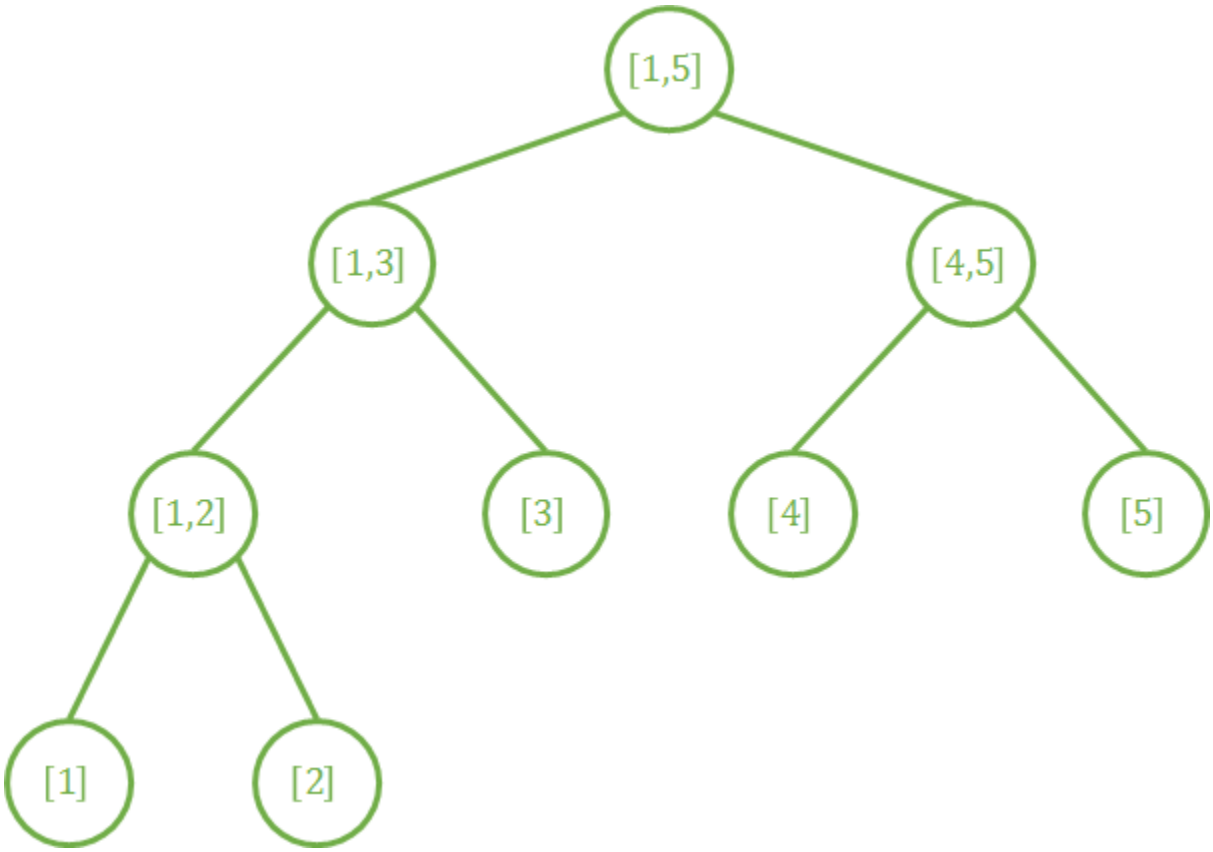
Chuyên đề: Interval Tree

1. Giới thiệu:

Cây Interval Tree – IT (hay còn gọi là cây Segment Tree - ST) là một cấu trúc dữ liệu được sử dụng rất nhiều, đặc biệt là trong các bài toán xử lý trên dãy số.

IT là một cây nhị phân đầy đủ (mỗi nút là lá hoặc có 2 nút con) với mỗi nút của cây sẽ quản lý một dãy các phần tử liên tiếp.

Với một dãy số gồm N phần tử, nút gốc sẽ lưu thông tin về đoạn $[1, N]$, nút con trái của nó sẽ lưu thông tin về đoạn $[1, \lfloor N/2 \rfloor]$ và nút con phải sẽ lưu thông tin về đoạn $[\lfloor N/2 \rfloor + 1, N]$. Nói một cách tổng quát: nếu nút A lưu thông tin đoạn $[L, R]$ thì 2 nút con A_1 và A_2 sẽ lưu thông tin của các đoạn $[L, \lfloor (L + R)/2 \rfloor]$ và đoạn $[\lfloor (L + R)/2 \rfloor + 1, R]$.



2. Cài đặt:

Chúng ta có thể cài đặt bằng mảng 1 chiều, phần tử thứ nhất của mảng thể hiện nút gốc. Phần tử thứ i sẽ có 2 phần tử con là $2\times i$ (nút con trái) và $2\times i+1$ (nút con phải), và có phần tử cha là $\lfloor i/2 \rfloor$.

Với cách cài đặt này, người ta đã chứng minh được bộ nhớ cần dùng cho IT không vượt quá $4\times N$ phần tử.

- *Dựng cây:* Gọi thủ tục $Build(x, left, right)$ → Dựng cây IT gốc x quản lý đoạn $[left, right]$.

1. Lưu trữ phạm vi: $l_x = left, r_x = right$.

2. Nếu $left = right$ → x là nút lá:

→ Cập nhật $leaf_{left} = x$ (tức nút lá chứa duy nhất một giá trị phần tử $left$ là nút x).

→ Lưu trữ thông tin của phần tử $left$ vào nút IT_x.

3. Nếu $left < right$ → x là nhánh:

→ Tính $middle = \lfloor (left + right)/2 \rfloor$.

→ Dựng cây IT từ nút con trái → Gọi $Build(2\times x, left, middle)$.

→ Dựng cây IT từ nút con phải → Gọi $Build(2\times x + 1, middle, right)$.

→ Cập nhật thông tin IT_x của đoạn $[left, right]$ bằng 2 nhánh con $IT_{2\times x}$ và $IT_{2\times x+1}$.

```
procedure build(x,left,right);
begin
    l[x]:=left; r[x]:=right;
    if left = right then
    begin
        leaf[left]:=x;
        <lưu trữ thông tin nút lá IT[x]>
    end
    else
    begin
        middle:=(left+right) div 2;
        build(2*x,left,middle);
        build(2*x+1,middle+1,right);
        <cập nhật IT[x] bằng IT[2*x] và IT[2*x+1]>
    end
end;
```

- *Cập nhật:* Thay đổi một phần tử thứ x trong dãy.

→ Cập nhật lại thông tin nút lá IT_{leaf_x}.

→ Cập nhật lại thông tin các nút cha của $leaf_x$.

```
procedure update(x:longint; <thông tin mới>);
begin
    k:=leaf[x];
    <cập nhật lại thông tin nút IT[k]>;
    while k>1 do // cập nhật cho đến khi k là nút
gốc
    begin
        k:=k div 2;
        <cập nhật IT[k] bằng IT[2*k] và IT[2*k+1]>
    end
end;
```

- *Trả lời truy vấn:* Trả lời thông tin đoạn $[a, b]$ → Truy cập tới từng đoạn con để xét:

→ $Query(a, b) = Request(1)$: Truy cập tới nút gốc chứa đoạn $[1, N]$ để xét.

→ Nếu $[a, b] \cap [l_x, r_x] = \emptyset$, trả về giá trị rỗng (không làm ảnh hưởng đến việc xét các đoạn khác).

→ Nếu $[a, b] \supseteq [l_x, r_x]$, tức là đoạn mà nút x quản lý là một phần của đoạn đang xét $[a, b]$, nên ta trả về thông tin của đoạn $[l_x, r_x]$, tức IT_x.

→ Trong trường hợp còn lại, tức $[a, b] \cap [l_x, r_x] \neq \emptyset$, ta truy cập tiếp đến 2 nút con $2 \times x$ và $2 \times x + 1$ để lấy thông tin và loại trừ đoạn không thuộc $[a, b]$.

```
function query(a,b:longint):<thông tin>;

   function request(x:longint):<thông tin>;
   begin
      // trường hợp [lx,rx] không giao [a,b]
      if (l[x]>b) or (r[x]<a) then
         exit(<giá trị rỗng>);
      // trường hợp [lx,rx] là đoạn con [a,b]
      if (a<=l[x]) and (b<=r[x]) then
         exit(IT[x]); // trả về thông tin đoạn
      <gọi request(2*x), request(2*x+1) và trả về thông tin>
   end;

begin
   query:=request(1) // truy cập tới nút gốc
end;
```

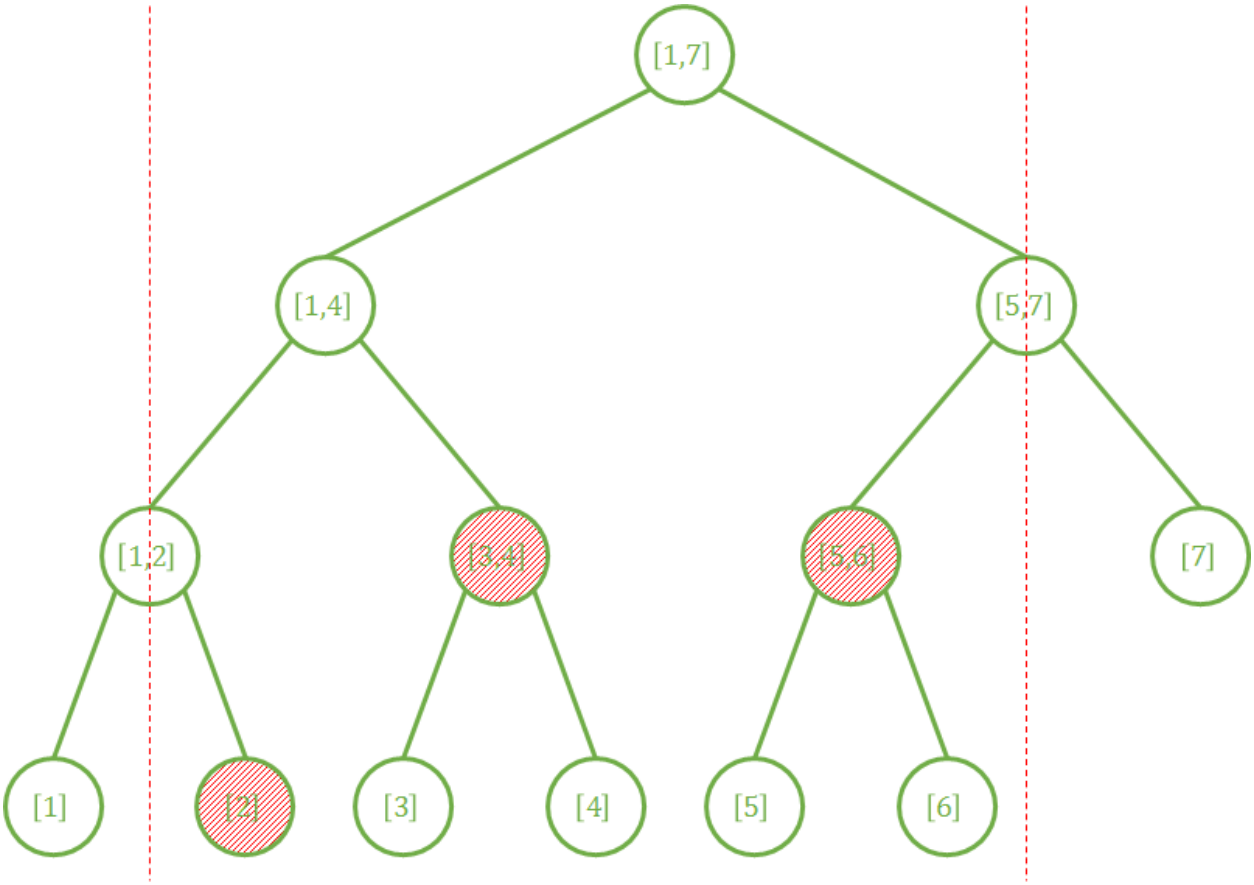
3. Phân tích thời gian chạy:

- Ở thao tác dựng cây, chiều rộng của cây là N và chiều cao là $\log_2 N$, vì vậy tốn $O(N \times \log_2 N)$.

- Ở thao tác cập nhật, tìm được nút lá chứa thông tin trực tiếp của phần tử x trong $O(1)$ và cập nhật các nút cha tốn khoảng $O(\log_2 N)$ (chiều cao của cây là $\log_2 N$), vì vậy thao tác này tốn khoảng $O(\log_2 N)$.

- Ở thao tác trả lời truy vấn, có thể thấy thao tác chỉ tốn khoảng $\log_2(b-a+1)$.

Ví dụ trong một dãy có 7 phần tử, xét lấy thông tin đoạn [2,6]:



Có thể thấy để lấy thông tin đoạn [2,6], đối với cây IT chỉ cần truy cập tới 3 nút [2], [3,4], [5,6] là được, vì đoạn [3,4] và [5,6] đã được tích hợp thông tin từ lúc dựng cây (và cập nhật nếu có) nên không cần xét đến các nút con. Tổng hợp thông tin của 3 đoạn này là thông tin của đoạn [2,6].

Vậy, thao tác trả lời truy vấn tốn khoảng $O(\log_2(b-a+1))$.

4. Phân tích bộ nhớ:

Ta xét 2 trường hợp:

- $N = 2^k$: Cây IT đầy đủ, ở độ sâu cuối cùng có đúng 2^k lá, và các độ sâu thấp hơn không có nút lá nào (và các nút này đều có đúng 2 con). Như vậy: tầng k có 2^k nút, tầng $k-1$ có 2^{k-1} nút... Tổng các nút không quá 2^{k+1} nút.

- $2^k < N < 2^{k+1}$: Số nút của cây IT chắc chắn sẽ không quá số nút của cây IT có $N = 2^{k+1}$ → Tổng các nút không quá 2^{k+1} nút.

Do đó, số nút của cây cho dãy có N phần tử, với $N \leq 2^k$, là không quá 2^{k+1}, giá trị này xấp xỉ $4 \times N$. Bằng thực nghiệm cho thấy dùng mảng $4 \times N$ là đủ.

5. Một số bài tập cơ bản:

- *Đề:* Có N nhân viên, đến ngày phát lương thì lương của nhân viên thứ i là A_i. Nhưng nhân viên nhận nhiệm vụ phát lương là người mới, nên chưa có nhiều kinh nghiệm và khá hậu đậu. Anh ta không biết mức lương mình đã phát là bao nhiêu và đôi khi còn phát thừa hoặc thiếu lương nữa. Hãy giúp anh ta biết tổng số lương đã phát cho nhân viên từ x đến y là bao nhiêu và cập nhật lại nếu cần.

INP: Dòng đầu chứa 2 số N ($N \leq 50000$) và Q ($Q \leq 100000$). Dòng sau chứa N số lần lượt là $A_1, A_2, \ldots, A_N$. Q dòng tiếp theo, mỗi dòng chứa 3 số t, x, y. Nếu $t = 1$ thì in ra tổng lương của các nhân viên từ x đến y. Nếu $t = 2$ thì cập nhật lại lương của nhân viên x là y (tức $A_x = y$).

OUT: Nhiều dòng là kết quả của các truy vấn $t = 1$.

Sample:

INP:

5 3

1 2 3 4 5

1 1 4

2 5 6

1 2 5

OUT:

10

15

Hướng dẫn: Ban đầu với mảng A ta gọi thủ tục $Build(1,1,N)$ để dựng cây IT với IT_x là tổng $A_{l_x}, A_2, \dots, A_{r_x}$. Với truy vấn loại 1, ta gọi hàm $Query(x,y)$ để đưa ra kết quả. Với truy vấn loại 2, ta gọi $Update(x,y)$ để cập nhật lại giá trị của A_x và các nút trên cây IT.

```
procedure build(x,left,right:longint);
var middle:longint;
begin
   l[x]:=left; r[x]:=right;
   if left=right then
   begin
      leaf[left]:=x;
      IT[x]:=A[left]
   end
   else
   begin
      middle:=(left+right) div 2;
      build(2*x,left,middle);
      build(2*x+1,middle+1,right);
      IT[x]:=IT[2*x]+IT[2*x+1]
      // cập nhật tổng của đoạn [lx,rx] bằng tổng của
      // 2 đoạn con
   end
end;

procedure update(x,y:longint);
var k:longint;
begin
   k:=leaf[x]; IT[k]:=y; // IT[k]=A[x] nên ta cập nhật thẳng IT[k] luôn
   while k>1 do // cập nhật các nút cha
   begin
      k:=k div 2;
      IT[k]:=IT[2*k]+IT[2*k+1]
   end
end;
```

```
procedure update(x,y:longint);
var k:longint;
begin
   k:=leaf[x]; IT[k]:=y; // IT[k]=A[x] nên ta cập nhật thẳng IT[k] luôn
   while k>1 do // cập nhật các nút cha
   begin
      k:=k div 2;
      IT[k]:=IT[2*k]+IT[2*k+1]
   end
end;


function query(x,y:longint):longint;

   function request(k:longint):longint;
   begin
      if (l[k]>y) or (r[k]<x) then exit(0);
      // đưa về giá trị 0 để không ảnh hưởng đến tổng
      if (l[k]<=x) and (y<=r[k]) then exit(IT[k]);
      // vì k quản lý một đoạn thuộc [x,y] nên trả IT[k] tức là tổng của
      // đoạn đế tính tổng đoạn [x,y]
      exit(request(2*x)+request(2*x+1))
      // trong trường hợp còn lại, cây IT tính tổng 2 đoạn con cho đến
      // khi loại được đoạn không giao hoặc gặp đoạn con của [x,y]
   end;

begin query:=request(1) end;

{ chương trình chính }
begin
   input;
   build(1,1,N);
   for i:=1 to q do
   begin
      read(t,x,y);
      if t=1 then writeln(query(x,y))
      else update(x,y)
   end
end.
```

- *vn.spoj.com/problems/NKLINEUP/*

Hướng dẫn: Cây IT mỗi nút lưu 2 giá trị min và max là chiều cao min và chiều cao max trong đoạn quản lý.

Hàm $Query$ trong trường hợp $[a, b] \cap [l_x, r_x] = \emptyset$ thì trả về giá trị $min = \infty$ và $max = -\infty$ để lúc lấy min, max của kết quả không bị ảnh hưởng.

Bài này không có update nên không cần code thủ tục update và cũng không cần khai báo mảng $leaf$.

Code: http://ideone.com/C5kRWq

- *vn.spoj.com/problems/QMAX/*

Hướng dẫn: Đầu tiên cho nguyên mảng A bằng 0. Trong đoạn từ u đến v, mỗi phần tử tăng k đơn vị, vậy ta coi đó là k đoạn thẳng trải dài từ tọa độ u đến tọa độ v, sử dụng công thức lùa: $A[u] \coloneqq A[u] + k; A[v + 1] \coloneqq A[v + 1] - k$ để lùa rồi chạy từ 1 đến N để tính giá trị của mỗi phần tử.

Sau đó ta dựng cây IT với mỗi nút chứa giá trị max trong đoạn nó quản lý.

Code: http://ideone.com/a2AaIu