Chuyên đề: CÂY PHÂN ĐOẠN (SEGMENT TREES) VÀ MỘT SỐ ỨNG DỤNG

ĐẶT VẤN ĐỀ

Sử dụng cấu trúc dữ liệu nâng cao để giải quyết các bài toán thi học sinh giỏi cấp Quốc gia và khu vực là một trong những kĩ năng cần có của học sinh dự thi.

Cây phân đoạn là một cấu trúc dữ liệu được sử dụng khá hiệu quả để giải quyết một số bài toán của các kì thi trong những năm gần đây.

Phần nội dung được trình bày sau đây là những kinh nghiệm bản thân tôi rút ra được sau một thời gian tìm tòi nghiên cứu và đưa vào hướng dẫn học sinh đội tuyển.

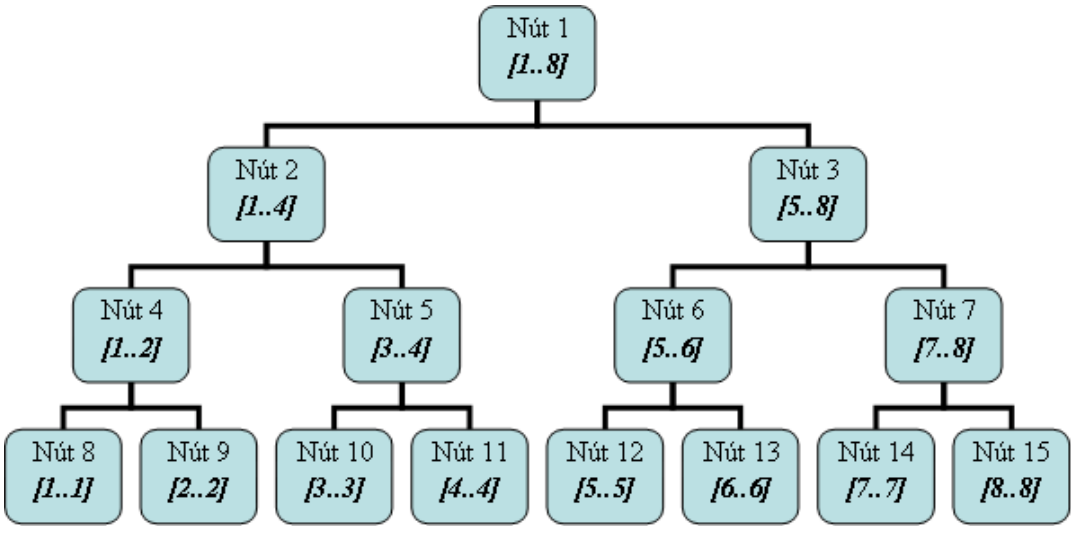
NỘI DUNG

1. Cây phân đoạn

Cây phân đoạn là một cấu trúc dữ liệu cho phép thực hiện các thao tác truy vấn cũng và cập nhật trên một đoạn các phần tử của mảng với chi phí thực hiện mỗi thao tác có độ phức tạp là hàm logarit. Cây phân đoạn là một cây nhị phân đầy đủ. Để quản lí các phần tử trong đoạn ***[i..j]*** của mảng, cây phân đoạn được tổ chức như sau:

- Nút đầu tiên lưu thông tin của đoạn ***[i..j]***.
- Nếu ***i<j*** thì nút con bên trái lưu thông tin đoạn ***[i..(i+j) div 2]*** và nút con bên phải lưu thông tin đoạn ***[(i+j) div 2 + 1..j]***.
- Nếu ***i=j*** thì nút này không có nút con.

Ví dụ: Sau đây là cây phân đoạn quản lí đoạn ***[1..8]***:



Mỗi nút của cây lưu trữ thông tin của một đoạn cố định. Thông tin này thông thường là tổng các phần tử, giá trị lớn nhất, giá trị nhỏ nhất... của các phần tử trong đoạn. Các nút lá của cây lưu trữ các giá trị tương ứng với các phần tử của mảng.

Các thao tác khi làm việc trên cây phân đoạn tương ứng với một mảng N phần tử là:

- Tạo cây tương ứng với mảng đã cho với độ phức tạp O(NlogN).
- Truy vấn thông tin mỗi đoạn trên cây với độ phức tạp O(logN).
- Cập nhật thông tin mỗi phần tử trên cây với độ phức tạp O(logN).
- Cập nhật thông tin mỗi đoạn trên cây với độ phức tạp O(logN).

Ví dụ 1:

Cho một mảng ***N*** (***1≤ N≤10^5***) số nguyên ***a[1],a[2],..,a[N]*** (***1≤ a[i]≤10^9***). Có ***M*** (***1≤ M≤10^5***) câu hỏi, mỗi câu hỏi gồm hai số nguyên ***u***, ***v*** (***1≤ u ≤ v ≤N***), yêu cầu tìm số nguyên nhỏ nhất trong đoạn ***a[u]***,***a[u+1]***,..,***a[v]***. Thời gian thực hiện mỗi test là một giây.

Bài toán này có thể giải quyết được trọn vẹn bằng sử dụng cây phân đoạn với chỉ hai thao tác là tạo cây phân đoạn tương ứng với mảng đã cho và thực hiện truy vấn thông tin mỗi đoạn trên cây. Chi phí cho toàn bộ chương trình là O(Max(N,M) log N).

Mảng ***Tree[1..MaxNut]*** dùng để lưu cây phân đoạn tương ứng với mảng ***a[1..N]***. Mỗi nút của cây lưu giá trị nhỏ nhất của đoạn mà nó quản lý. Thủ tục tạo cây được viết như sau:

```
Procedure TaoCay(L,R: Int; Nut: int); {Tree[Nut] lưu giá trị nhỏ nhất của đoạn a[L..R]}
Var mid : int;
Begin
  If L=R then begin Tree[Nut] := A[L]; exit; end;
  mid := (L+R) div 2;
  TaoCay(L,mid, Nut*2); {tạo nút con bên trái lưu giá trị nhỏ nhất đoạn a[L..mid]}
  TaoCay(mid+1,R,Nut*2+1); {tạo nút con bên phải lưu giá trị nhỏ nhất đoạn a[mid+1..R]}
  Tree[Nut] := Min(Tree[Nut*2],Tree[Nut*2+1]);
End;
```

Để tạo cây phân đoạn ta cho mảng ***a[1..N]*** ta gọi *TaoCay(1,N,1);*.

Giá trị nhỏ nhất của đoạn ***a[u..v]*** được trả về thông qua lời gọi hàm *TruyVan(u,v,1,N,1)*. Hàm *TruyVan* được viết như sau:

```
FuncTion TruyVan(u,v: int; L,R: int; Nut: int): Int;
Var mid : int;
Begin
    If (u<=L)and(R<=v) Then exit(Tree[Nut]);
    mid :=(L+R) div 2;
    If v<=mid Then exit(TruyVan(u,v,L,mid,Nut*2);
    If u>mid Then exit(TruyVan(u,v,mid+1,Nut*2+1);
    TruyVan := Min(TruyVan(u,v,L,mid,Nut*2),TruyVan(u,v,mid+1,R,Nut*2+1));
End;
```

Ví dụ 2:

Cho một mảng ***N*** (***1≤ N≤10^5***) số nguyên ***a[1],a[2],..,a[N]*** (***1≤ a[i]≤10^9***). Có ***M*** (***1≤ M≤10^5***) câu hỏi, mỗi câu hỏi thuộc một trong hai dạng sau:

- ***1 p k***: thay giá trị ***a[p]*** bằng ***k*** (***1≤ p ≤N, 1≤ k≤10^9***);
- ***2 u v***: yêu cầu tìm số nguyên nhỏ nhất trong đoạn ***a[u],a[u+1],..,a[v]*** (***1≤ u ≤ v ≤N***).

Thời gian thực hiện mỗi test là một giây.

Bài toán này có thể giải quyết được trọn vẹn bằng sử dụng cây phân đoạn với ba thao tác là tạo cây phân đoạn tương ứng với mảng đã cho, thực hiện truy vấn thông tin mỗi đoạn trên cây và cập nhật thông tin mỗi phần tử của mảng trên cây. Chi phí cho toàn bộ chương trình này cũng là O(Max(N,M) log N).

Ở ví dụ này tổ chức dữ liệu và các thao tác tạo cây phân đoạn tương ứng với mảng đã cho, thực hiện truy vấn thông tin mỗi đoạn trên cây hoàn toàn giống ***ví dụ 1***. Thao tác cập nhật thông tin mỗi phần tử của mảng trên cây được viết như sau:

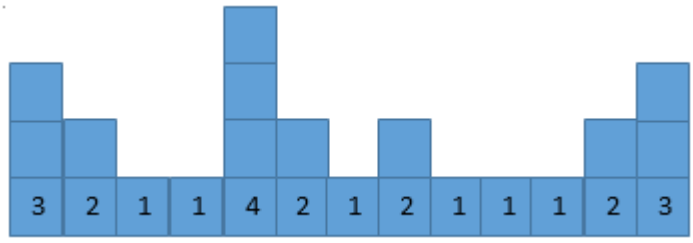
```
Procedure CapNhat(p:Int; k: int; L,R: Int; Nut: int);
Var mid : int;
Begin
  If L=R then begin Tree[Nut] := k; exit; end;
  mid := (L+R) div 2;
  If p<=mid Then CapNhat(p,k,L,mid,Nut*2) Else CapNhat(p,k,mid+1,R,Nut*2+1);
  Tree[Nut] := Min(Tree[Nut*2],Tree[Nut*2+1]);
End;
```

Mỗi lần gặp truy vấn dạng ***1 p k*** ta gọi thủ tục *CapNhat(p,k,1,N,1)* với chi phí thực hiện của thủ tục này là O(log N).

Ví dụ 3:

Cô bò Bessice rất thích trò chơi điện tử đặc biệt là trò chơi xếp gạch Tetris. Cô đưa yêu sách muốn nông dân John phải mua cho cô máy điện tử để giải trí, nếu không cô sẽ xúi giục các con bò khác nổi loạn. Để tránh một cuộc biểu tình do Bessice đứng đầu, John đành phải chiều lòng cô, ông mua cho cô máy điện tử xếp hình chỉ có 2 thanh thẳng đứng và nằm ngang vì tay cô bò chỉ có hai móng.

Có máy điện tử, ngày cô bò đi ăn cỏ, lấy sữa, tối về lại tụ tập các con bò khác sang chuồng mình để chơi. Tại lần chơi này, cô thấy trên màn hình đã có sẵn ***N*** thanh thẳng đứng xếp sát nhau đánh chỉ số từ ***1*** đến ***N***, các thanh có độ cao là ***H[1],H[2],..,H[N]***.



Phía trên của màn hình xuất hiện lần lượt ***M*** thanh nằm ngang, mỗi thanh có độ dài lần lượt là ***L[1]***,***L[2]***,..,***L[M]***. Các thanh xuất hiện được xác định tọa độ của mép bên trái thanh đó là vị trí ***P[1]***,***P[2]***,.., ***P[M]***. Khi một thanh rơi xuống và mép dưới thanh tiếp xúc với một cột nào đó thì mới xuất hiện thanh tiếp theo trên màn hình.

Bạn hãy giúp Bessice xác định xem độ cao của mỗi thanh ngang khi nó tiếp xúc với một cột nào đó trên màn hình.

- ***Dữ liệu nhập***:

- Dòng đầu chứa hai số ***N*** và ***M*** ($1 \le N, M \le 10^5$);
- Dòng thứ hai chứa ***N*** số nguyên không âm ***H[1]***,***H[2]***,..,***H[N]*** ($1 \le H[i] \le 10^9$);
- ***M*** dòng tiếp theo mỗi dòng chứa hai số P_i và L_i ($1 \le P_i, L_i \le 10^5$) – là vị trí xuất hiện và chiều dài của thanh thứ ***i***.

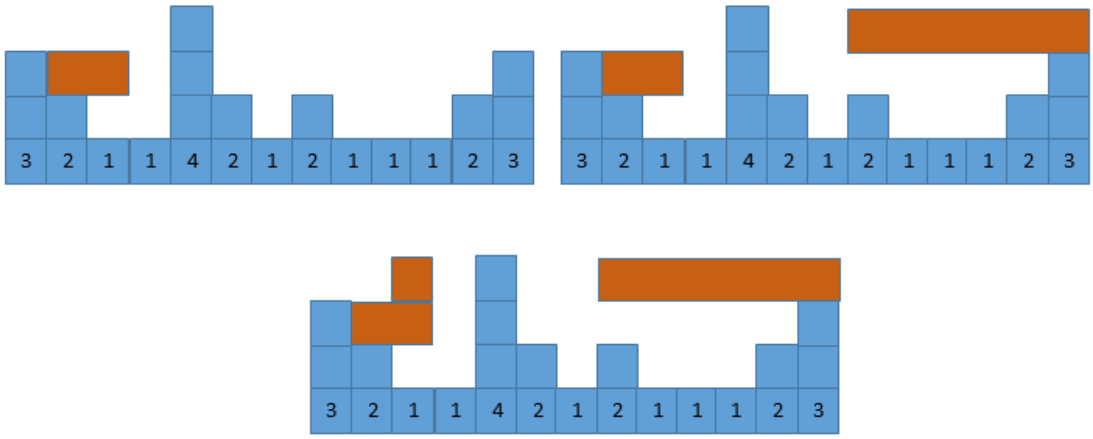
- ***Dữ liệu xuất:***

- ***M*** dòng, mỗi dòng thứ ***i*** là độ cao của thanh ngang thứ ***i*** khi rơi xuống.

Ví dụ

Dữ liệu nhập	***Dữ liệu xuất***
13 3 3 2 1 1 4 2 1 2 1 1 1 2 3 2 2 8 6 3 1	3 4 4

Giải thích:



Giới hạn: thời gian thực hiện mỗi test là một giây.

Bài toán này có thể giải quyết được trọn vẹn bằng sử dụng cây phân đoạn với ba thao tác là tạo cây phân đoạn tương ứng với mảng đã cho, thực hiện truy vấn thông tin mỗi đoạn trên cây và cập nhật thông tin mỗi đoạn của mảng trên cây. Chi phí cho toàn bộ chương trình này cũng là O(Max(N,M) log N).

Ở hai ví dụ đầu tiên là truy vấn tìm giá trị nhỏ nhất trong một đoạn của mảng, ví dụ này yêu cầu tìm giá trị lớn nhất. Điều này xử lí hoàn toàn tương tự. Ví dụ này chỉ khác ***ví dụ 2*** là thay vì cập nhật giá trị một phần tử ta phải cập nhật giá trị trên một đoạn. Nếu sử dụng cập nhật từng phần tử của đoạn như ở ***ví dụ 2*** thì không thể đảm bảo yêu cầu thời gian thực hiện vì chi phí cho bài toán lúc này là O(M.N log N).

Mỗi nút trên cây phân đoạn cần phải lưu trữ hai thông tin, đó là giá trị lớn nhất của đoạn và đánh dấu đoạn đã được cập nhật giá trị mới nhất hay chưa. Do vậy dữ liệu để lưu trữ cây phân đoạn này gồm 2 mảng:

- Mảng ***Tree[1..MaxNut]*** dùng để lưu cây phân đoạn tương ứng với mảng ***H[1..N]***;
- Mảng ***CN[1..MaxNut]*** dùng để đánh dấu; ***CN[Nut]=True*** nếu như đoạn được quản lí bởi ***Nut*** đã được cập nhật, ngược lại ***CN[Nut]:=False***.

❖ Thủ tục tạo cây được viết như sau:

```
Procedure TaoCay(L,R: Int; Nut: int);
Var mid : int;
Begin
  CN[Nut] := True;{đánh dấu nút đã được cập nhật giá trị mới nhất}
  If L=R then begin Tree[Nut] := H[L]; exit; end;
  mid := (L+R) div 2;
  TaoCay(L,mid, Nut*2); {tạo nút con bên trái lưu giá trị nhỏ nhất đoạn a[L..mid]}
  TaoCay(mid+1,R,Nut*2+1); {tạo nút con bên phải lưu giá trị nhỏ nhất đoạn a[mid+1..R]}
  Tree[Nut] := Max(Tree[Nut*2],Tree[Nut*2+1]);
End;
```

Để tạo cây cho mảng ***H[1..N]*** ta thực hiện lời gọi *TaoCay(1,N,1);*

Khi thực hiện thay đổi giá trị lên một đoạn ***H[u..v]*** của mảng, ta chỉ tiến hành cập nhật giá trị này trên các đoạn con của đoạn ***[u..v]*** được quản lí bởi các nút gần gốc nhất. Những đoạn nhỏ hơn được quản lí bởi các nút xa gốc hơn được trì hoãn cập nhật giá trị này. Trước khi cập nhật hay truy vấn đến bất kì đoạn nào, đoạn đó cần được cập nhật thông tin mới nhất.

Trong các chương trình con tiếp theo có sử dụng thủ tục để cập nhật lại thông tin mới nhất của một đoạn được quản lý bởi nút tương ứng trên cây. Công việc này được mô tả bởi thủ tục *CapNhatMoi* như sau:

```
Procedure CapNhatMoi( Nut: int);
Begin
   If CN[Nut]=True Then exit; {Thông tin của đoạn Nut quản lý đã được cập nhật mới}
   Tree[Nut*2] := Tree[Nut];  CN[Nut*2] := False;
   Tree[Nut*2+1]:= Tree[Nut]; CN[Nut*2+1] := False;
   CN[Nut] := True;
End;
```

❖ Hàm truy vấn giá trị lớn nhất của đoạn ***H[u..v]*** trên cây:

```
FuncTion TruyVan(u,v: int; L,R: int; Nut: int): Int64;
Var mid : int;
Begin
        If L<R Then CapNhatMoi(Nut);{cập nhất giá trị mới của đoạn H[L..R]}
        If (u<=L)and(R<=v) Then exit(Tree[Nut]);
        mid :=(L+R) div 2;
        If v<=mid Then exit(TruyVan(u,v,L,mid,Nut*2);
        If u>mid Then exit(TruyVan(u,v,mid+1,Nut*2+1);
        TruyVan := Max(TruyVan(u,v,L,mid,Nut*2),TruyVan(u,v,mid+1,R,Nut*2+1));
End;
```

Để lấy giá trị lớn nhất của đoạn ***H[u..v]*** ta thực hiện lời gọi hàm *Kq := TruyVan(u,v,1,N,1);*

❖ Thủ tục cập nhật giá trị của đoạn ***H[u..v]*** bằng ***k*** được viết như sau:

```
Procedure CapNhat(u,v: Int; k: int; L,R: Int; Nut: int);
Var mid : int;
Begin
    If (v<L)or(R<u) then exit;
    If L<R then CapNhatMoi(Nut);
    If (u<=L)and(R<=v) Then begin
      Tree[Nut] := k; CN[Nut] := False; exit; end;
    mid := (L+R) div 2;
    CapNhat(u,v,L,mid,Nut*2);
    CapNhat(u,v,mid+1,R,Nut*2+1);
    Tree[Nut] :=Max(Tree[Nut*2],Tree[Nut*2+1]);
End;
```

Để thực hiện cập nhật giá trị đoạn ***H[u..v]*** bằng ***k*** trên cây phân đoạn ta thực hiện lời gọi chương trình con: *CapNhat(u,v,k,1,N,1);*

Thủ tục *XuLy* sau đây dùng để giải quyết bài toán:

```
Procedure Xuly;
Var i, P, L : int;  res : int64;
Begin
    readln(N,M);
    For i:=1 to N do read(H[i]);
    For i:=1 to M do begin     Read(P,L);
        res := TruyVan(P,min(P+L-1,N), 1,N, 1)+1; Writeln(res);
        CapNhat(P,min(P+L-1,N),res, 1,N,1);
    end;
End;
```

2. Một số bài tập áp dụng

1. http://vn.spoj.com/problems/AREA/
2. http://vn.spoj.com/problems/QMAX/
3. http://vn.spoj.com/problems/QMAX2/
4. http://vn.spoj.com/problems/LITES/
5. http://vn.spoj.com/problems/C11PAIRS/
6. http://vn.spoj.com/problems/NKINV/
7. http://vn.spoj.com/problems/NKLINEUP/
8. http://vn.spoj.com/problems/NKMAXSEQ/
9. http://vn.spoj.com/problems/GSS/

KẾT LUẬN

Cây phân đoạn là một cấu trúc dữ liệu rất phù hợp với những bài toán cần quản lý và truy vấn các thông tin trên một đoạn của mảng. Cấu trúc này tương đối dễ cài đặt. Tuy nhiên trong một số bài toán đặc thù nó không đạt được hiệu quả cao như Binary Index Tree hoặc là Heap ...

Trên đây chỉ là một chút ít kiến thức và kinh nghiệm bản thân đã tích lũy được trong thời gian qua. Dù đã cố gắng nhưng chắc chắn không thể tránh khỏi những thiếu sót. Rất mong nhận được ý kiến đóng góp của các thầy cô!