NCKU Programming Contest Training Course

2016/04/06

Chung-Chuan Wu

Department of Computer Science and Information Engineering
National Cheng Kung University
Tainan, Taiwan

made by electron & free999& a711186 & ioariroi

Outline

Segment Tree

made by electron & free999& a711186 & ioariroi

Segment Tree

Problem

給定一個數列$A_1 A_2 \ldots A_n$。並可能多次進行下列操作:

1. 求區間內數字最大
2. 對數列中某個數進行加減

陣列長度 = N

Query 數 = Q

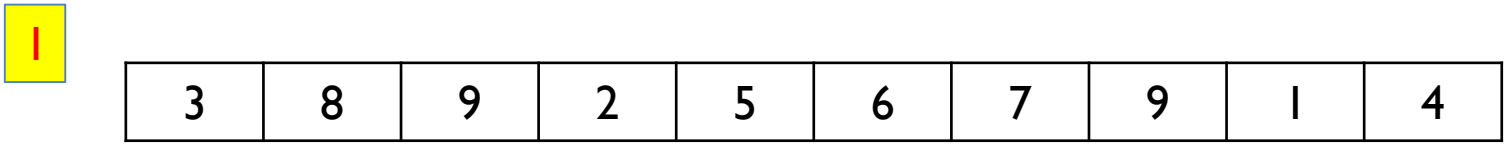
時間複雜度 = O (?)

3	8	9	2	5	6	7	9	1	4

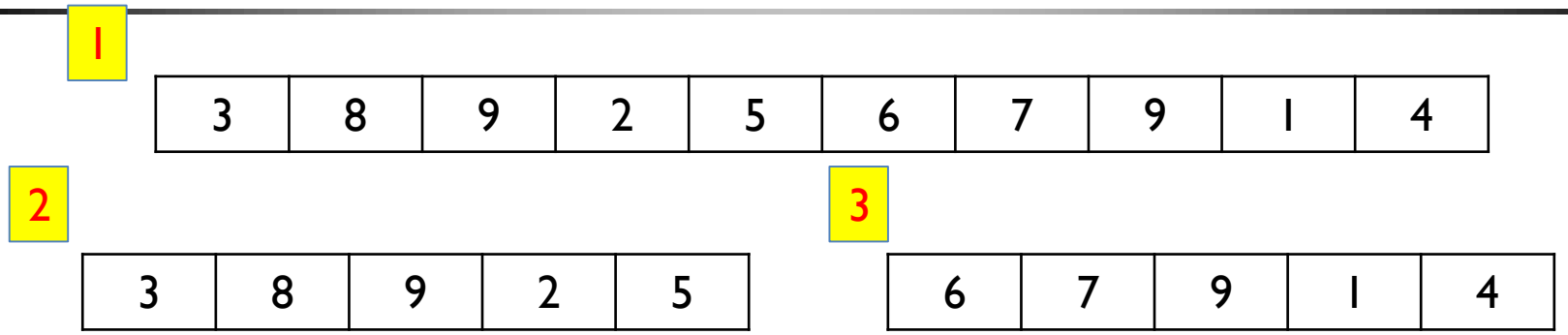
Segment Tree

- Naïve Solution
  - 陣列長度 = N, Query 數 = Q
  - 求區間總和，時間複雜度 O(QN)…
- Segment Tree
  - A tree-based data structure
  - 每個節點記載一個區間[L,R]的資訊，且都有兩個孩子，左節點記載[L, (*L*+*R*)/2]，右節點記載[(*L*+*R*)/2+1, R]
  - 線段樹節點的分支度不是0就是2, 因此若葉子節點數目為N,則線段樹總結點數目為2N-1
  - Construct tree with O(N)
  - RMQ (range minimum/maximum query problem) in O(logN)

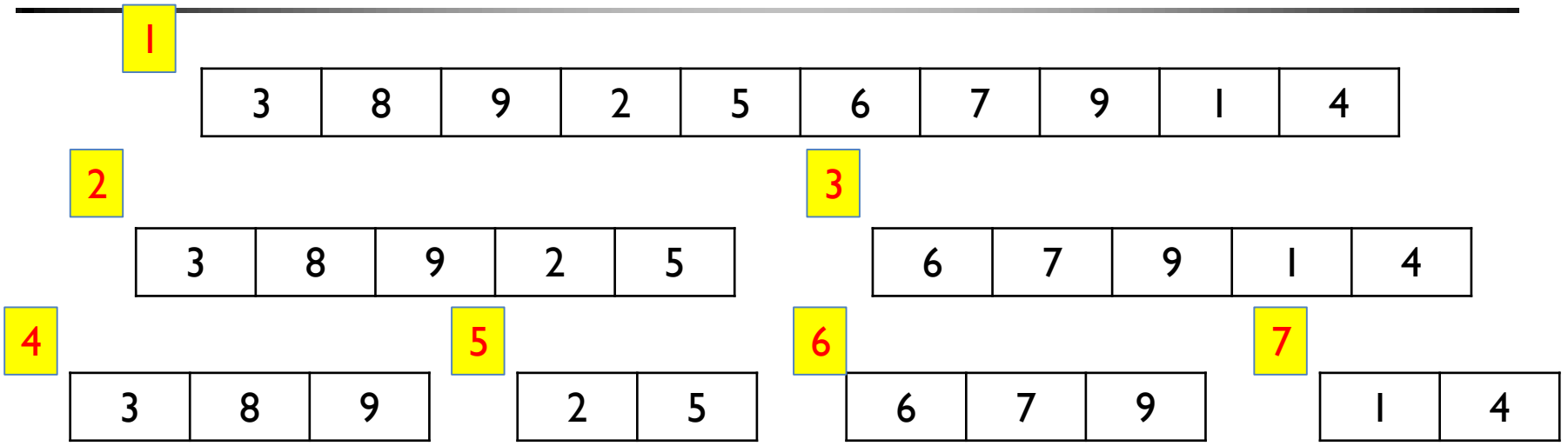
Segment Tree



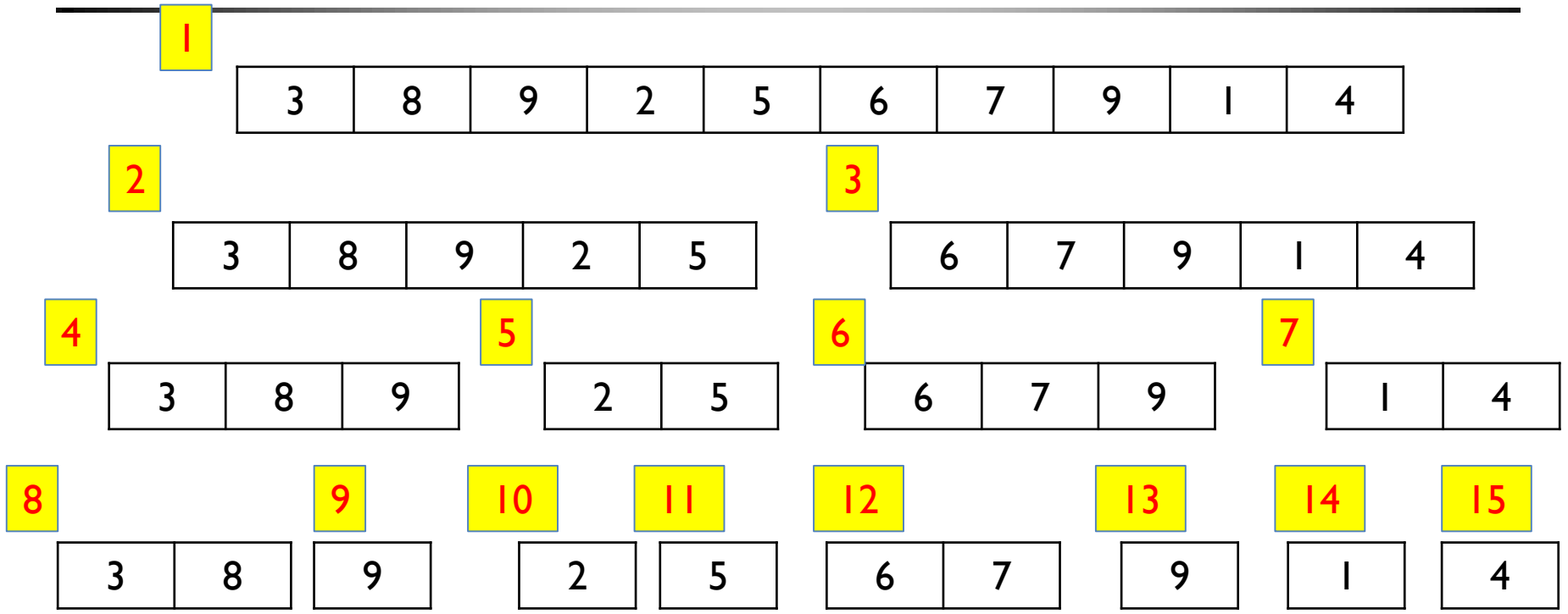
Segment Tree



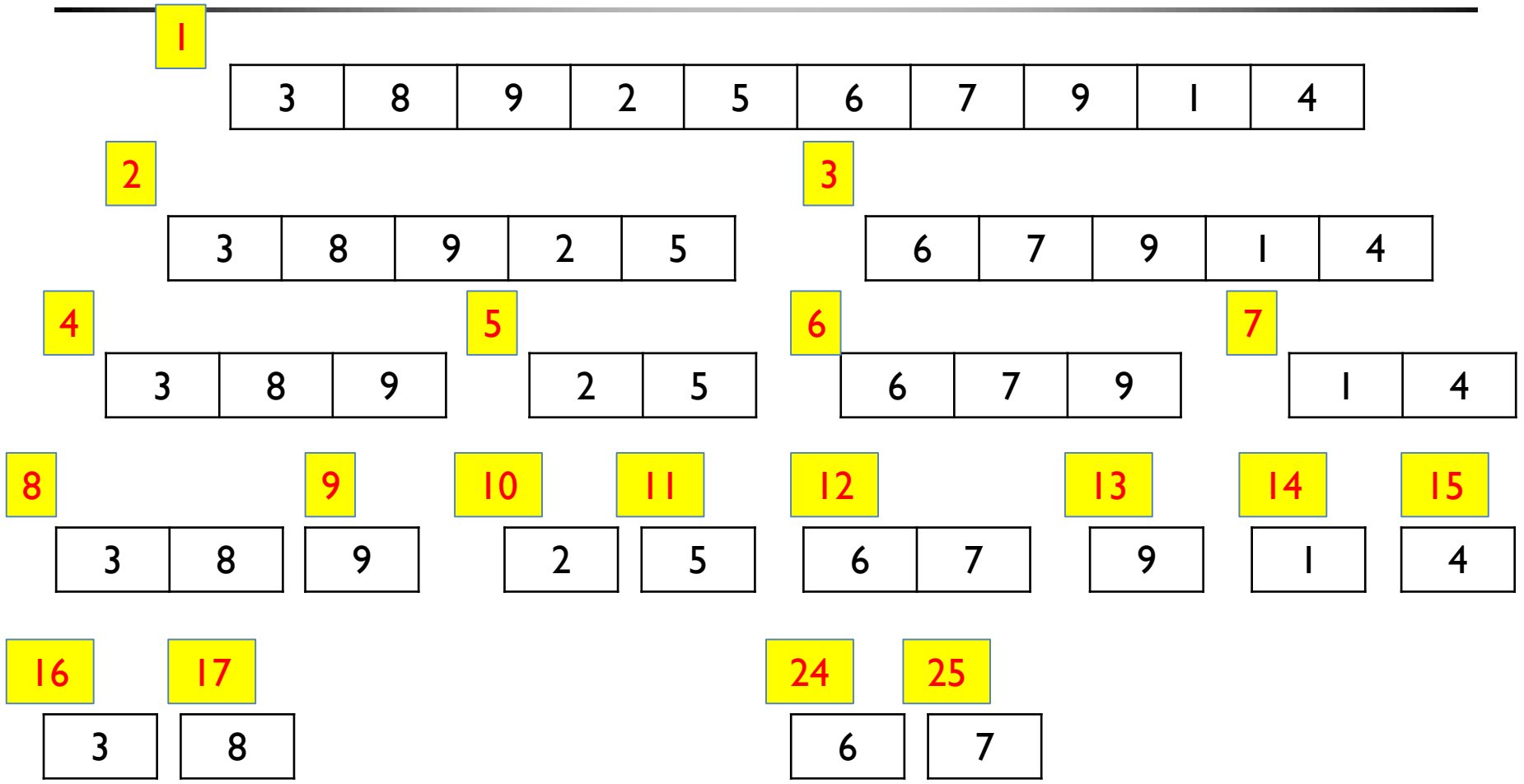
Segment Tree



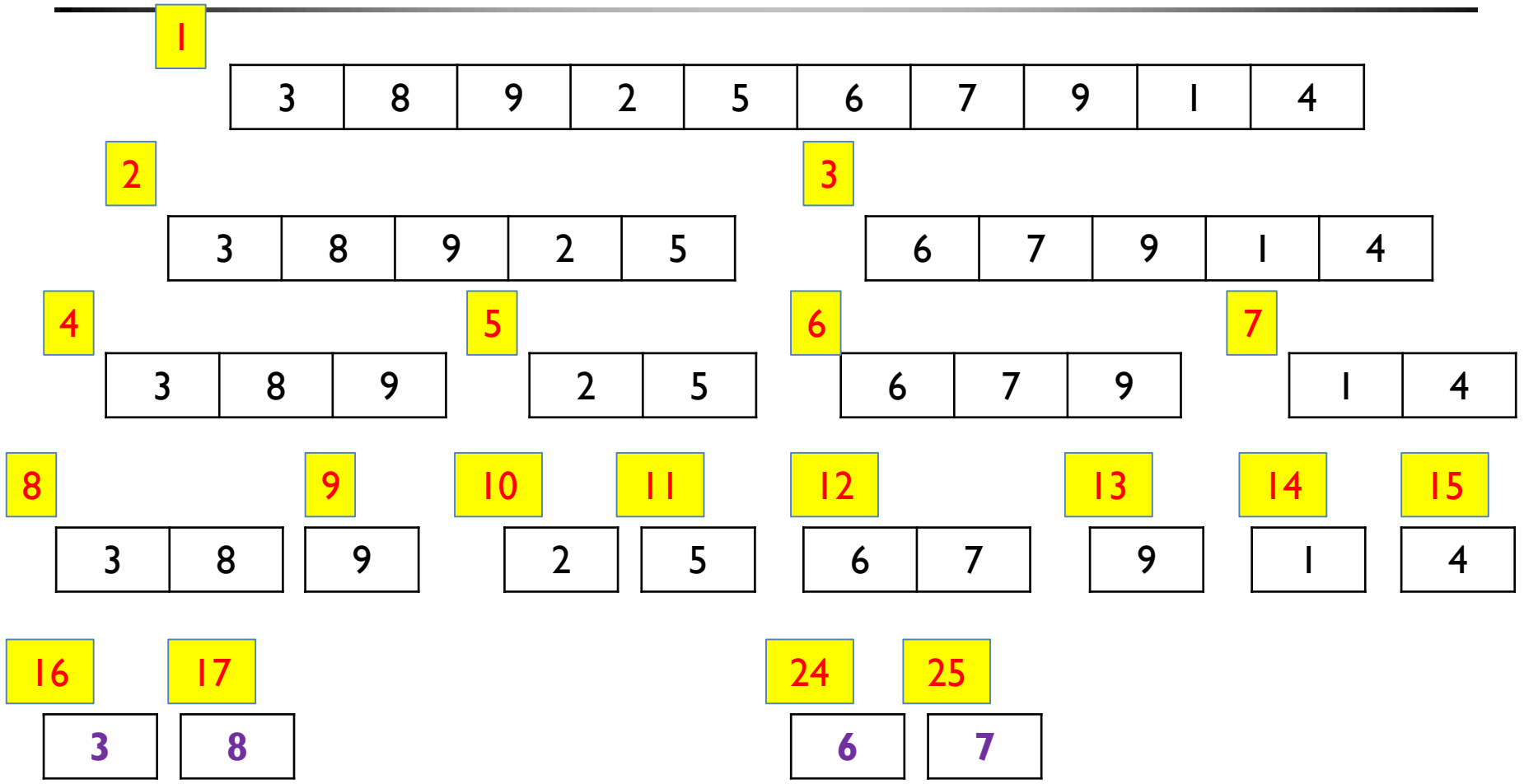
Segment Tree



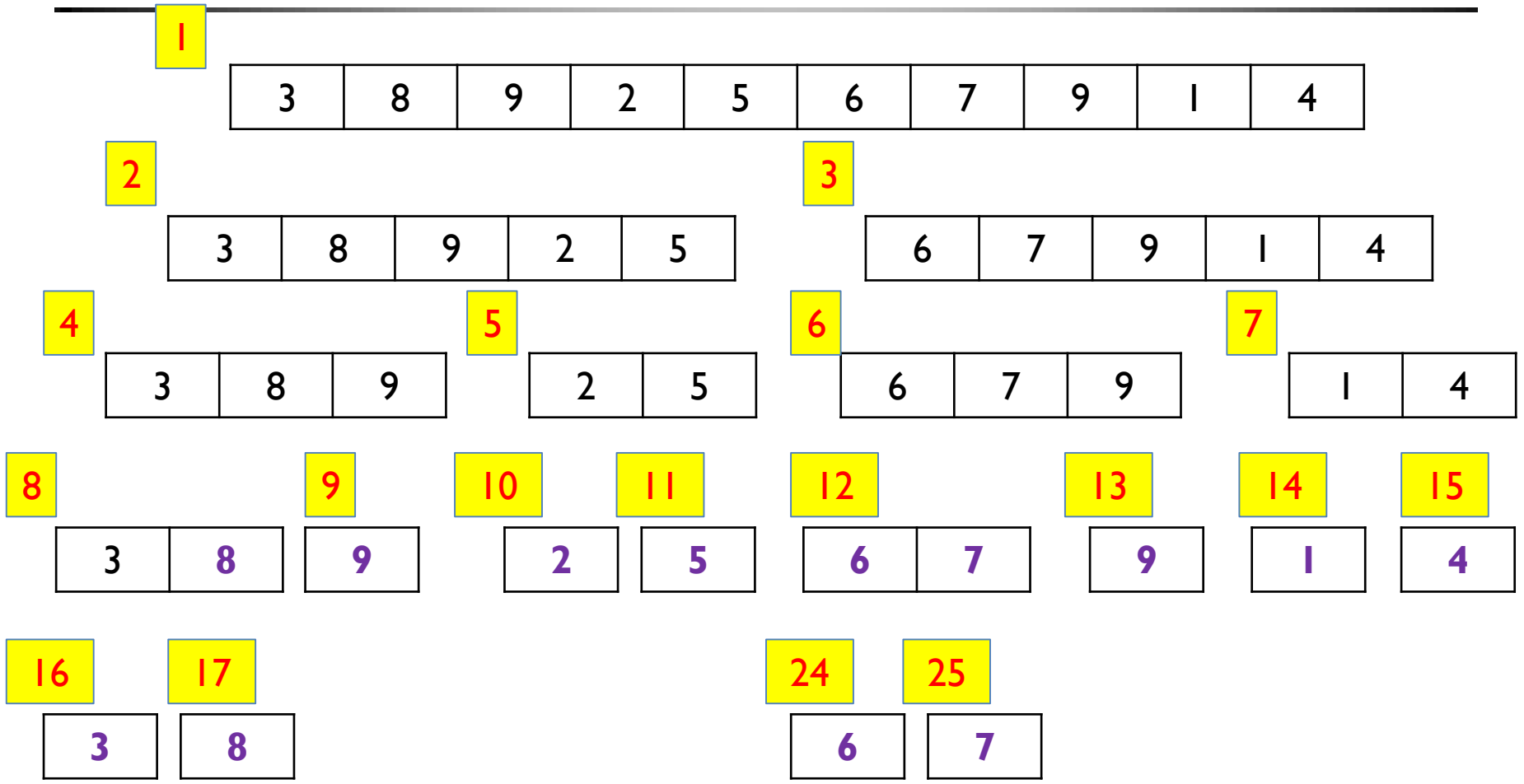
Segment Tree



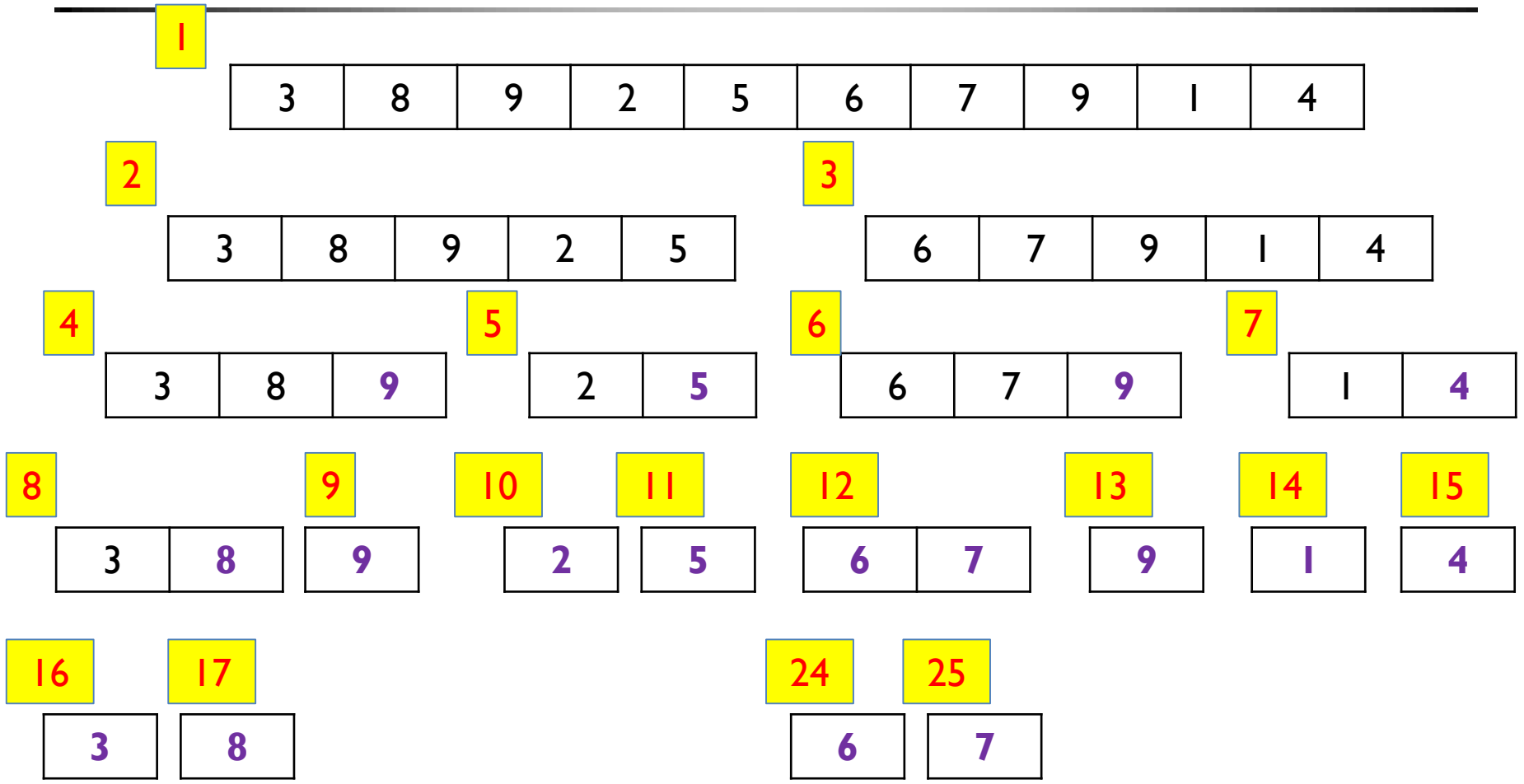
Segment Tree



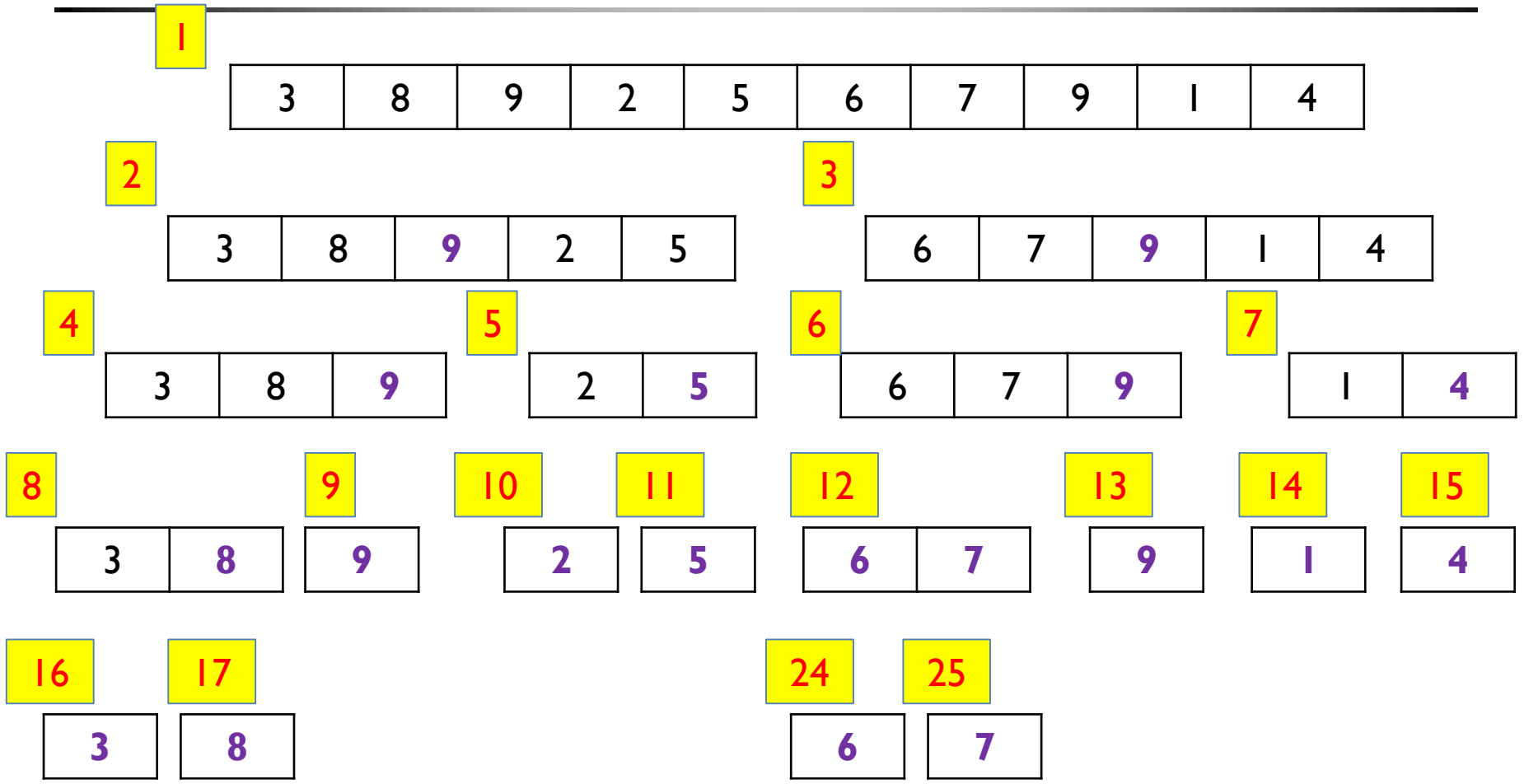
Segment Tree



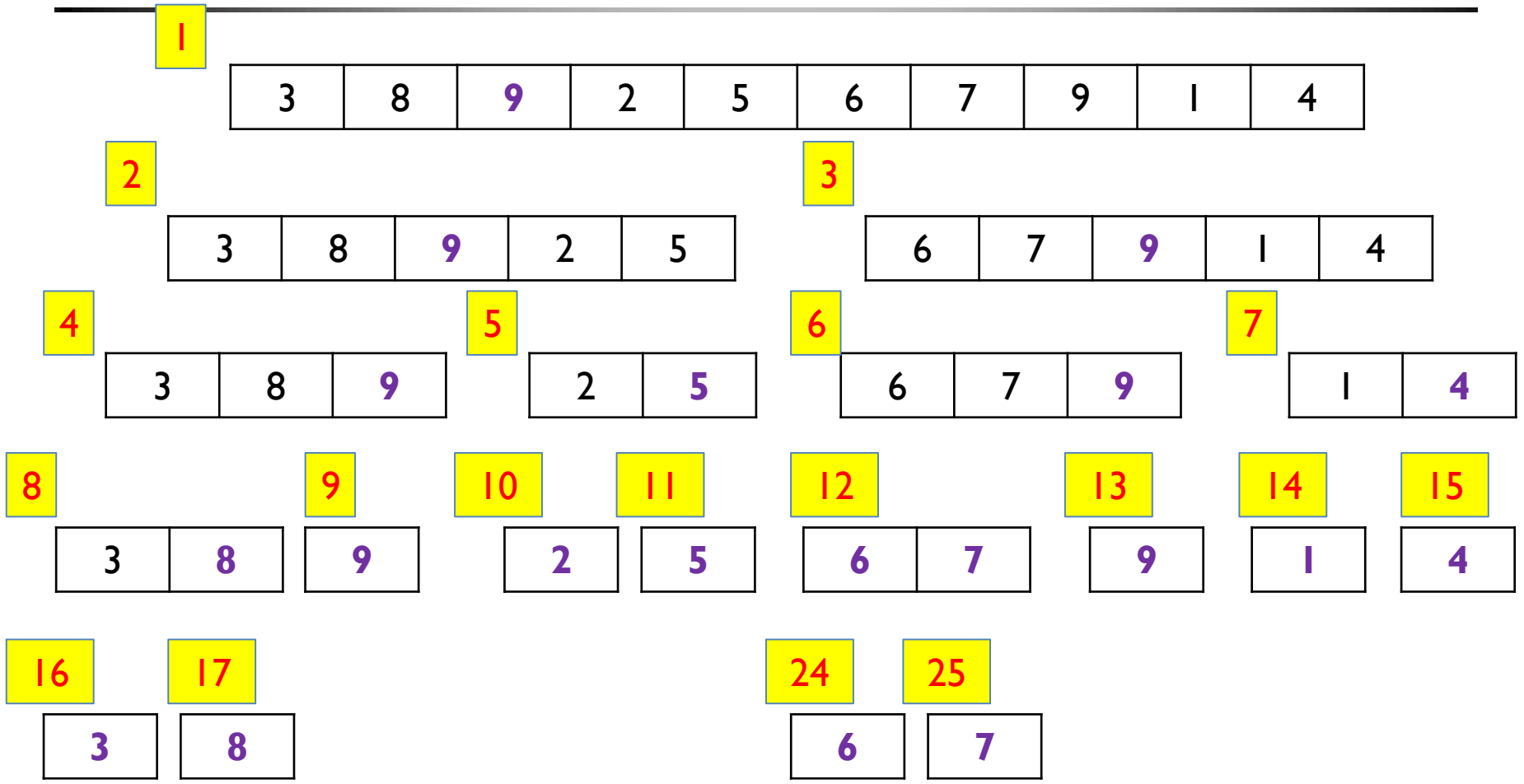
Segment Tree



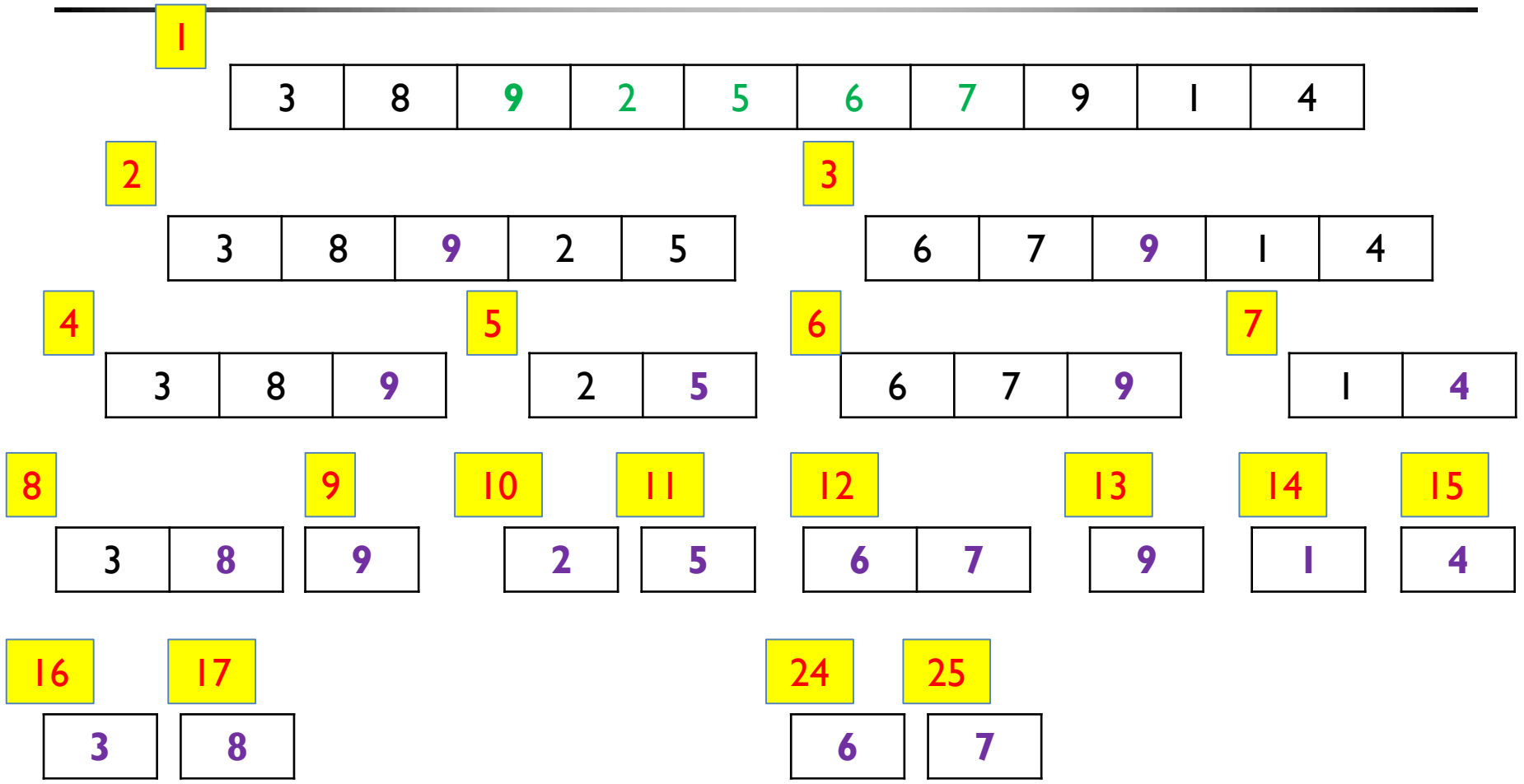
Segment Tree



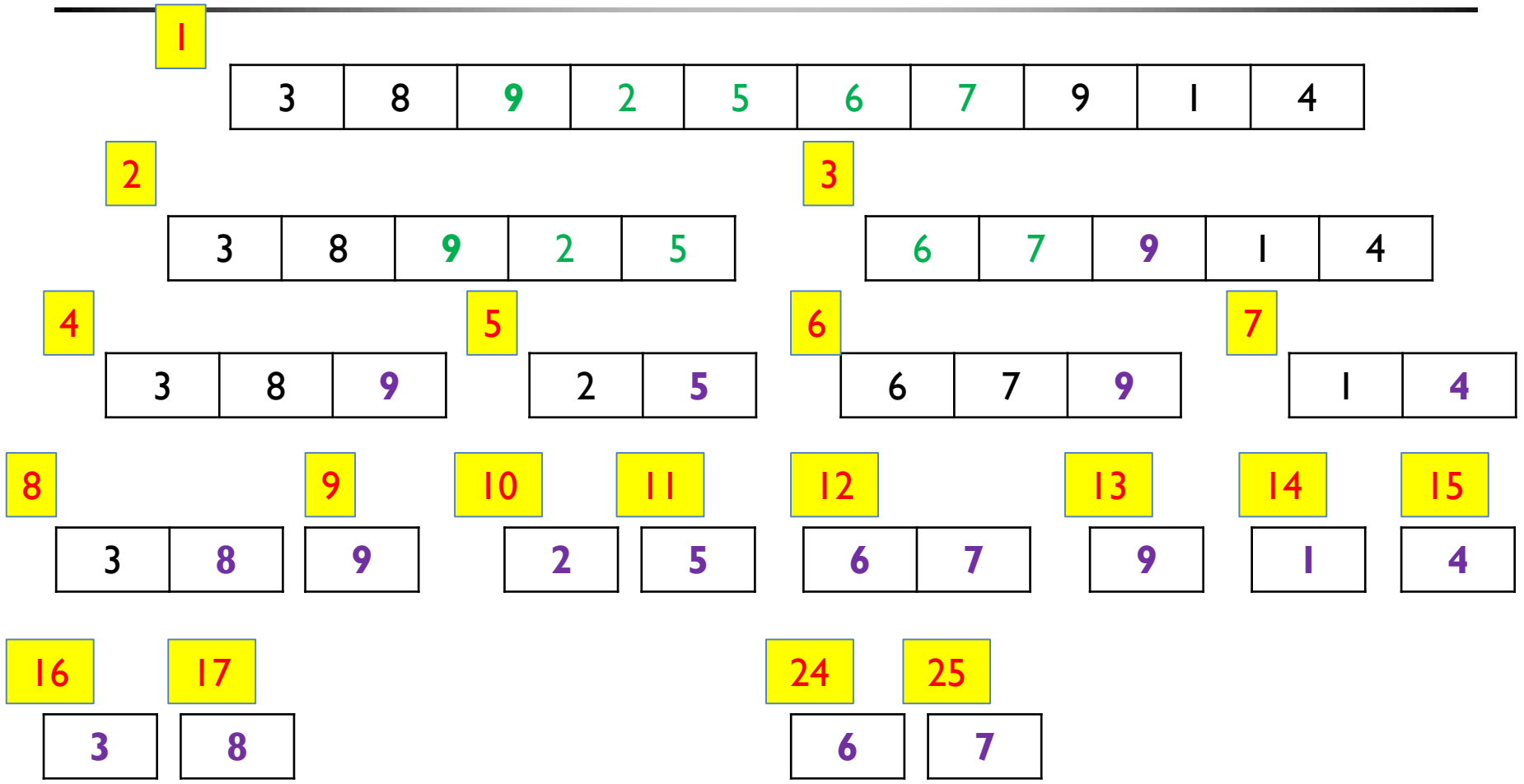
Segment Tree



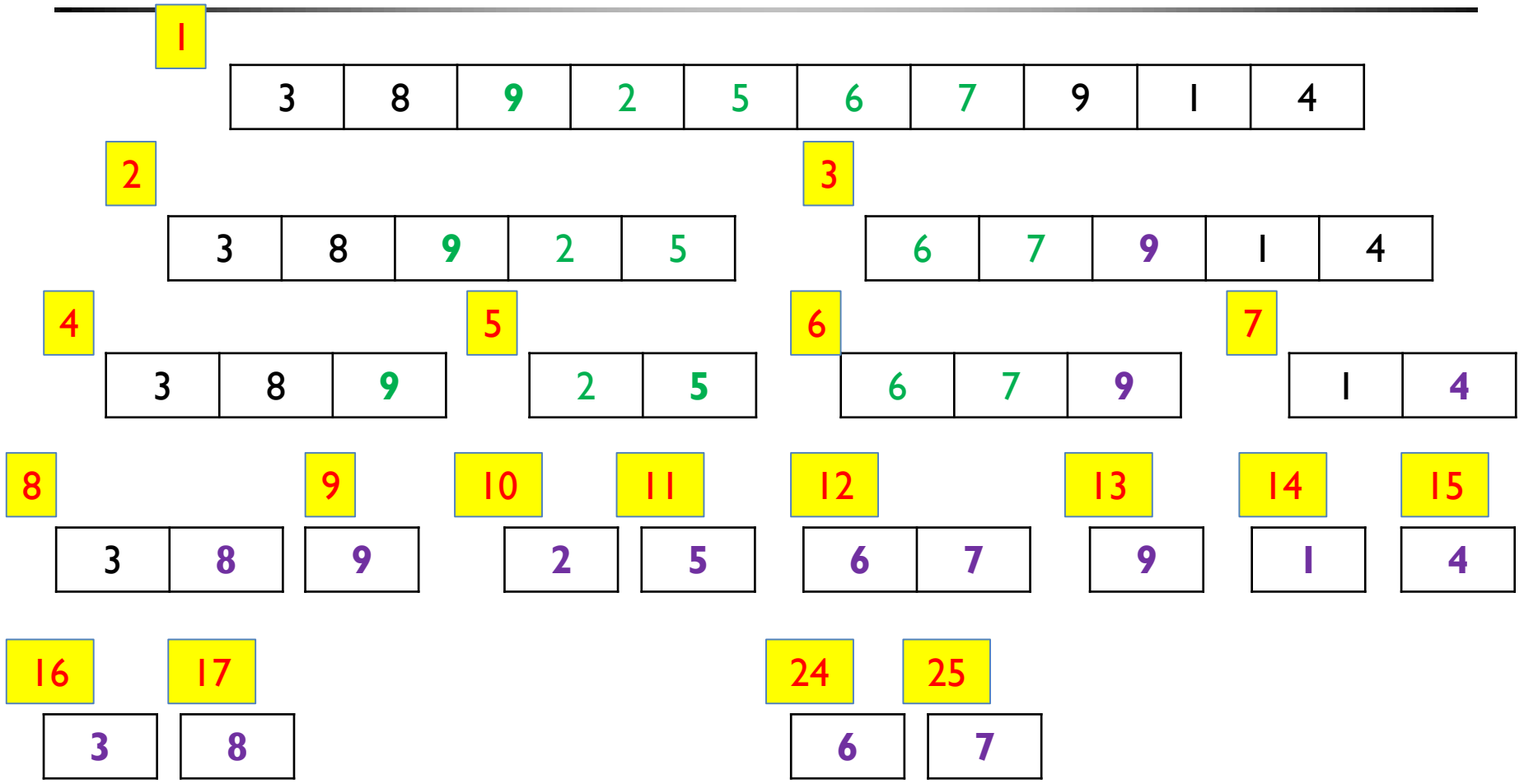
Segment Tree



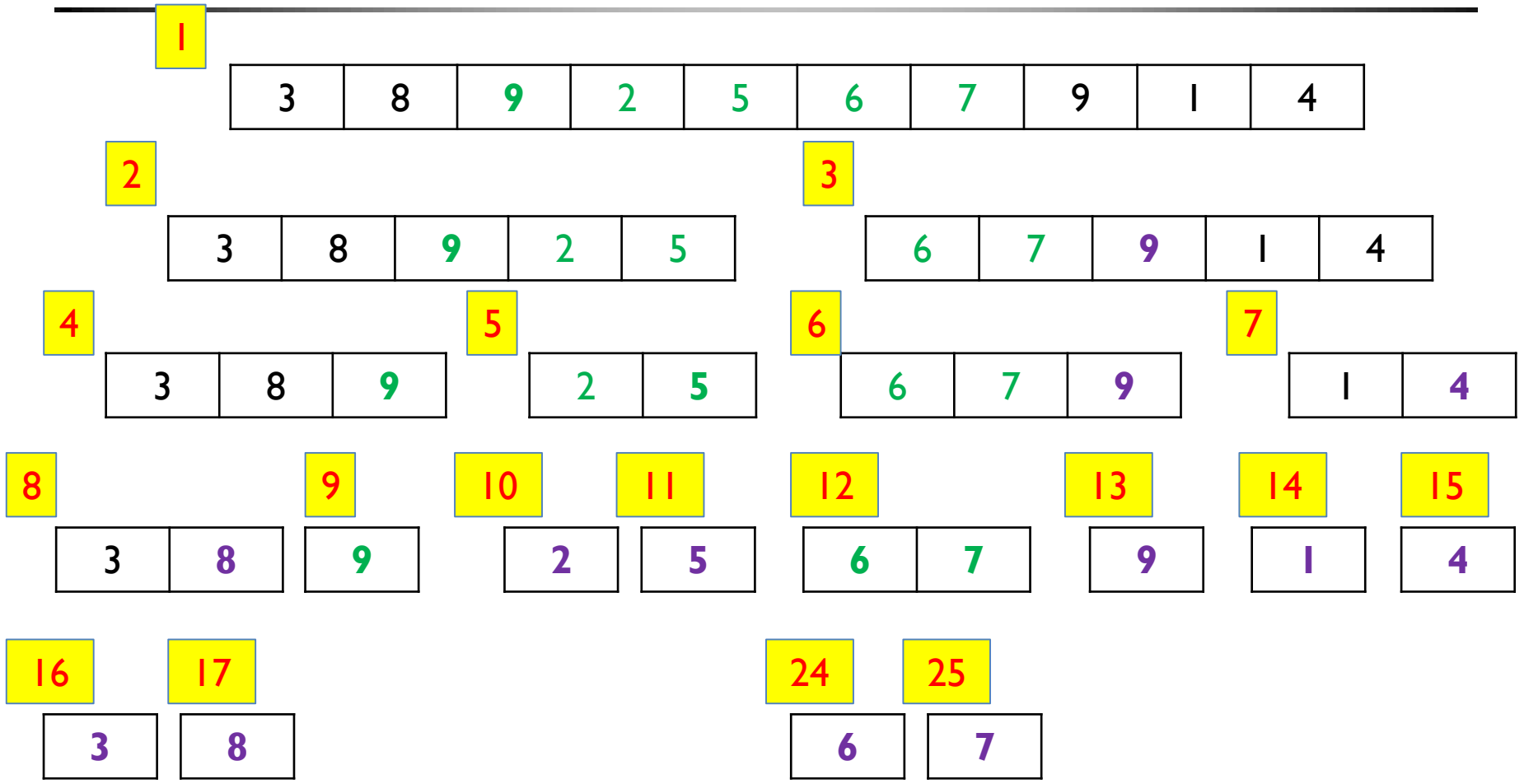
Segment Tree



Segment Tree



Segment Tree



Segment Tree

- 資料結構
  - 以找區間最大值為例

```
 8 struct Node{
 9     int l, r;
10     int max;
11     Node *left, *right;
12 };
```

- 實作
  - 建立build

    bottom up建立線段樹的初始狀態
  - 修改update

    修改線段樹，又可分為單點修改
    或區間修改(懶人標記)
  - 查詢query

    對線段樹查詢區間[l , r]

Segment Tree

- 建立Build

```
14 Node tree[MAXN << 1];
15 int num[MAXN];
16 int nNodeCnt = 0;
17 void build(Node *rt, int l ,int r){
18     rt->l = l;
19     rt->r = r;
20     if(l == r){
21         rt->max = num[l];
22         return ;
23     }
24
25     nNodeCnt++;
26     rt->left = tree + nNodeCnt;
27     nNodeCnt++;
28     rt->right = tree + nNodeCnt;
29
30     int m = (l + r) >> 1;
31     build(rt->left, l, m);
32     build(rt->right, m + 1, r);
33     rt->max = max(rt->left->max, rt->right->max);
34 }
```

Segment Tree

- 單點修改update

```
54 void update(Node *rt, int x, int v){
55     if(rt->l == rt->r){
56         rt->max += v;
57         return ;
58     }
59
60     int m = (rt->l + rt->r) >> 1;
61     if(x <= m){
62         update(rt->left, x, v);
63     }else{
64         update(rt->right, x, v);
65     }
66
67     rt->max = max(rt->left->max, rt->right->max);
68 }
```

Segment Tree

- 查詢query

```
36 int ans_max;
37 void query(Node *rt, int l, int r){
38     if(rt->l == l && rt->r == r){
39         ans_max = max(ans_max, rt->max);
40         return ;
41     }
42
43     int m = (rt->l + rt->r) >> 1;
44     if(r <= m){
45         query(rt->left, l, r);
46     }else if(l > m){
47         query(rt->right, l, r);
48     }else{
49         query(rt->left, l, m);
50         query(rt->right, m + 1, r);
51     }
52 }
```

Example POJ-3264

給定N $(1 \leq N \leq 50{,}000)$個數$S_1 S_2 \ldots S_Q$，多次求任一區間$S_A - S_B (1 \leq A \leq B \leq N)$中最大數和最小數的差。

本題樹節點的資料結構?

Example POJ-3264

本題樹節點的資料結構:

```
12 struct Node{
13     int l, r;
14     int min, max; // 當前區間的最大值和最小值
15     Node *left, *right;
16 };
```

Example POJ-3264

也可以不要左右節點指標，用一個陣列存放線段樹，其中根節點下標為1。假設線段樹上某節點下標為i，則其左子節點下標為i*2 , 右子節點下標為 i * 2 + 1 即(i << 1) 和(i << 1) + 1。

若用此種方式儲存線段樹，則陣列大小需4*N(N為數列大小)

證明:

http://scinart.github.io/acm/2014/03/19/acm-segment-tree-space-analysis/

Example POJ-3468

給定Q ($1 \leq Q \leq 100{,}000$)個數$A_1 A_2 \ldots A_Q$,，以及可能多次進行的兩個操作:

1. 對某個區間$A_i A_{i+1} \ldots A_j$的每個數都加c($-10000 \leq c \leq 10000$)
2. 求某個區間$A_i A_{i+1} \ldots A_j$的數的和

本題樹節點的資料結構?只存該區間的數的總和?

Example POJ-3468

如果只存和，會導致每次加數的時候都要更新到葉子節點，時間複雜度可能達到O(n)，為了避免這種情況，引入懶人修改(lazy propagation)。

本題樹節點結構：

```
12 struct Node{
13     int l, r;
14     Node *left, *right;
15     long long sum; // 當前區間數的和
16     long long inc; // 增加的值，藉此表示此區間是否被修改過
17                    // 以更新左右子樹
18 };
```

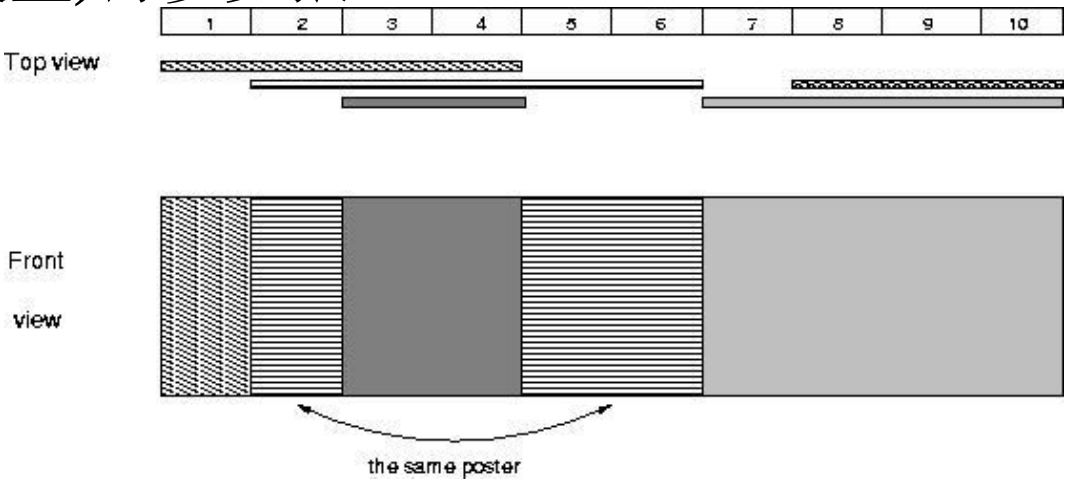
Example POJ-3468

- 區間修改(懶人標記)

```
45 void pushdown(Node *rt){
46     if(rt->inc){
47         rt->left->inc += rt->inc;
48         rt->right->inc += rt->inc;
49         rt->left->sum += (((rt->left->r - rt->left->l) + 1)
50                           * rt->inc);
51         rt->right->sum += (((rt->right->r - rt->right->l) + 1)
52                           * rt->inc);
53         rt->inc = 0;
54     }
55 }
```

Example POJ-2528

給定一些海報，可能相互重疊，告訴我們每個海報寬度(每張海報高度相同)和先後疊放次序，問可以被看見的海報(即沒有被完全覆蓋)有多少張。



海報最多10,000張，但是牆有10,000,000單位區間長。

Example POJ-2528

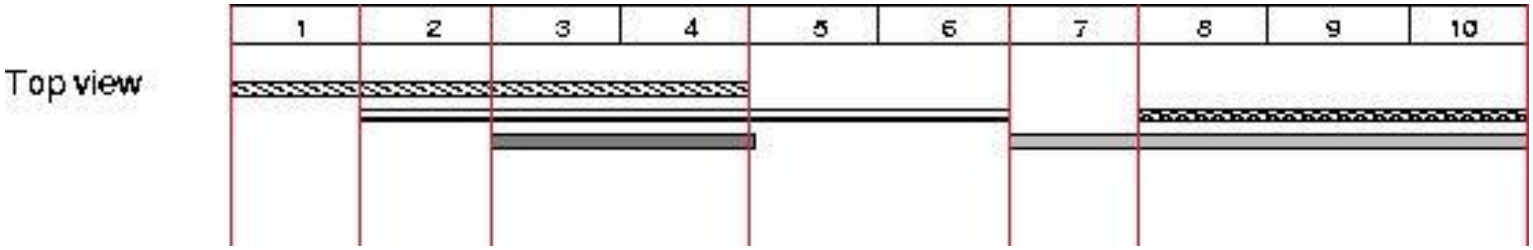
如果每個葉子節點都代表一單位區間，將會造成超過記憶體限制，也就是說單位區間的數目太多。

解決辦法:離散化

我們可以知道，由於最多10,000張海報，共計20,000個端點，這些端點把牆最多分成19,999個單位區間。

我們只要對這19,999個區間編號，然後建立線段樹。

Example POJ-2528



然而按上圖的離散化方法(直接對端點進行編號)，求每張海報覆蓋了哪些單位區間會造成錯誤。

舉個例子：依次貼[1,10], [1,4], [6,10] ， 離散化後1->1, 4->2, 6->3, 10->4，直接對端點進行離散化後變成[1,4],[1,2],[3,4]，那麼最終答案變成了2 ， 但正確答案是3。

解決辦法：在相鄰端點間距大於1時(即並不是緊鄰的端點)再添加一個二者中間的數當作沒有覆蓋到的區域。(這樣最多會有 20000 + 19999個單位區間)

Example POJ-2528

解題關鍵： 插入數據的順序由張貼順序最後的海報依序往前插入，這樣後插入的海報不可能覆蓋先插入的海報，因此插入一張海報時，如果發現海報對應區間有一部分或全部露出來，就說明了該海報部分或全部可以被看見。

本題樹節點的資料結構?這些資料應該如何進行更新呢?

Example POJ-2528

本題樹節點的資料結構：

```
24 struct Node{
25     int l, r;
26     bool covered; // 當前區間是否已經被海報覆蓋
27     Node *left, *right;
28 };
```

本題離散化:

```
86          sort(bounds.begin(), bounds.end());
87          bounds.erase(unique(bounds.begin()
88                              , bounds.end()), bounds.end());
89
90          int nInterval = 1, nCnt = bounds.size();
91          for(int i = 0; i < nCnt; ++i){
92              hash[bounds[i]] = nInterval;
93              if(i < nCnt - 1){
94                  if((bounds[i + 1] - bounds[i]) == 1){
95                      nInterval++;
96                  }else{
97                      nInterval += 2;
98                  }
99              }
100         }
```

Example POJ-1151

給定一些矩形的左上角座標和右下角座標(坐標是浮點數)，問這些矩形所覆蓋的面積是多大。

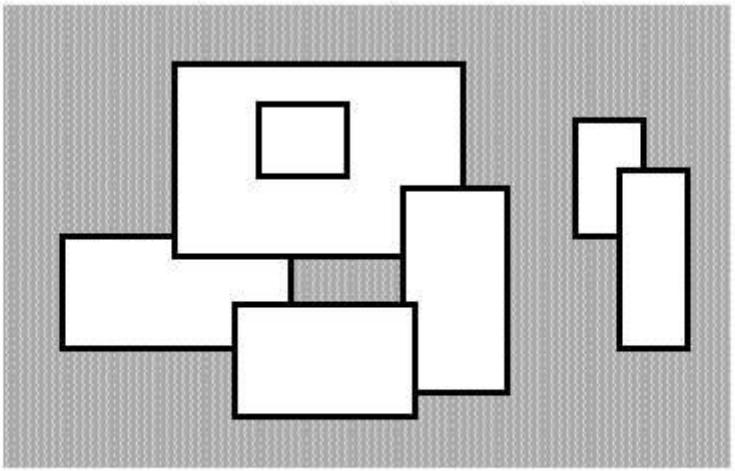


Figure 1. A set of 7 rectangles

Example POJ-1151

對縱軸(y軸)進行離散化。 n個矩形的2n個橫邊縱坐標共構成最多2n-1個區間的邊界，對這些區間編號，建立線段樹。

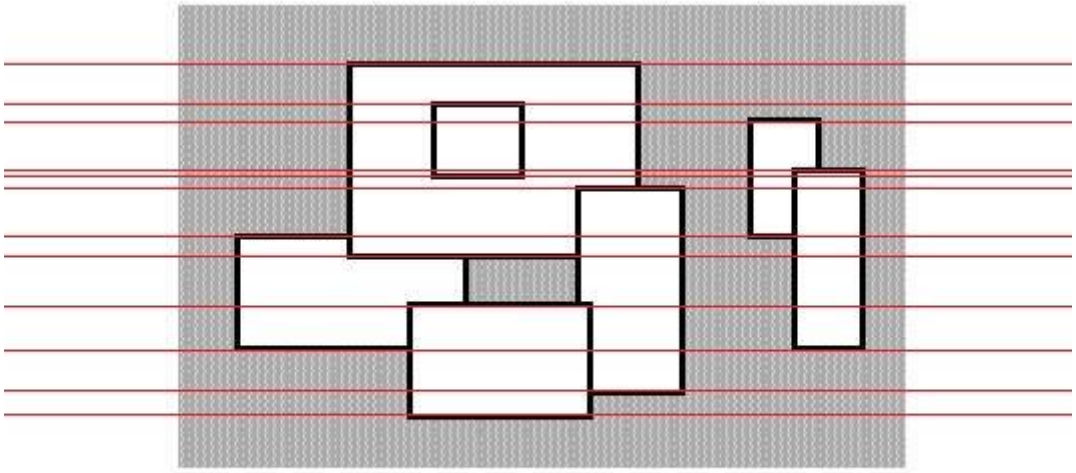


Figure 1. A set of 7 rectangles

Example POJ-1151

本題樹節點的資料結構?如何將一個個矩形插入線段樹?插入過程中這些資料如何更新?怎樣查詢?

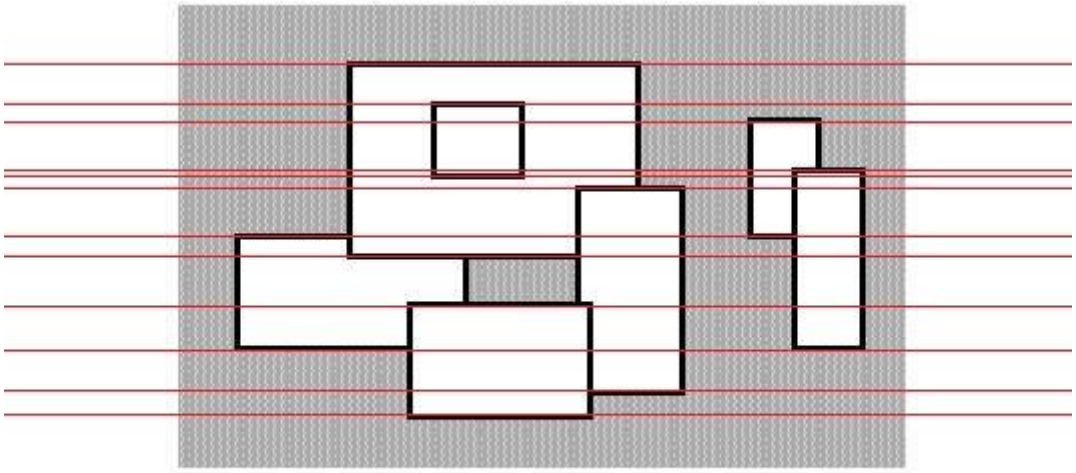


Figure 1. A set of 7 rectangles

Example POJ-1151

本題樹節點的資料結構：

```
15 struct Node{
16     int l, r;
17     Node *left, *right;
18     double Len; // 矩形於當前節點的區間中覆蓋的總長
19     int covers; // 當前節點的區間被多少矩形完全覆蓋
20 };
```

Example POJ-1151

用一條直線從左到右掃描，碰到一條矩形縱邊的時候，就計算該直線有多長被矩形覆蓋，以及被覆蓋部分(完全覆蓋當前節點的區間)是覆蓋了幾次。碰到矩形左邊，要增加被覆蓋的長度，碰到右邊，則要減少被覆蓋的長度。

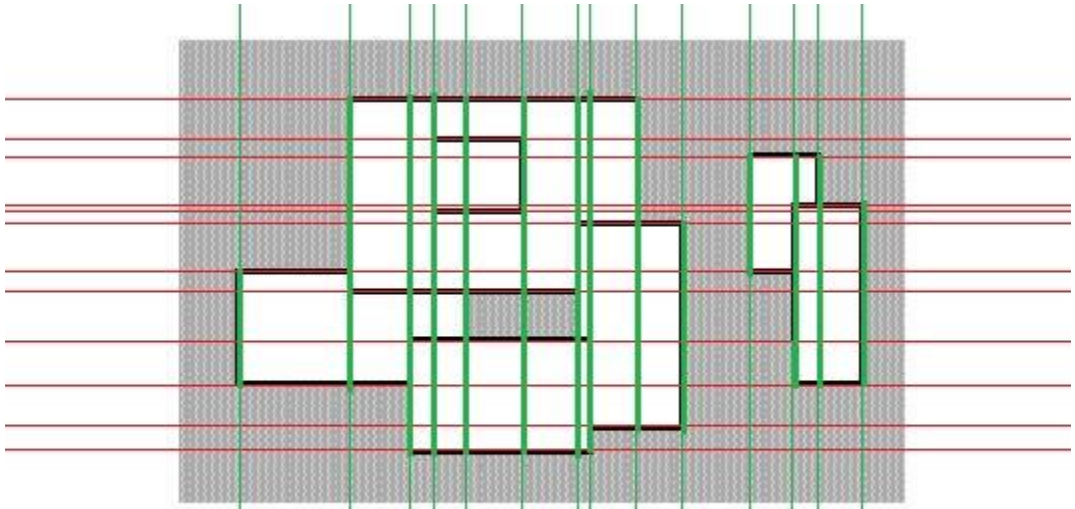


Figure 1. A set of 7 rectangles

Example POJ-1151

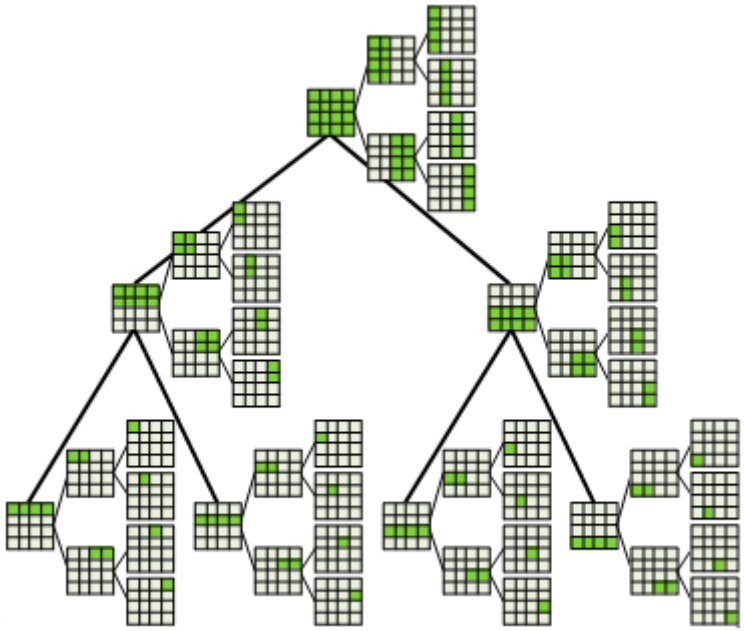
插入資料的順序：

將矩形的縱邊從左到右排序，然後依次將這些縱邊插入線段樹。要記住哪些縱邊是一個矩形的左邊(開始邊，或稱入邊)，哪些縱邊是一個矩形的右邊（結束邊，或稱出邊），以便插入時，對len和covers做不同的修改。插入一條邊後，就根據根節點的len 值增加總覆蓋面積的值。

增加的值為len * 當前的邊到下一條邊的距離。

2D Segment Tree

二維線段樹:先建立一維線段樹，接著在第一維區間的每個節點再建立一線段樹，樹節點只存放對應的區間（矩形）的數字之和。



Example POJ-1195

一個由數字構成的大矩陣，初始化為0，能多次進行兩種操作:

1. 對矩陣裡的某個數加上一個整數(可正可負)
2. 查詢某個子矩陣裡所有數字的和並輸出結果

Instruction	Parameters	Meaning
0	S	Initialize the matrix size to S * S containing all zeros. This instruction is given only once and it will be the first instruction.
1	X Y A	Add A to the number of active phones in table square (X, Y). A may be positive or negative.
2	L B R T	Query the current sum of numbers of active mobile phones in squares (X,Y), where L <= X <= R, B <= Y <= T
3		Terminate program. This instruction is given only once and it will be the last instruction.

Example POJ-1195

- 修改:於第一維區間中找到欲修改區間，再於當前區間的第二維區間找到欲修改的區間。

```
 9 void add_x(int rty, int rtx, int l, int r, int x, int delta){
10     tree[rty][rtx] += delta;
11     if(l == r){
12         return ;
13     }
14     int m = (l + r) >> 1;
15     if(x <= m){
16         add_x(rty, (rtx << 1) + 1, l, m, x, delta);
17     }else{
18         add_x(rty, (rtx << 1) + 2, m + 1, r, x, delta);
19     }
20 }
```

```
22 void add_y(int rty, int l, int r, int y, int x, int delta){
23     add_x(rty, 0, 1, s, x, delta);
24     if(l == r){
25         return ;
26     }
27     int m = (l + r) >> 1;
28     if(y <= m){
29         add_y((rty << 1) + 1, l, m, y, x, delta);
30     }else{
31         add_y((rty << 1) + 2, m + 1, r, y, x, delta);
32     }
33 }
```

Example POJ-1195

- 查詢:同修改

```
35 int query_x(int rty, int rtx, int l, int r, int x1, int x2){
36     if(l == x1 && r ==x2){
37         return tree[rty][rtx];
38     }
39
40     int m = (l + r) >> 1;
41     if(x2 <= m){
42         return query_x(rty, (rtx << 1) + 1, l, m, x1, x2);
43     }else if(x1 > m){
44         return query_x(rty, (rtx << 1) + 2, m + 1, r, x1, x2);
45     }else{
46         return query_x(rty, (rtx << 1) + 1, l, m, x1, m)
47             + query_x(rty, (rtx << 1) + 2, m + 1, r, m + 1, x2);
48     }
49 }
```

Example POJ-1195

- 查詢

```
51 int query_y(int rty, int l, int r, int y1, int y2, int x1, int x2){
52     if(l == y1 && r == y2){
53         return query_x(rty, 0, 1, s, x1, x2);
54     }
55
56     int m = (l + r) >> 1;
57     if(y2 <= m){
58         return query_y((rty << 1) + 1, l, m, y1, y2, x1, x2);
59     }else if(y1 > m){
60         return query_y((rty << 1) + 2, m + 1, r, y1, y2, x1, x2);
61     }else{
62         return query_y((rty << 1) + 1, l, m, y1, m, x1, x2)
63             + query_y((rty << 1) + 2, m + 1, r, m + 1, y2, x1, x2);
64     }
65 }
```

Example POJ-2155

Homework

- UVA
  - 11297, 11423, 11610, 12299, 12698, 11601
- POJ
  - 3264, 3468, 2528, 1151, 1195, 3321, 2155, 2352, 3067, 2481, 2299, 3368, 2528, 2828, 2777, 2886, 2750, 2482, 2352, 2155

- 基本門檻6題

Reference

- 演算法筆記-Sequence
  http://www.csie.ntnu.edu.tw/~u91029/Sequence.html#1
- 2015 IOI camp
  http://ioicamp.csie.org/content
- Segment Tree
  https://github.com/vo01github/Data_Structures/blob/master/Tree/Segment%20Tree/Segment%20Tree.md
- PKU Judge Online
  http://poj.org/