NCKU Programming Contest Training Course

Range Query

2017/05/17

Chia-Ying Lin

p78041099@mail.ncku.edu.tw

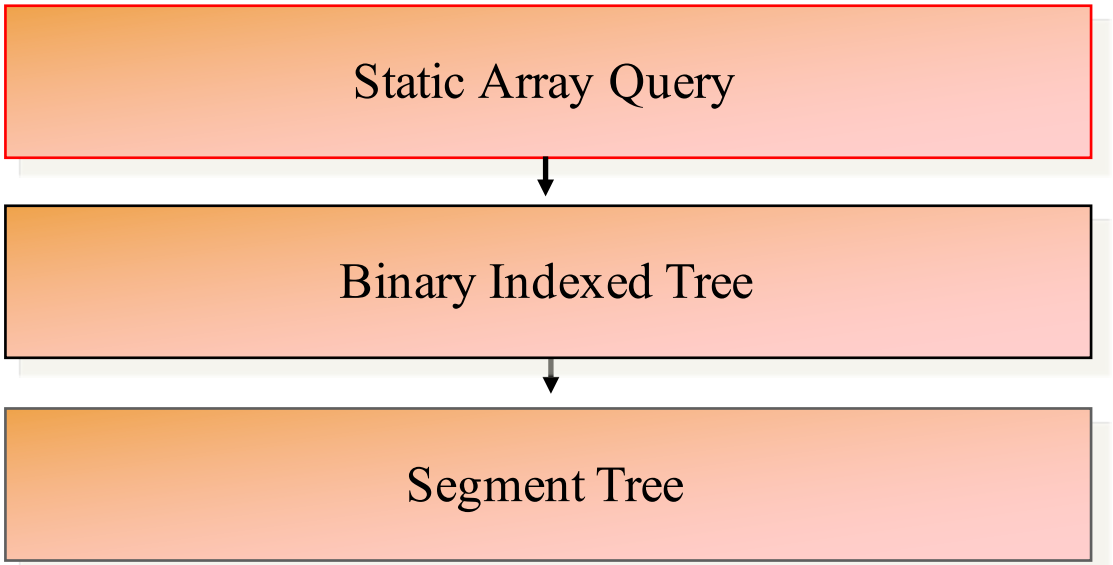
Department of Computer Science and Information Engineering
National Cheng Kung University
Tainan, Taiwan

made by electron & free999& a711186 & ioariroi

Sequence

- **Sequence**
  - In mathematics, a sequence is an enumerated collection of objects in which repetitions are allowed.
  - Ex. 9, 4, 8, 7
  - Data Structure (static, dynamic, advanced)
    - Array
    - List
    - Binary Search Tree
    - **Binary Indexed Tree**
    - **Segment Tree**
    - Sparse Table
    - Cartesian Tree
- **Range Queries**
  - Sum query, minimum query, maximum query

Outline



made by electron & free999& a711186 & ioariro

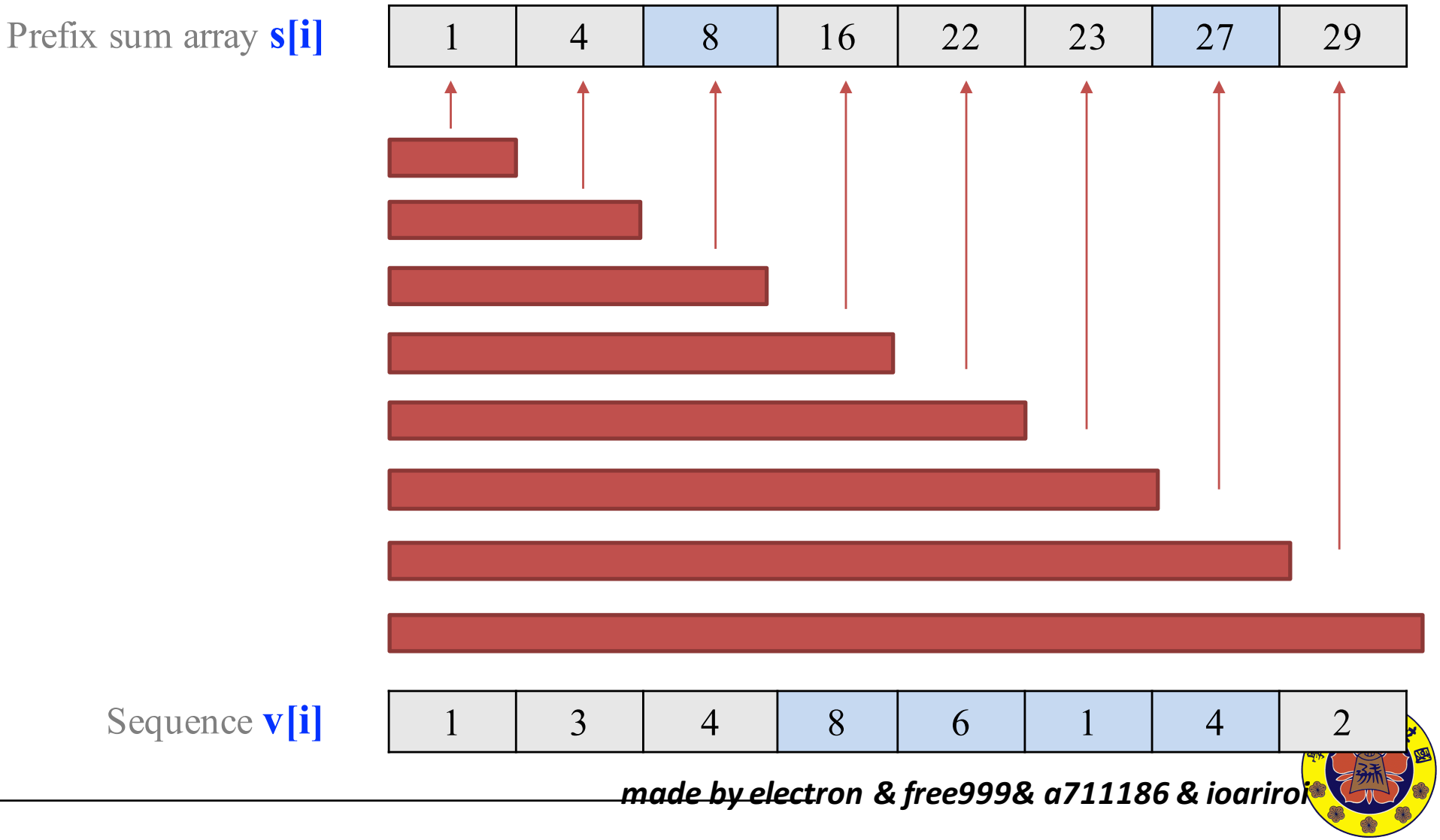
Static Array Query

- prefix sum array

Index	0	1	2	3	4	5	6	7
Sequence **v[i]**	1	3	4	8	6	1	4	2
Prefix sum array **s[i]**	1	4	8	16	22	23	27	29

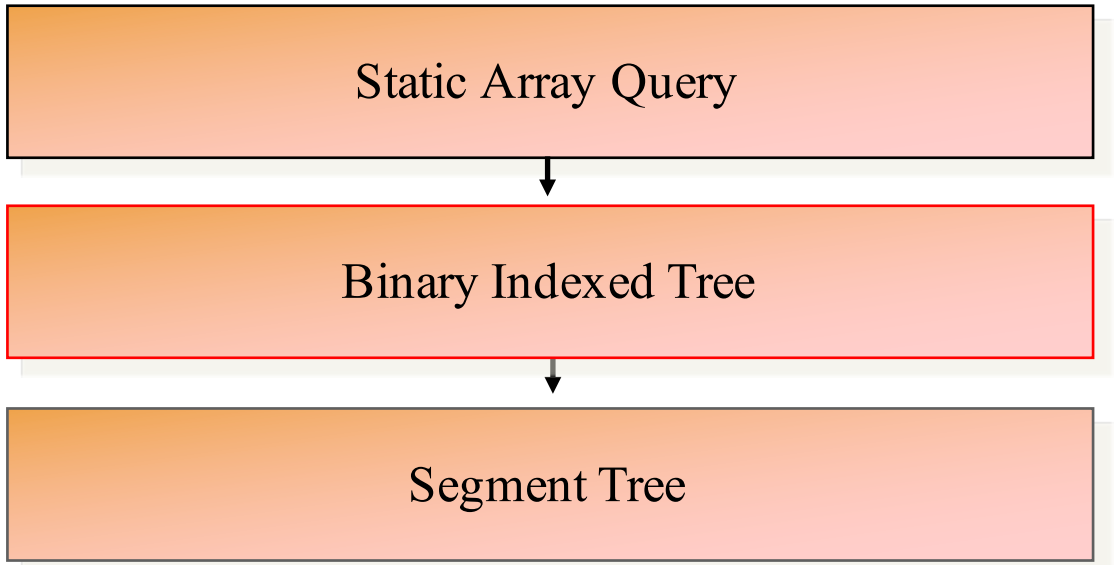
- $sum(a, b) = sum(0, b) - sum(0, a - 1) = s[b] - s[a-1]$
  - $Sum(3, 6) = v[3] + v[4] + v[5] + v[6] = s[6] - s[3-1] = s[6] - s[2] = 19$

Static Array Query



made by electron & free999& a711186 & ioariro

Outline



Binary Indexed Tree

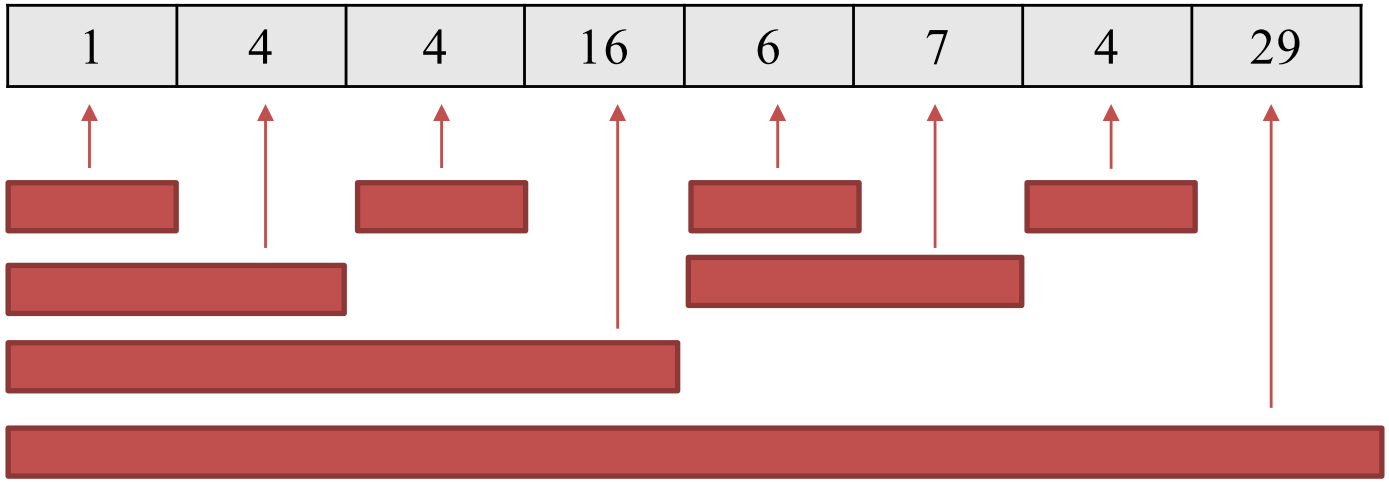
- A **binary indexed tree** or a **Fenwick tree** can be seen as a dynamic version of a prefix sum array.

- **Operations**
  - calculating the sum of elements in a range => $O(\log n)$
  - modifying the value of an element => $O(\log n)$

- **limitation**
  - Binary indexed tree can't insert or delete the nodes.

Binary Indexed Tree

S[i]:
the sum of elements in the range [i − k + 1 , i]
(*k* is the largest power of two that divides *i*)



Sequence **v[i]**

1	3	4	8	6	1	4	2

index	1	2	3	4	5	6	7	8
bit	0001 (0)	0010 (1)	0011 (0)	0100 (2)	0101 (0)	0110 (1)	0111 (0)	1000 (3)
k	1	2	1	4	1	2	1	8
range	[1, 1]	[1, 2]	[3, 3]	[2, 4]	[5, 5]	[5, 6]	[7, 7]	[1, 8]

Binary Indexed Tree

- lowbit (k)

```
int lowbit(int in)
{
    return in&(-in);
}
```

ex:
lowbit(1) = 1 [0001]
lowbit(2) = 2 [0010]
lowbit(3) = 1 [0011]
lowbit(4) = 4 [0100]

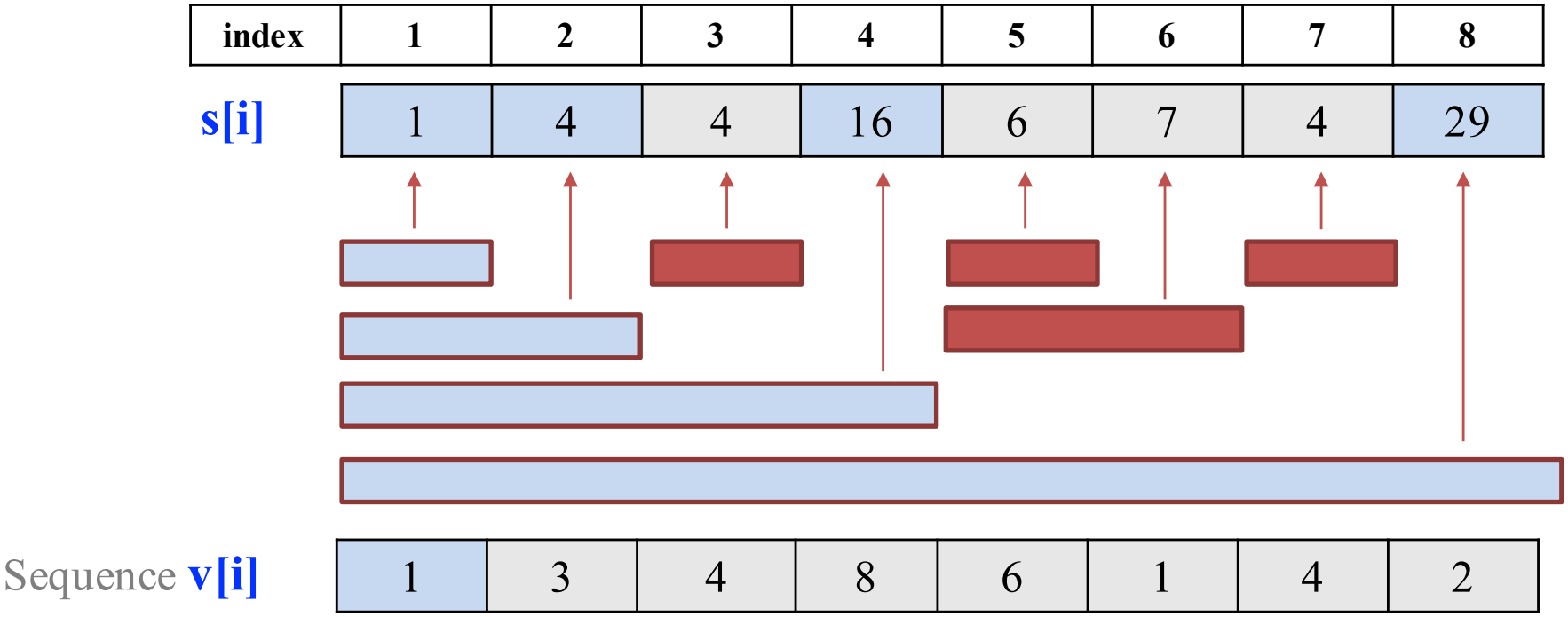
	十進位	最高位元表示法	1的補數表示法		2的補數表示法	
正數	128	無	無		無	
	127	01111111	01111111		01111111	
	126	01111110	01111110		01111110	
	:	:	:		:	
	2	00000010	00000010		00000010	
	1	00000001	00000001		00000001	
	0	00000000	00000000		00000000	
負數	0	10000000	11111111		00000000	
	-1	10000001	11111110		11111111	
	-2	10000010	11111101		11111110	
	:	:	:		:	
	-126	11111110	10000001		10000010	
	-127	11111111	10000000		10000001	
	-128	無	無		10000000	

(互為1的補數)　(互為2的補數)

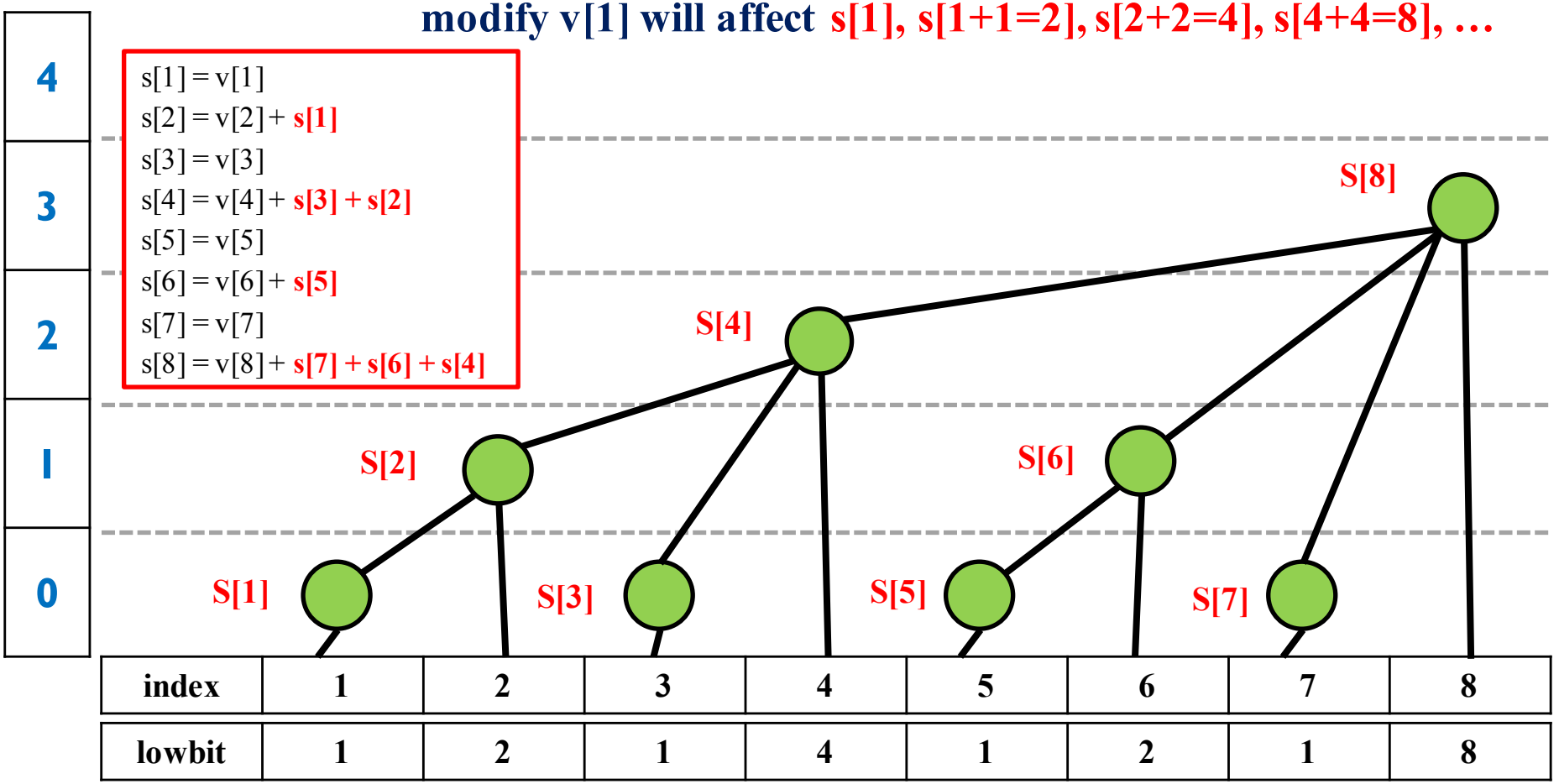
index	1	2	3	4	5	6	7	8
lowbit	1	2	1	4	1	2	1	8

Binary Indexed Tree

- modify



Binary Indexed Tree



index	1	2	3	4	5	6	7	8
lowbit	1	2	1	4	1	2	1	8

Binary Indexed Tree

- modify

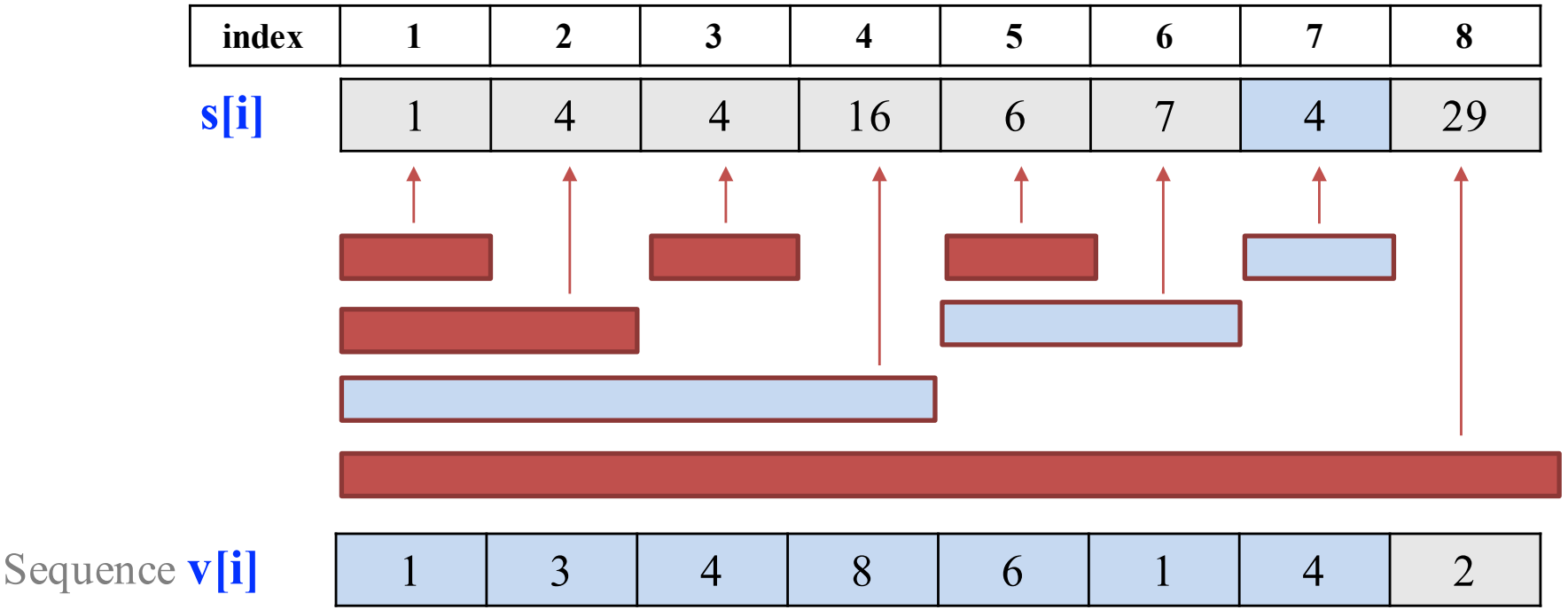
```
void modify(int index,int delta)
{
    for(int i=index;i<MAXX;i+=lowbit(i))
        s[i]+=delta;
}
```

Notice: index starts from 1

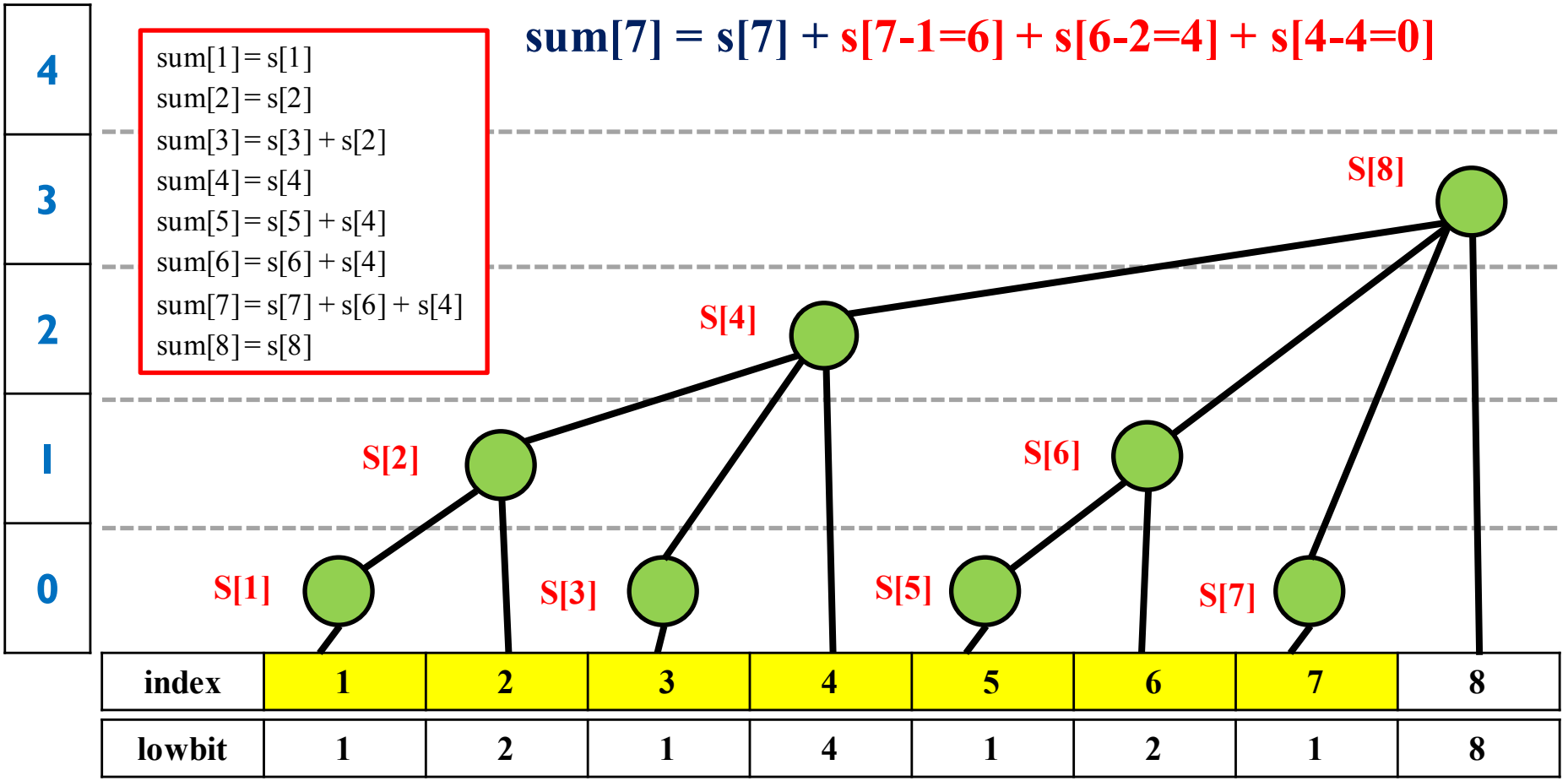
index= 0 ， 0 + Lowbit(0) = 0 infinite loop ！ ！ ！

Binary Indexed Tree

- getSum



Binary Indexed Tree



Binary Indexed Tree

- getSum

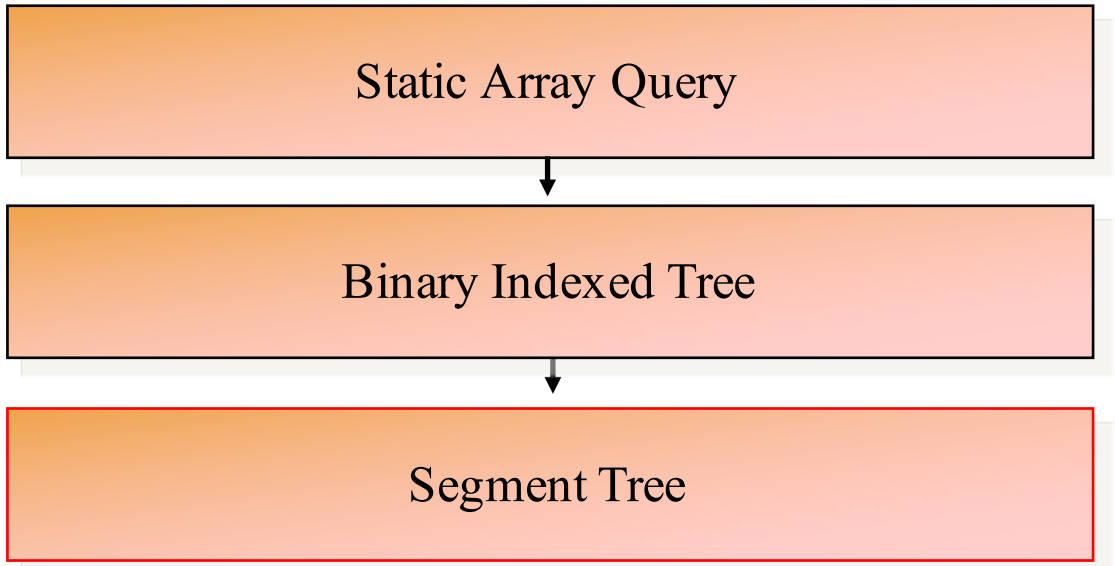
```
int getsum(int index)
{
    int ans=0;
    while(index>0)
    {
        ans+=s[index];
        index-=lowbit(index);
    }
    return ans;
}
```

- How to find the summation between interval [a, b] ?
  - call the subroutine **"getsum[b] – getsum[a-1]"**

Practice

POJ 2352: Stars

Outline



made by electron & free999& a711186 & ioariro

Segment Tree

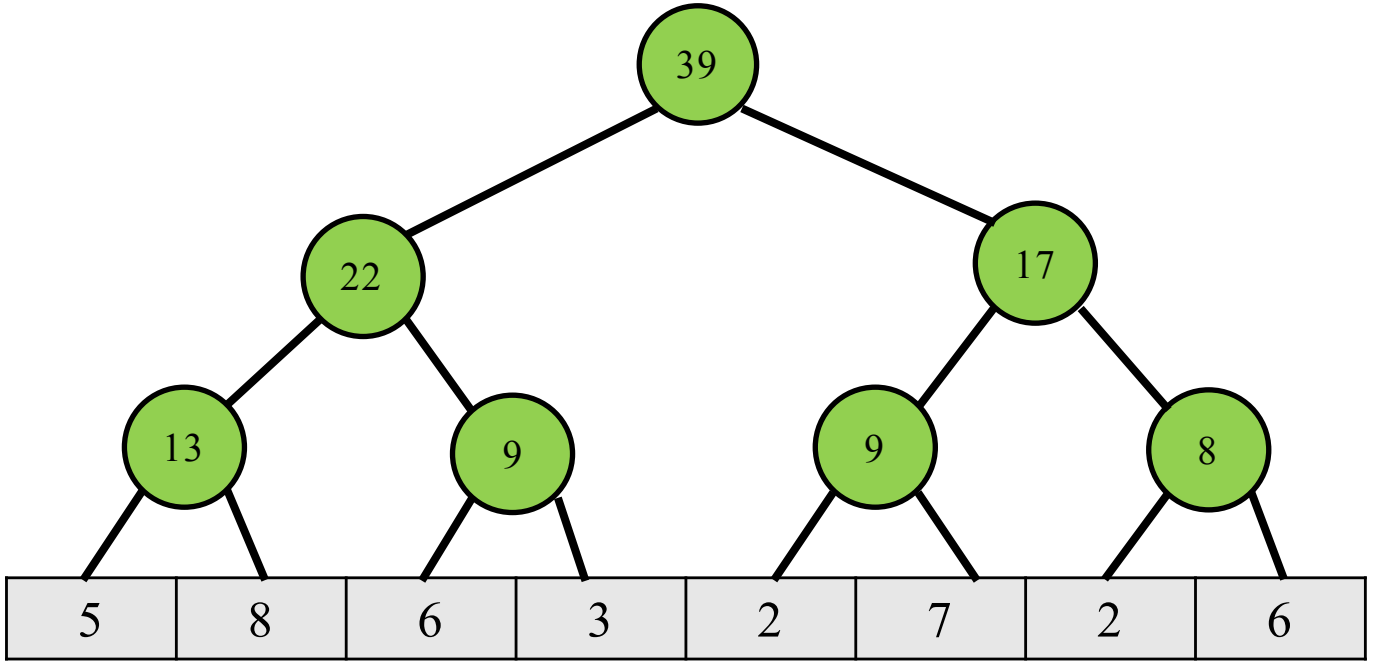
- A **segment tree** is a **binary tree** such that the nodes on the bottom level of the tree correspond to the array elements, and the other nodes contain information needed for processing range queries.

- **Operations**
  - calculating the sum, minimum, maximum of elements in a range
    => $O(\log n)$
  - inserting, deleting and modifying the value of an element
    => $O(\log n)$

- **Limitation**
  - A segment tree requires more memory and is a bit more difficult to implement.

Segment Tree

- Structure
  - 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)



Segment Tree

- Structure
  - Ex. find the minimum and maximum value in a range

```
struct Node
{
    int valmax,valmin;
    Node *l,*r;

    void update(int v)
    {
        valmin=v;
        valmax=v;
    }
    void pull()
    {
        valmax=max(l->valmax,r->valmax);
        valmin=min(l->valmin,r->valmin);
    }
};
```

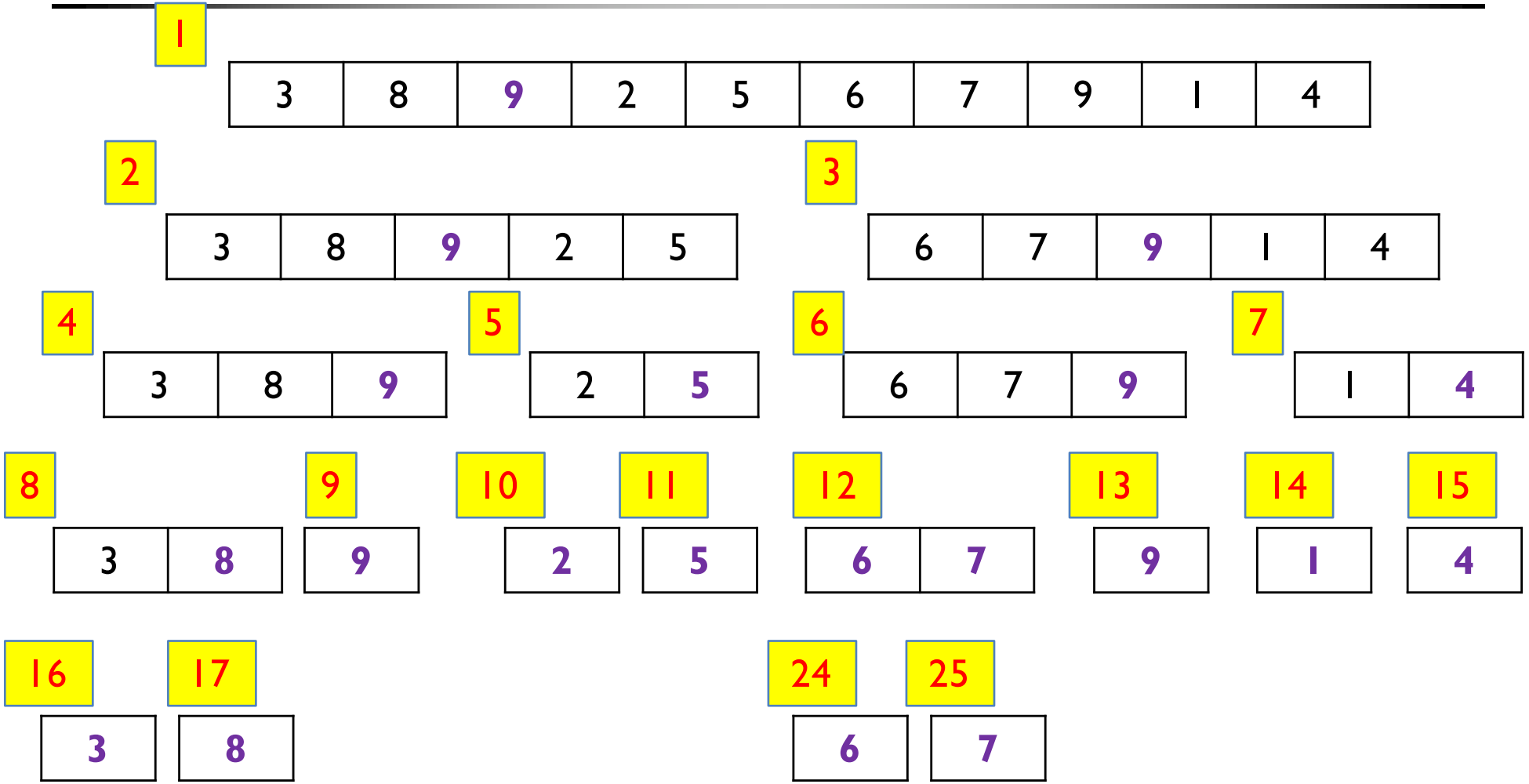
Segment Tree

- Build Segment tree
  - build the initial state

- Implement
  1) Is this node a leaf
  2) Create left subtree
  3) Create right subtree
  4) pull

```
Node* build(int L,int R)
{
    // create new node
    Node* now=new Node();
    // (1) is this node a leaf
    if(L==R)
    {
        now->update(s[L]);
        return now;
    }
    int mid=(L+R)>>1;
    // (2) create left subtree
    now->l=build(L,mid);
    // (3) create right subtree
    now->r=build(mid+1,R);
    // (4) pull
    now->pull();
    return now;
}
```

Segment Tree

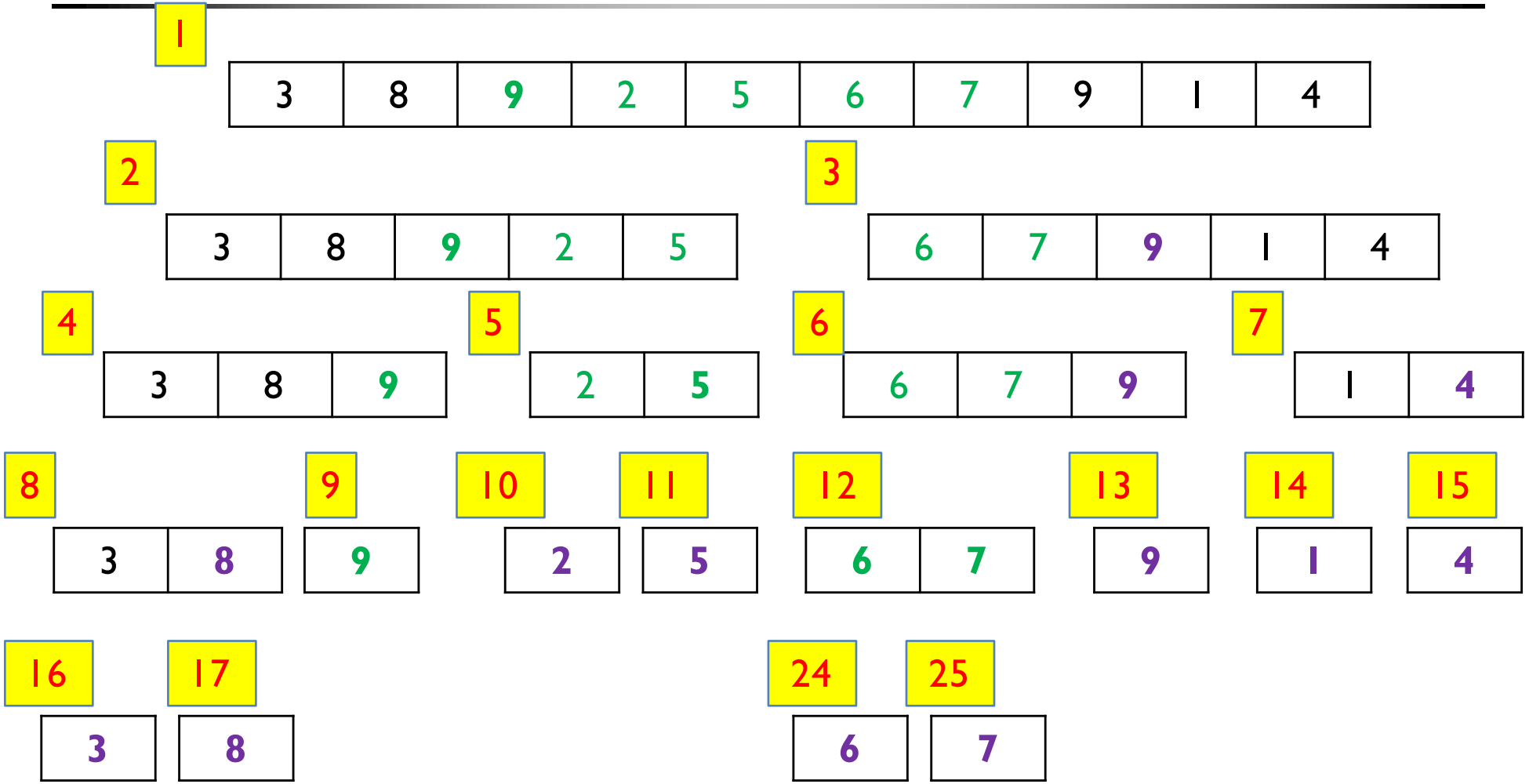


Segment Tree

- Query Segment tree
  - Find the value in range [L, R]
- Implement
  1) There is no overlap between [L, R] and [x, y] => return -INF
  2) [x, y] completely include [L, R] => return the value of this node
  3) Others (partial overlap)=> keep query from left and right subtree

```
int querymax(Node* now,int L,int R,int x,int y)
{
    // (1) There is no overlap between [L, R] and [x, y]
    //      => [x y L R] or [L R x y]
    if(x>R || y<L)  return -INF;
    // (2) [x, y] completely include [L, R]
    //      => [x L R y]
    if(x<=L && y>=R)    return now->valmax;
    // (3) Others (partial overlap)
    int mid=(L+R)>>1;
    return max( querymax(now->l,L,mid,x,y), querymax(now->r,mid+1,R,x,y));
}
```

Segment Tree



Segment Tree

- Modify Segment tree
  - Single node modification
  - Range modification (lazy tag)

- Single node modification
  1) Is this node a leaf
  2) Modify the subtree which contains x
  3) pull

```
void modify(Node* now, int L, int R, int x, int v)
{
    // (1) is this node a leaf
    if(L==R){
        now->update(now->val+v);
        return;
    }
    int mid=(L+R)>>1;
    // (2) modify the subtree which contains x
    if(x<=mid){
        modify(now->l, L, mid, x, v);
    }else{
        modify(now->r, mid+1, R, x, v);
    }
    // (3) pull
    now->pull();
}
```

Practice

POJ-3264: Balanced Lineup

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
- Codeforces
  - 438D

- 基本門檻6題，第二次修課同學請從橘色的題號選擇

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/
- Competitive Programmer‘s Handbook (written by Antti Laaksonen)
- https://cses.fi/book.html