# CHUYÊN ĐỀ CẤU TRÚC DỮ LIỆU ĐẶC BIỆT

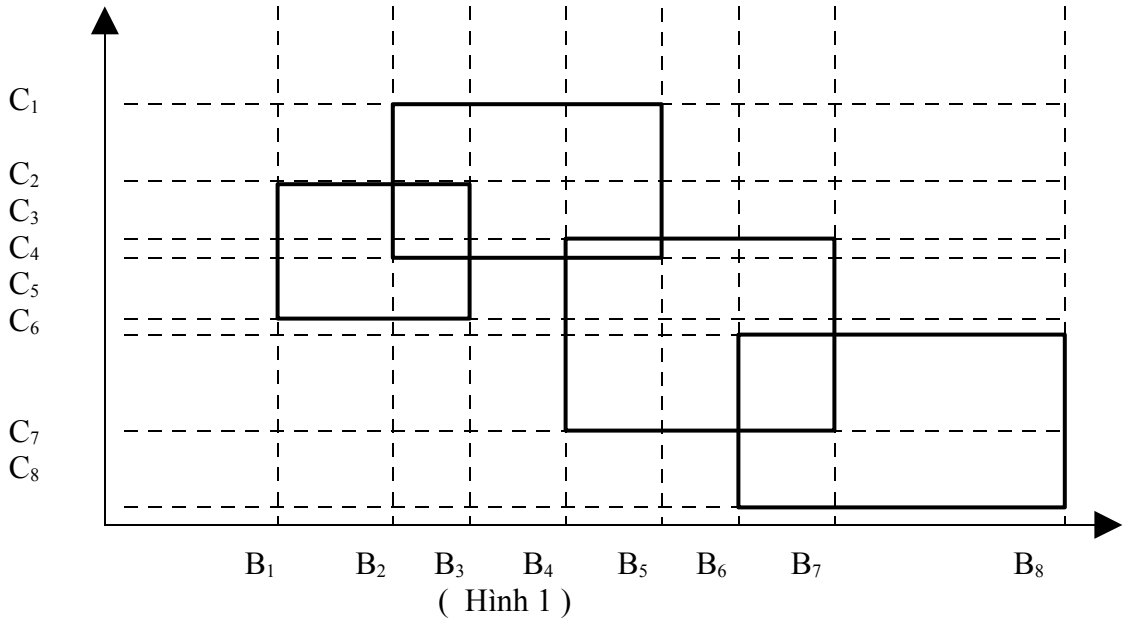
## A . Lý thuyết :

Trong chuyên đề này ta sẽ nhắc tới 2 loại cấu trúc đặc biệt , đó là Interval Tree và Binary Index Tree. Đó là 2 cách tổ chức dữ liệu rất thông minh , việc tổ chức này cũng dẫn tới việc tìm ra những thuật toán hay với cấp độ trung bình thấp O(NlogN) . Và để trình bày ý tưởng của các thuật toán này ta sẽ xem xét nó thông qua các bài toán cụ thể để có thể hiểu rõ hơn.

### I . Interval Tree :

**Bài toán** : Cho N hình chữ nhật trong mặt phẳng toạ độ. Hãy tính diện tích bị phủ bởi N hình chữ nhật này.
Giới hạn : + $1 \leq N \leq 2000$. Các toạ độ đều là số nguyên .
+ Time limit 0.5 s , bộ nhớ 200 KB.

**Phân tích :** Đối với bài toán này ta có thể giải bằng giải thuật thông thường với cấp độ $O(N^2)$. Đó là sắp xếp các hình chữ nhật theo toạ độ Y , sau đó tính diện tích bị phủ giữa 2 khe. Tổng diện tích bị phủ sẽ là tổng diện tích bị phủ giữa 2 khe (H.1).

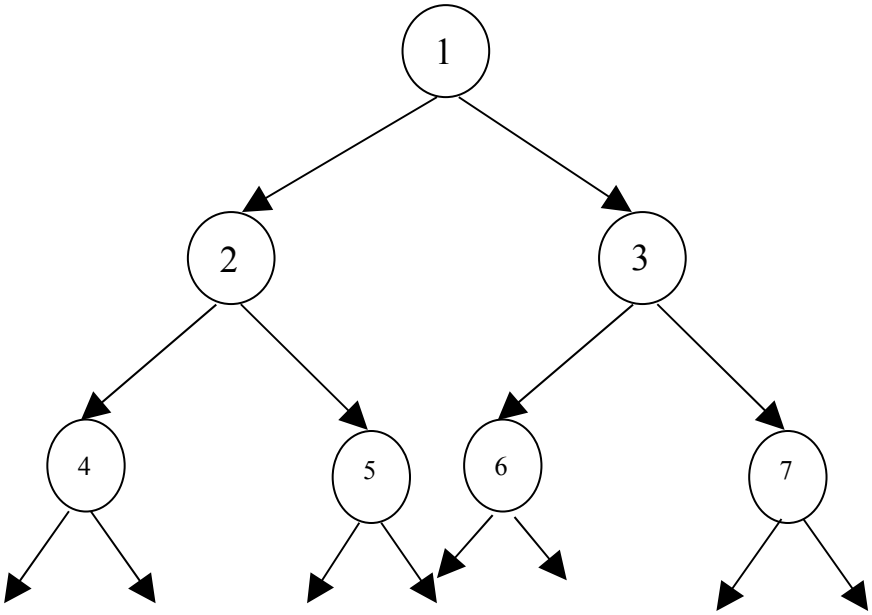


( Hình 1 )

( Ở hình 1 ta thấy có các khe $B_1B_2$ , $B_2B_3$ ,…, $B_7B_8$. )
Vì đã sắp xếp các HCN tăng dần theo tung độ nên với mỗi khe ta chỉ cần thao tác đơn giản là xét từ 1 -> N những HCN nào phủ lên khe đó mà thôi. Có tất cả khoảng N*2 – 1 khe , với mỗi khe ta xét N HCN -> Cấp độ chính xác $O(2*N^2)$.
Rõ ràng với N >= 2000 thì trong vòng 0.5 s chương trình khó có thể trả ra kết quả ngay được.

Chắc hẳn rất nhiều bạn cũng sẽ nghĩ ra thuật toán này và có thể sẽ băn khoăn một điều rằng liệu có cách gì để chỉ xét mỗi hình chữ nhật đúng một lần hay không ?
Cấu trả lời là : nếu muốn chỉ xét 1 lần thì hiện thời mình cũng không biết nhưng mà mình biết có thuật toán có thể đáp ứng yêu cầu với số lần xét là 2 lần !

**Nội dung thuật toán và cách làm:** Nếu như ở thuật toán $O(N^2)$ ta chỉ xét các khe theo hoành độ ( các khe B ) thì ở đây ta lại quan tâm tới khe theo tung độ ( các khe C ) . Tuy nhiên về mặt bản chất thuật toán vẫn không có gì thay đổi vẫn chỉ là tính diện tích giữa các khe ( hoành độ ) mà thôi. Ta sẽ phân tách 1 HCN ra thành 2 đỉnh :
+ Đỉnh 1 là đỉnh trái dưới . ( ta cứ gọi là đỉnh Mở của 1 HCN )
+ Đỉnh 2 là đỉnh phải trên.. ( ta cứ gọi là đỉnh Đóng của 1 HCN )
Với mỗi đỉnh ta sẽ lưu 4 thông số: toạ độ , và nếu là đỉnh 2 ta lưu tung độ đỉnh 1, nếu là đỉnh 1 ta lưu tung độ đỉnh 2 và một biến cho biết đỉnh đó là trái dưới hay phải trên. Rõ ràng với 2 đỉnh này thôi là cũng có thể đại diện cho 1 HCN được rồi.
➔Ta sẽ có 2*N đỉnh và 1 dãy $C_1,\ldots C_{2*N-1}$( tất nhiên dãy này đã sắp xếp theo thứ tự tăng dần). Gọi tập 2*N đỉnh này là tập đỉnh HCN.
Ta sắp xếp 2*N đỉnh theo thứ tự tăng dần của hoành độ. Bây giờ ta sẽ tính diện tích bằng cách tính tổng diện tích của các khe ( hoành độ ) giữa 2 điểm liên tiếp trong số 2*N điểm nói trên. Ta xét tuần tự các khe từ khe 1 -> khe 2*N-1.
Với dãy C ta sẽ tổ chức 1 cây nhị phân đầy đủ như sau :




….
Trong đó đỉnh 1 lưu số phần bị phủ lên tung độ từ $C_1$ -> $C_{2*N}$.
đỉnh 2 lưu số phần bị phủ lên phủ lên tung độ từ $C_1$ -> $C_N$, đỉnh 3 lưu số phần bị phủ lên tung độ từ $C_{N+1}$ -> $C_{2*N}$.
đỉnh 4 lưu số phần bị phủ lên tung độ từ $C_1$ -> $C_{N \text{ div } 2}$, đỉnh 5 lưu số phần bị phủ lên tung độ từ $C_{N \text{ div } 2+1}$ -> $C_N$ , đỉnh 6 lưu số phần bị phủ lên tung độ từ $C_{N+1}$ -> $C_{N+N \text{ div } 2}$ , đỉnh 7 lưu số phần bị phủ lên tung độ từ $C_{N+N \text{ div } 2 +1}$ -> $C_{2*N}$. v.v…

Với mỗi đỉnh trên cây nhị phân này ta có 2 thông số cần lưu đó là số HCN đang phủ lên đoạn này và phủ lên là bao nhiêu .
Bây giờ ta sẽ xét 2*N đỉnh trong tập các đỉnh của HCN.
Xét tới khe L ( hoành độ ) giữa đỉnh i và i+1 ta làm như sau :

Nếu đỉnh i là đỉnh Mở của một HCN R , nó có tung độ là Y1 , tung độ của đỉnh còn lại là Y2, tức là hiện thời nó sẽ phủ lên đoạn từ Y1 -> Y2 ( tung độ ).
Ta xét đoạn Y1 -> Y2 này trên cây nhị phân mà ta vừa dựng xong.

Xét tới nút P của cây, ( nút P này phủ từ $C_S$ -> $C_F$ ):
- Nếu Y1 <= $C_S$ , $C_F$ <= Y2 thì ta có thể thấy là HCN R này đã phủ lên toàn bộ tung độ trong đoạn từ $C_S$ -> $C_F$ trong khe L này. -> Ta sẽ phải sửa lại thông số của nút P này đó là tăng số HCN phủ đoạn này lên 1 và cho biết đoạn này đã bị phủ toàn bộ = $C_F$ – $C_S$.
- Nếu Y1 >= $C_F$ hoặc Y2 <= $C_S$ thì suy ra đoạn Y1- > Y2 này hoàn toàn chẳng phủ gì lên đoạn $C_S$ -> $C_F$ cả -> Ta không phải xét tới các nút con của nó nữa.
- Nếu đoạn [Y1,Y2] $\cap$ [$C_S$,$C_F$] $\neq \varnothing$ thì ta sẽ gọi tới các nút con của nó, xét tiếp các nút con của nó với đoạn Y1,Y2 này.

Chương trình minh hoạ :

```
Procedure Mo( Y1 , Y2 ,  P ,  C[S] , C[F]  :  Integer)  ;
Var
        mid : Integer ;
Begin
        If  (Y1 >= C[F]) or (Y2 <= C[S]) then Exit ;
        If  (Y1 <= C[S]) and (C[F] <= Y2) then Begin
             SoHCNphu[P] := SoHCNphu[P] + 1 ;
             Biphu[P]  := C[F] – C[S] ;  Exit;
        End ;
        If S+1 >= F then Exit ; { tức là nút P này là nút lá }
        mid := (S+F) div 2 ;
        Mo( Y1 , Y2 , P*2 , C[S] , C[mid]  )     ;
        Mo(  Y1 , Y2 , P*2+1, C[mid] , C[F] ) ;
        If SoHCNphu[P] = 0 then Biphu[P] := Biphu[P*2] + Biphu[P*2+1] ;
End ;
```

Nếu đỉnh i là đỉnh Đóng của một HCN R , nó có tung độ là Y2 , tung độ của đỉnh còn lại là Y1, tức là hiện thời nó sẽ phủ lên đoạn từ Y1 -> Y2 ( tung độ ).
Ta xét đoạn Y1 -> Y2 này trên cây nhị phân :
Xét tới nút P của cây, ( nút P này phủ từ $C_S$ -> $C_F$ ):
- Nếu Y1 <= $C_S$ , $C_F$ <= Y2 thì ta có thể thấy là HCN R này đã phủ lên toàn bộ tung độ trong đoạn từ $C_S$ -> $C_F$ trong khe L này. -> Ta sẽ phải sửa lại thông số của nút P này đó là giảm số HCN phủ đoạn này lên 1 , tức là một hình chữ nhật đã không còn phủ lên đoạn này nữa. Nếu như số HCN phủ lên đoạn này = 0 -> Đoạn này sẽ bị phủ lên một đoạn = tổng số phần bị phủ của 2 nút con của nó . ngược lạI ta không cần phải xét tới nút con của nó nữa

•Nếu Y1 >= $C_F$ hoặc Y2 <= $C_S$ thì suy ra đoạn Y1- > Y2 này hoàn toàn chẳng phủ gì lên đoạn $C_S$ -> $C_F$ cả -> Ta không phải xét tới các nút con của nó nữa.
•Nếu đoạn [Y1,Y2] ∩ [$C_S$,$C_F$] ≠ ∅ thì ta sẽ gọi tới các nút con của nó, xét tiếp các nút con của nó với đoạn Y1,Y2 này.
Chương trình minh hoạ :

```
Procedure Dong(Y1 , Y2 , P , S , F : Integer ) ;
Var
        mid : Integer ;
Begin
        If (Y1 >= C[F])or(Y2 <= C[S]) then Exit ;
        If  (Y1 <= C[S] )and(C[F] <= Y2) then Begin
             SoHCNphu[P] := SoHCNphu[P] – 1 ;
             If SoHCNphu[P] > 0 then Exit ;
             BiPhu[P] := BiPhu[ P*2 ] + BiPhu[P*2+1] ;
             Exit ;
       End ;
       If  S + 1 >= Fn then Begin { Tức là P là nút lá }
           Biphu[P] := 0 ;
           Exit ;
       End ;
       mid := (S+F) div 2 ;
       Dong( Y1 , Y2 , P*2     , S , mid ) ;
       Dong( Y1 , Y2 , P*2+1 , mid ,F ) ;
       If SoHCNphu[P] = 0 then Biphu[P] := Biphu[P*2] + Biphu[P*2+1] ;
End ;
```

Như vậy Biphu[1] cho ta biết tới khe L này thì tung độ từ $C_1$ -> $C_{2*N}$ đã bị phủ là bao nhiêu , diện tích bị phủ khe L = Độ rộng * Biphu[1] . Sau đây là chương trình mô tả đoạn này :

```
Procedure Solve ;
Var
       Dientich , Rong , i : LongInt ;
Begin
        Dientich := 0 ;
        Mo( A[1].Y1 , A[1].Y2 , 1 , 1 , N ) ;
        For i  := 2 to 2*n do Begin
             Rong := A[i].x – A[i-1].x ;
             Dientich := Dientich + Rong * BiPhu[1] ;
             If A[i].Y1 < A[i].Y2 then Mo(  A[i].Y1 , A[i].Y2 , 1 , 1 , N )
             Else Dong( A[i].Y2 ,  A[i].Y1 , 1 , 1 , N ) ;
        End;
End ;
```

Ta có thể khẳng định thuật toán này là hoàn toàn đúng đắn bởi khi ta xét tới 1 điểm Đóng i thì chắc chắn tồn tại 1 điểm Mở j đã xuất hiện trước đó , và 2 điểm i và j này là đại diện cho 1 HCN R , HCN R này có hoành độ bắt đầu từ điểm j và kết thúc ở điểm i, nên chừng nào chưa xét tới điểm i thì HCN R này vẫn tồn tại , vẫn phủ lên 1 số đoạn nào đó

của tung độ. Mỗi khi ta gặp 1 điểm Mở tức là gặp 1 HCN có cạnh bên trái có hoành độ = điểm đang xét , và khi gặp một điểm Đóng tức là 1 HCN đã bị loại khỏi vùng đang xét và sẽ không được xét tới sau này nữa.

Như vậy bài toán đã được giải quyết. Với mỗi lân cập nhật đỉnh i vào cây nhị phân ta mất logN bước, có tất cả 2*N đỉnh -> cấp độ thuật toán O(2*NlogN), đúng như đã nói ở trên ở đây ta chỉ xét mỗi HCN thông qua 2 điểm , mỗi điểm đúng 1 lần.

**Ý nghĩa cây nhị phân :** Như vậy ta có thể thấy mỗi nút P của cây nhị phân đại diện cho một đoạn , mà giá trị của nó = tổng giá trị của các đoạn con . Bởi vậy nó giúp ta không phải truy xuất tới tất cả những nút mà chỉ thông qua một số nút cha mà thôi. ( Ví dụ ở đây là Y1 <= C[S] , C[F] <= Y2 , tức là đã phủ lên cả đoạn rồi , không phải xét các đoạn con làm gì nữa )

**Interval Tree :** Cây nhị phân mà ta sử dụng trong bài tập nói trên chính là Interval Tree. Vậy khái quát lại thì Interval Tree là gì ? Đó là một cây nhị phân mà mỗi nút đại diện cho một "đoạn " hay một dãy các phần tử liên tiếp có chung một tính chất nào đó và các nút con của nó đại diện cho một đoạn nhỏ hơn. Khi ta muốn đếm , liệt kê xem một đoạn cho trước có bao nhiêu phần tử thoả mãn một tính chất X ( biểu diễn trên máy tính được , thông thường chỉ là quan hệ > hoặc <) cho trước, thì khi xét tính chất này trên một nút của cây nhị phân thì xảy ra 3 tình huống :

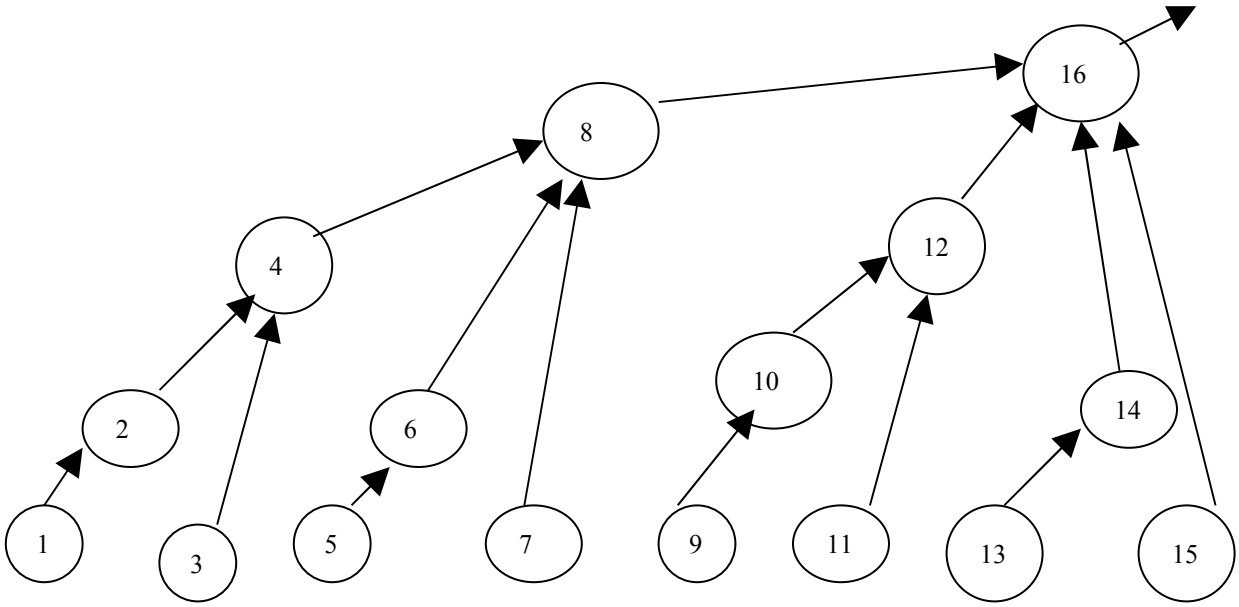
- Cả đoạn đều thoả mãn tính chất này , khi đó số phần tử thoả mãn trong đoạn đó = số phần tử của đoạn.
- Cả đoạn đều không thoả mãn tính chất -> số phần tử thoả mãn trong đoạn đó = 0.
- Có một số phần tử thoả mãn và các phần tử này nằm liên tiếp nhau trong đoạn đang xét. Khi đó ta sẽ lại phải kiểm tra với 2 nút con của nó. Nút con trái = nửa đoạn bên trái , nút con phải = nửa đoạn bên phải.

Khi muốn cập nhật thêm phần tử hay giảm vào đoạn ta cũng làm như vậy . Về mặt dung lượng bộ nhớ thì Internval Tree cần 2*N phần tử ( = số nút của cây ) nhưng có thể có nhiều trường hợp bị suy biến nên tốt nhất nên để 4-> 8*N .

Tuy nhiên nói thì là như vậy nhưng ta cũng cần làm nhiều bài tập mới có thể nắm rõ , sử dụng thuần thục nó được.

## II . Binary Index Tree :

Binary Index Tree cũng là một mô hình cây và nó cũng không khác Interval Tree về mục đích sử dụng , lấy dữ liệu được cập nhật từ nút con… . Về nguyên tắc thì bất cứ bài nào giải được bằng Binary Index Tree cũng đều đưa về Interval Tree được nhưng chưa chắc đã có chiều ngược lại ( theo ý kiến chủ quan của mình ) . Thuật toán cũng chỉ có cấp độ O(NlogN) nhưng tốt hơn ở chỗ là nó không cần lưu tất cả 2*N nút mà chỉ lưu N nút mà thôi.Sau đây là mô hình của cây Binary Index Tree :




Mỗi nút X ở đây đại diện cho các cho nút X và các nút con của X. Rõ ràng ta thấy mô hình của nó khác hẳn so với mô hình của cây nhị phân . Sau đây là mô tả chi tiết đồ thị :

+ Nếu đồ thị chỉ có 1 nút -> không có cung nào cả.
+ Nếu nút i lẻ , nó sẽ có cung nối với nút i+1 ( là nút chẵn ) .
+ Không xét các đỉnh lẻ nữa , các đỉnh chẵn còn lại sẽ được đánh số lại ( ngầm định trong đầu thôi ), số thứ i = i shr 1 . Quay lại bước 1.

Trong Binary Index Tree , để biết được nút i có cha là nút nào người ta sử dụng công thức đã được CM như sau :

Cha(i) = i + i and ( i xor ( i-1) ) ;

Mỗi khi cập nhật một phần tử ở có giá trị tương ứng là X ta sẽ tăng số phần tử = X lên: A[x] := A[x] + 1 và gửi thông báo lên cho cha của nó , cha của nó lại tiếp tuc gửi lên cho tới khi nào > N thì thôi. Mỗi lần gửi thông báo , ta lại tăng số phần tử của cha nó lên , tức là A[cha] := A[cha] + 1 . Như vậy cũng có nghĩa là A[x] của ta lưu lại số phần tử có giá trị x và số phần tử của các con của nó.

Và để kiểm tra xem từ 1 -> X có bao nhiêu phần tử người ta sử dụng một cách thức đệ quy rất thông minh như sau :

Số phần tử 1 -> X = Số phần tử lưu được ở nút X
+ Số phần tử 1 -> ( X – X and ( X xor ( X-1) ) ) .

Ví dụ mhư muốn biết có bao nhiêu phần tử ở <= 11 chẳng hạn :

Số phần tử = A[11] + Số phần tử( 1 => 11 – 11 and (11 xor 10) ) .
= A[11] + Số phần tử( 1 => 10 ) .
= A[11] + A[10] + Số phần tử (1 => 8) ;
= A[11] + A[10] + A[8] + Số phần tử ( 1 => 0) = A[11] + A[10] + A[8]. ( hoàn toàn chính xác , nhìn vào hình vẽ ).

Cấp độ của Binary Index Tree đã được CM rằng luôn luôn nhỏ hơn O(NlogN) , bởi vậy chương trình chạy rất nhanh , hơn nữa lại còn nhanh hơn rất nhiều so với dùng Interval Tree, bộ nhớ sử dụng cũng ít hơn .

## B. Bài tập ứng dụng :

### Bài 1 : Electronic Auction ( Đấu giá lợn sắt )

Có một sự thiếu hụt lợn sắt ở một đất nước nọ. Bởi vậy lợn sắt được bán đấu giá. Chúng được bán ở các phiên đấu giá điện tử . Khách hàng khi đến mua có quyền đặt giá của mình . Họ sẽ thông báo cho ban quản lý giá của mình sẵn sàng đưa ra để mua về một con lợn sắt ( số tiền này nằm trong khoảng 0.01 VND -> 10000.00 VND và luôn có chính xác 2 chữ số sau dấu phẩy, tức là không bao giờ có chuyện khách hàng đặt giá là 0.211 hay 3.412 mà chỉ có thể là 0.21 hoặc 3.41 mà thôi ). Hết lần này tới lần khác những người bán sẽ đưa ra K con lợn để đấu giá , và mỗi con lợn sẽ được bán cho K người đầu tiên trả giá >= X. Nếu như không có đủ K người thì số lợn còn lại sẽ bị chuyển tới nước khác ngay lập tức , và không được bán tiếp trên đất nước này nữa .

Khách hàng cũng có thể thông báo huỷ bỏ cái giá mà mình đã đưa ra . Sau mỗi cuộc giao dịch , khách hàng vẫn tiếp tục mua bán tiếp với cái giá mà họ đã thông báo cho tới khi nào họ thông báo huỷ bỏ giá mà mình đưa ra thì thôi. Mỗi con lợn sắt được bán thì ban quản lý đấu giá được nhận hoa hồng là 0.01 VND. Hãy tính xem sau khi kết thúc tất cả các cuộc giao dịch thì ban quản lý lãi bao nhiêu tiền.

Giới hạn :
Freepascal : + Số dòng trong file Input <= 100000 dòng.
+ Time limit 0.5 s , bộ nhớ 5000 KB.
Turbo Pascal:+ Số dòng trong file Input <= 60000 dòng. Giá tiền giảm xuống <= 300.
+ Time limit 0.5 s , bộ nhớ 200KB .

INPUT
Gồm nhiều dòng , mỗi dòng có thể có dạng 1 trong 3 trường hợp sau :
- "BID X " : Cho biết vừa có thêm 1 người thông báo giá của mình là X VND.
- "DEL X" : Cho biết vừa có 1 người thông báo huỷ cái giá X mà mình đã đưa ra.
- "SALE X K" : Cho biết có một người bán vừa quyết định dem bán K con lợn sắt với cái giá ít nhất cho mỗi con lợn là X VND. K người đầu tiên trả giá >= X sẽ được mua mỗi người 1 con.

Dòng cuối cùng ghi 1 từ duy nhất "QUIT" thông báo đã kết thúc tất cả các phiên giao dịch, các cuộc mua bán đều đã kết thúc .
OUTPUT
1 số thực duy nhất ( cũng ghi chính xác 2 chữ số sau dấu phẩy ) là lãi mà ban quản lý thu được .

Ví dụ :

| Input | Output |
|---|---|
| BID 0.01<br>BID 10000<br>BID 5000 | 0.06 |

| | |
|---|---|
| BID 5000<br>SALE 7000 3<br>DEL 5000<br>SALE 3000 3<br>SALE 0.01 3<br>QUIT | |

Giải thích :
- 4 dòng đầu tiên cho biết có 4 người đã đưa ra giá của mình, đó là các giá 0.01 , 10000 , 5000 , 5000.
- Dòng thứ 5 cho biết có một người đã đem bán 3 con lợn mỗi con giá tối thiểu là 7000 VND.->Chỉ có 1 người mua là người đặt mức giá 10000, còn lại 2 con lợn sẽ bị chuyển đi, không bán nữa -> Lãi 0.01 đồng.
- Dòng thứ 6 cho biết có một người đã huỷ bỏ cái giá 5000 VND mà anh ta đưa ra. Tức là lúc này chỉ còn lại 3 người với 3 mức giá 0.01 , 10000 , 5000 VND.
- Dòng thứ 7 cho biết có một người đã đem bán 3 con lợn mỗi con giá tối thiểu là 3000 VND.->Chỉ có 2 người mua là người đặt mức giá 10000 và 5000, còn lại 1 con lợn sẽ bị chuyển đi, không bán nữa -> Lãi 0.02 đồng.
- Dòng thứ 8 cho biết có một người đã đem bán 3 con lợn mỗi con giá tối thiểu là 0.01 VND.->Có 3 người mua là người đặt mức giá 10000 ,0.01 và 5000-> Lãi 0.03 đồng.
- Dòng 9 Cho biết các phiên giao dịch đã kết thúc .
- Vậy tổng lãi sẽ là 0.01 + 0.02 + 0.03 = 0.06 VND.

**Thuật giải :** Đây là một bài điển hình cho việc sử dụng Binary Index Tree, nếu biết sử dụng khéo thì cũng có thể sử dụng Interval Tree được.