـره بصـورت زیر میباشد.

الف - اگر گره q یک گره مرز بالا باشد، عملگر حذف رکـورد x را ، از پـدر سمت چپ خود دریافت میکند. اگر گره q فرزندی نداشته باشـد رکـورد x از این گره حذف میگردد. در غیر اینصورت یکی از چهار حالت زیـر پیـش آید.

۱- اگر رکـورد x از رکـورد LargeItem و متغـیر Balance از صفـر بزرگـتر باشند رکورد x از طریق فرزند سمت راست حذف میشود.

۲- اگر رکورد x از رکورد LargeItem بزرگتر و متغیر Balance کوچکتر یـا مسـاوی صفـر باشـند، از طریـق فرزنـد سـمت راسـت رکـورد هـای x و

Largeltem بترتیب حذف ودرج میشوند و سپس رکورد Prev (Largeltem) جای خالی Largeltem را پر میکند. نحوه اجرای این عمل در شکل زیر نشان داده شده است.

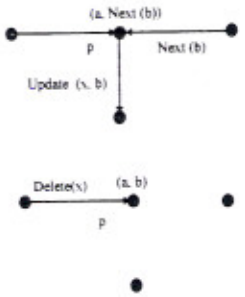


پس از اجرا عملگر درج قبل از اجرا عملگر درج

شکل ۸

۳- اگررکورد x ازرکورد Smalllteم بزرگتر و ازرکورد Largeltem کوچکتر و متغیر Balance کوچکتر یا مساوی صفر باشد، رکورد x از طریق فرزند سمت چپ حذف میشود.

۴- اگررکورد x ازرکورد Smalllteم بزرگتر و ازرکورد Largeltem کوچکتر و متغیر Balance بزرگتراز صفر باشد، رکورد های x و Largeltem بترتیب از طریق فرزند سمت چپ حذف و درج میگردند و سپس جای خالی Largeltem توسط رکورد Next (Largeltem) پر میشود. نحوه اجرای این عمل شکل زیر نشان داده شده است.



پس از اجرا عملگر درج قبل از اجرا عملگر درج

شکل ۹

۵- اگر رکورد x مساوی رکورد Smalllteم باشد، رکورد Smalllteم حذف و سپس جای خالی آن توسط رکورد Next (Smalllteم) پر میشود. اگر متغیر Balance بزرگتر از صفر باشد رکورد Largeltem برای چرخش ازطریق فرزند سمت چپ استفاده میشود وسپس جای خالی رکورد Largeltem توسط رکورد Next (Largeltem) 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 Largeltem باشد، رکورد Largeltem حذف میشود. اگر متغیر Balance بزرگتر از صفر باشد جای خالی Largeltem توسط رکورد Next (Largeltem) و اگر متغیر Balance مساوی یا کوچکتراز صفر باشد جای خالی Largeltem توسط رکورد Prev (Largeltem) پر میشود.

ب- اگر گره q یک گره میانی یا مرز چپ یا برگ باشد، عملگر حذف رکورد x را از پدر سمت راست خود دریافت میکند. اگر گره q یک گره برگ یا یک گره میانی نیمه پر 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 Smalllteم بزرگتر و ازرکورد Largeltem کوچکتر باشد، رکورد x ازطریق فرزند سمت چپ حذف میشود.

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

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

عملگر جستجو : اگر رکورد یا کلید x ، در مش 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 جستجورا 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 گره q باشد، کلید x در مش وجود دارد و جستجو پایان میپذیرد.

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

۳- اگر کلید x بزرگتر از کلید رکورد Smalllteم و کوچکتر از کلید رکورد Largeltem باشد، جستجو از طریق فرزند سمت چپ انجام میشود.

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

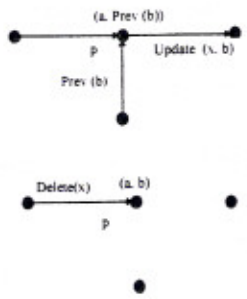
ب- اگر گره q یک گره میانی یا مرز چپ یا برگ باشد، عملگر جستجو را از پدر سمت راست خود دریافت میکند و یکی از سه حالت زیر پیش میاید.

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

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

۳- اگر کلید x کوچکتراز کلید رکورد Smalllteم یا بزرگتر از کلید رکورد Largeltem یا گره q خالی باشد، کلید x در مش موجود نمیباشد و جستجو پایان میپذیرد.

Largeltem بترتیب حذف ودرج میشوند و سپس رکورد Prev (Largeltem) جای خالی Largeltem را پر میکند. نحوه اجرای این عمل در شکل زیر نشان داده شده است.

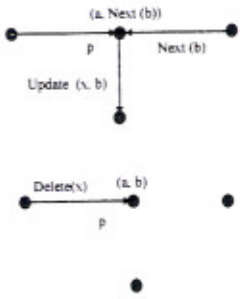


پس از اجرا عملگر درج قبل از اجرا عملگر درج

شکل ۸

۳- اگررکورد x ازرکورد Smalllteم بزرگتر و ازرکورد Largeltem کوچکتر و متغیر Balance کوچکتر یا مساوی صفر باشد، رکورد x از طریق فرزند سمت چپ حذف میشود.

۴- اگررکورد x ازرکورد Smalllteم بزرگتر و ازرکورد Largeltem کوچکتر و متغیر Balance بزرگتراز صفر باشد، رکورد های x و Largeltem بترتیب از طریق فرزند سمت چپ حذف و درج میگردند و سپس جای خالی Largeltem توسط رکورد Next (Largeltem) پر میشود. نحوه اجرای این عمل شکل زیر نشان داده شده است.



پس از اجرا عملگر درج قبل از اجرا عملگر درج

شکل ۹

۵- اگر رکورد x مساوی رکورد Smalllteم باشد، رکورد Smalllteم حذف و سپس جای خالی آن توسط رکورد Next (Smalllteم) پر میشود. اگر متغیر Balance بزرگتر از صفر باشد رکورد Largeltem برای چرخش ازطریق فرزند سمت چپ استفاده میشود وسپس جای خالی رکورد Largeltem توسط رکورد Next (Largeltem) 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 Largeltem باشد، رکورد Largeltem حذف میشود. اگر متغیر Balance بزرگتر از صفر باشد جای خالی Largeltem توسط رکورد Next (Largeltem) و اگر متغیر Balance مساوی یا کوچکتراز صفر باشد جای خالی Largeltem توسط رکورد Prev (Largeltem) پر میشود.

ب- اگر گره q یک گره میانی یا مرز چپ یا برگ باشد، عملگر حذف رکورد x را از پدر سمت راست خود دریافت میکند. اگر گره q یک گره برگ یا یک گره میانی نیمه پر 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 Smalllteم بزرگتر و ازرکورد Largeltem کوچکتر باشد، رکورد x ازطریق فرزند سمت چپ حذف میشود.

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

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

عملگر جستجو : اگر رکورد یا کلید x ، در مش 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 جستجورا از پدر سمت چپ خود دریافت میکند. و یکی از چهار حالت زیر پیش میاید.

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

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

۳- اگر کلید x بزرگتر از کلید رکورد Smalllteم و کوچکتر از کلید رکورد Largeltem باشد، جستجو از طریق فرزند سمت چپ انجام میشود.

۴- اگر کلید x کوچکتراز کلید رکورد Smalllteم یا گره q خالی باشد، کلید x در مش 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 گره q باشد، کلید x در مش وجود دارد و جستجو پایان میپذیرد.

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

۳- اگر کلید x کوچکتراز کلید رکورد Smalllteم یا بزرگتر از کلید رکورد Largeltem یا گره q خالی باشد، کلید x در مش موجود نمیباشد و جستجو پایان میپذیرد.

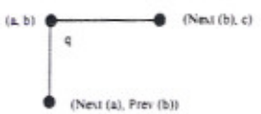
عملگر رکورد دارای کوچکترین کلید: وظیفه این عملگر پیدا نمودن رکورد دارای کوچکترین کلید در مش میباشد. رکورد دارای کوچکترین کلید در متغیر SmallItem گره ریشه قرار دارد.

عملگر رکورد دارای بزرگترین کلید: وظیفه این عملگر پیدا نمودن رکورد دارای بزرگترین کلید در مش 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 مش 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.

عملگر رکورد بعد : وظیفه این عملگر در صورتیکه کلید x بزرگترین کلید مش 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 در این شرایط یکی از چهار حالت زیر پیش میآید.

۱- اگر گره q یک گره نیمه پر باشد، این رکورد دارای بزرگترین کلید میباشد.

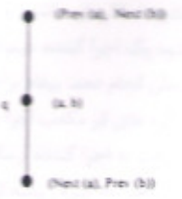
۲- اگر گره q یک گره پر بدون فرزند و کلید x مساوی کلید رکورد SmallItem , باشد، رکورد بعد، رکورد LargeItem گره q است.

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- اگر گره q یک گره پر دارای فرزند سمت راست و کلید x مساوی کلید رکورد LargeItem , باشد، رکورد بعد، رکورد SmallItem فرزند سمت راست این گره است.

۵- اگر گره q یک گره پر دارای فرزند سمت چپ و کلید x مساوی کلید رکورد SmallItem , باشد، رکورد بعد، رکورد SmallItem فرزند سمت چپ 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 بصورت زیر میباشد.



شکل ۱۲

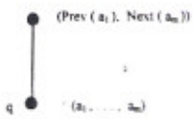
برای اجرای این عملگردر این شرایط یکی از سه حالت زیر پیش میآید.

۱- اگر گره q یک گره نیمه پر باشد، رکورد بعد، رکورد LargeItem پدر سمت راست این گره است.

۲- اگرکلید x مساوی کلید رکورد , LargeItem باشد، رکورد بعد، رکورد LargeItem پدر سمت راست این گره است.

۳- اگر گره q یک گره دارای فرزند سمت چپ و کلید x مساوی کلید رکورد , SmallItem باشد، رکورد بعد، رکورد SmallItem فرزند سمت چپ این گره است.

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

در شکل فوق m و a_k $(k = 1, \ldots, m)$ بترتیب تعداد و کلید رکورد های ذخیره شده (با ترتیب $a_k < a_{k+1}$) در برگ q میباشند برای اجرای این عملگر در این شرایط 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 مساوی کلید رکورد , a_k $(k < m)$ باشد، رکورد بعد، رکورد a_{k+1} گره q است.

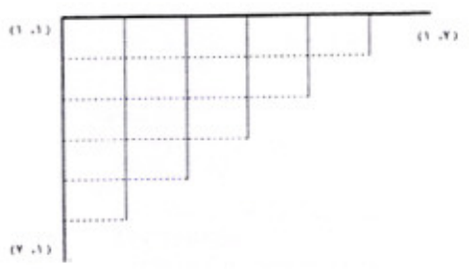
۳- پیاده سازی

در این قسمت چگونگی پیاده سازی مش جستجوی دودوئی کامل روی یک کامپیوتر موازی با حافظه توزیع شده و ساختار ابر مکعب یا مش شرح داده میشود.

نیم مش 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 مش دو بعدی به گره S_1S_2 در ابر مکعب نگاشت داده میشود بطوریکه x_i ، S_i امین کد گری d_i بیتی میباشد [۸][۹][۳۰].

گره هایی از ابر مکعب که عضو نیم مش نگاشت شده نیستند تشکیل یک نیم مش (n-1) × (n-1) را میدهند که نیم مش آینه ای نامیده

میشود. نیم مش آینه ای را میتوان برای ذخیره نمودن ساختمان داده دیگری و یا ترکیب نتایج تولید شده توسط برگ ها استفاده نمود. شکل زیر یک مش جستجوی دودوئی کامل ۷ × ۷ را نشان میدهد.



شکل ۱۴ : مش جستجوی دودوئی کامل ۷×۷

مطابق شکل فوق کمانهای نیم مش 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 مش 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 مش جستجوی دودوئی کامل

SmallItem کوچکترین رکورد در هر گره و LargeItem بزرگترین رکورد در هر گره است. درصورتیکه پردازنده برگ باشد بدلیل اینکه بیشتر 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.

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

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[1] 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 برای اجرا به پردازنده <processor> ارسال میکند. پردازنده ای 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 دراینصورت دستورالعمل RECEIVE زمانی کامل میگردد که اطلاعات از یک پردازنده دریافت شود. برنامه کاربردی که روی پردازنده p اجرا میشود عملگر دیکشنری را توسط دستور SEND (root, <instruction>) به ریشه ارسال میکند که <instruction> یکی از عملگرهای دیکشنری است. سپس اگر دستورالعمل صادر شده 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 anyprocessor شماره یک پردازنده در ابر مکعب یا مش 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 ازدستورMin (p)، برای پیدا کردن رکورد دارای بزرگترین کلید از دستور Max (p)، برای حذف رکورد داری کوچکترین کلید از دستور XMin (p) و برای رکورد بعد از دستور Next (p, x) استفاده میشود. p پردازنده ای است که عملگر را شروع میکند. حال به چگونگی پیاده سازی عملگرها ماشین دیکشنری میپردازیم.

عملگر درج : عملکرد این عملگر برای پردازنده های مرز بالاو 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 Insert (x) را از پدر سمت چپ خود دریافت میکند. در اجرای این عملگر، اگررکورد x ، از طریق

1- Executive

فرزند سمت راست درج شود مقدار متغیر RightCount و در صورتیکه از طریق فرزند سمت چپ درج شود مقدار متغیر 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
      dec (LeftCount (q))
    end if
  end if
end if
```

ب- اگر پردازنده q یک پردازنده میانی، مرز چپ یا برگ باشد عملگر 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 بالا و پردازنده های دیگر متفاوت است. نحوه اجرای این عملگر با توجه به نوع پردازنده بصورت زیر میباشد.

الف- اگر پردازنده q یک پردازنده مرز بالا باشد عملگر Delete (x) را از پدر سمت چپ خود دریافت میکند. در اجرای این عملگر، اگرکورد x ازطریق فرزند سمت راست حذف شود مقدار متغیر RightCount ودر صورتیکه از طریق فرزند سمت چپ حذف شود مقدار متغیر 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- اگر پردازنده q یک پردازنده میانی ، مرز چپ یا برگ باشد عملگر Delete (x) را از پدر سمت 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 پردازندههای دیگر متفاوت است. نحوه اجرای این عملگر با توجه به نوع پردازنده بصورت زیر میباشد.

الف- اگر q پردازنده مرز بالا باشد عملگر Search (p, x) را از پدر سمت چپ خود دریافت 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 راست 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 رکورد دارای کوچکترین کلید: بدلیل اینکه رکورد دارای کوچکترین کلید در پردازنده ریشه 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
```

عملگر 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
```

عملگر 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 عملگر Next (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 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
```

عملگر حذف رکورد دارای کوچکترین کلید: بدلیل اینکه رکورد دارای کوچکترین کلید در پردازنده ریشه قرار دارد عملگر XMin (p) تنها در پردازنده ریشه 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 عملگر Delete (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 PopupSmall را 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 از صفر باشد، برای حفظ تعادل در مش ، رکورد LargeItem برای چرخش به چپ به فرزند سمت چپ 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 یا مرز چپ یا برگ با شد عملگر PopupSmall 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 مربوط به 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 یا مرز چپ 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 و برگ اجرا میشود باعث میشود تا رکورد LargeItem از این گره حذف و برای جایگزینی در رکورد LargeItem مربوط به پدر سمت راست ارسال شود. اگر پردازنده q، یک پردازنده میانی 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 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) : عملگر PushDown (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 باشد 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 و رکورد x بزرگتر از رکورد LargeItem باشد رکورد SmallItem حذف و به پدر سمت چپ، 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- اگر پردازنده q دارای فرزند سمت چپ و رکورد x کوچکتر از رکورد LargeItem و بزرگتر از رکورد SmallItem باشد رکورد SmallItem حذف و به پدر سمت چپ، رکورد x برای چرخش به چپ به فرزند سمت چپ ارسال میگردند و سپس جای خالی رکورد SmallItemتوسط 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 شود 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 یک پردازنده مرز بالا باشد عملگر Update (x, y) را از پدر سمت چپ 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 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 برگتر از رکورد LargeItem و کوچکتر از رکورد SmallItem 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 یا برگ بـا شد عملگر Update (x, y) را از پدر سمت راست خود دریافت میکند. با توجه بـه نـوع 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 برگتر از رکورد LargeItem باشد رکورد SmallItem برای چرخش به راست به فرزند سمت چپ و رکوردهای x و LargeItem 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 برگتر 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، مـرز چپ و برگ q اجرا میشود و سبب میشود تا رکورد x از طریق پردازنـده درج

شود و سپس رکورد Largeltem پردازنده q حذف و به پدر سمت راست ارسال گردد. اگر پردازنده q، 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 Largeltem بزرگتر باشد رکورد x به پدر سمت راست ارسال میشود.

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

۳- اگر رکورد x کوچکتر از رکورد Largeltem و بزرگتر از رکورد Smallltem باشد رکورد Largeltem حذف و به پدر سمت راست و رکورد x برای چرخش به چپ به فرزند سمت چپ ارسال میگردند و سپس جای خالی رکورد Largeltem توسط رکورد Prev (Largeltem) 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 > Largeltem (q) then
      SEND (RP (q), SetLarge (x))
    else
      SEND (RP (q), SetLarge (Largeltem (q)))
      if x < Smallltem (q) then
        SEND (LC (q), RotateRight (Smallltem (q)))
        Smallltem (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 و برگ q اجرا میشود و سبب میشود تا رکورد x از طریق پردازنده درج شود و سپس رکورد Smallltem پردازنده q حذف و به پدر سمت راست ارسال گردد. اگر پردازنده q، 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 ، کوچکتر از رکورد Smallltem باشد رکورد x به پدر سمت راست ارسال میشود.

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

۳- اگر رکورد x کوچکتر از رکورد Largeltem و بزرگتر از رکورد Smallltem باشد رکورد Smallltem حذف و به پدر سمت راست و رکورد x برای چرخش به راست به فرزند سمت چپ ارسال میگردند و سپس جای خالی رکورد Smallltem توسط رکورد Next (Smallltem) 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 < Smallltem (q) then
      SEND (RP (q), SetSmall (x))
    else
      SEND (RP (q), SetSmall (Smallltem (q)))
      if x > Largeltem (q) then
        SEND (LC (q), RotateLeft (Largeltem (q)))
        Largeltem (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
  Smallltem = x
end if

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

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

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

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

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

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

اثبات قضایای فوق در [۵] آمده است.

قضیه ۱۱: زمان پاسخ عملگرهای رکورد بعد و رکورد دارای بزرگترین 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 میتوان نتیجه گرفت که 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 بنابراین زمان پاسخ آنها O (1) است.

۴- سازگاری درساختمان داده موازی

در ادامه بحث سازگاری در ساختمان داده های موازی تعریف میشود و سپس سازگاری در مش 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 مش جستجوی دودوئی کامل روی مش یا ابر مکعب سازگار نیست.

اثبات: فرض کنید عملگر درج و سپس عملگر جستجو بترتیب وارد مش شوند. اگر عملگر درج سبب چرخش در مش 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 مش جستجوی دودوئی کامل روی مش یا ابر مکعب تضمین میگردد. این رویه بصورت زیر میباشد. عملگر های دیکشنری برای اجرا به ریشه مش 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 ایصورت 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 ذ رکورد بعد باید با دو واحد تاخیر آغاز شود. یک واحد از این تاخیر مربوط به حالت زیر میباشد. اگرکلید x مساوی کلید رکورد Largeltem 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 Largeltem پدر سمت راست این پردازنده قرار دارد و این خود نیاز به یک ارتباط در جهت عکس حرکت عملگرهای ساختمان داده دارد و باعث میشود تا یک تاخیر اضافه نیاز گردد.

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

قضیه ۱۴: توان عملیاتی مش 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 مش جستجوی دودوئی کامل برای یک ماشین های 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 برای عملگرهای کوچکترین کلید، بزرگترین کلید و جستجو نیاز به یک واحد تاخیر دارند.

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

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

۲- با توجه به اینکه شبکه پخش عملگرها یک مسیر یکتا بین ریشه و گره های دیگرمش جستجوی دودوئی کامل ایجاد میکند بنابراین 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 برسند و این باعث میگردد تا سازگاری در مش جستجوی دودوئی کامل بسادگی قابل پیاده سازی باشد

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

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

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

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

عملگر	Insert	Max	XMin	Next	Min	Delete	Search
مش 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 Schroder و Schmeck	O(1)	•	•	•	O(1)	O(1)	O(1)
ماشین Santoro و Dehne	O(1)	•	•	•	O(1)	O(1)	O(1)

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

عملگر	Search	Max	XMin	Next	Min
مش جستجوی دودوئی کامل	O(n)	O(n)	O(1)	O(n)	O(1)
ماشین Youn و Young	O(n)	•	•	•	O(1)
ماشین Schroder و Schmeck	O(n)	•	•	•	O(1)
ماشین Santoro و Dehne	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. Betley 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 Mishigan, Ann arbor, Mishigan, pp. 151-156, 1984.

[16] J. Ghosh, S. K. Das, and A. John, "Concurrent Processing of Linearly 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 California, 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.