Splay Tree

Sogang ICPC Team
2020 Spring 임지환

Contents

Splay Tree Introduction

- Self-adjusting BBST(Balanced Binary Search Tree)
- 대표적인 BBST인 Red-Black tree보다 구현이 단순한 편

- std::set, std::map의 기능과 더불어 k번째 원소 찾기, 구간 쿼리 등 다양한 역할 수행 가능

- Self-adjustment의 관점
  1. access하고자 하는 노드를 root로 옮기는 연산을 하여 트리의 구조를 변경
  2. 전체 노드들에 대하여 각각의 깊이의 평균을 낮춤

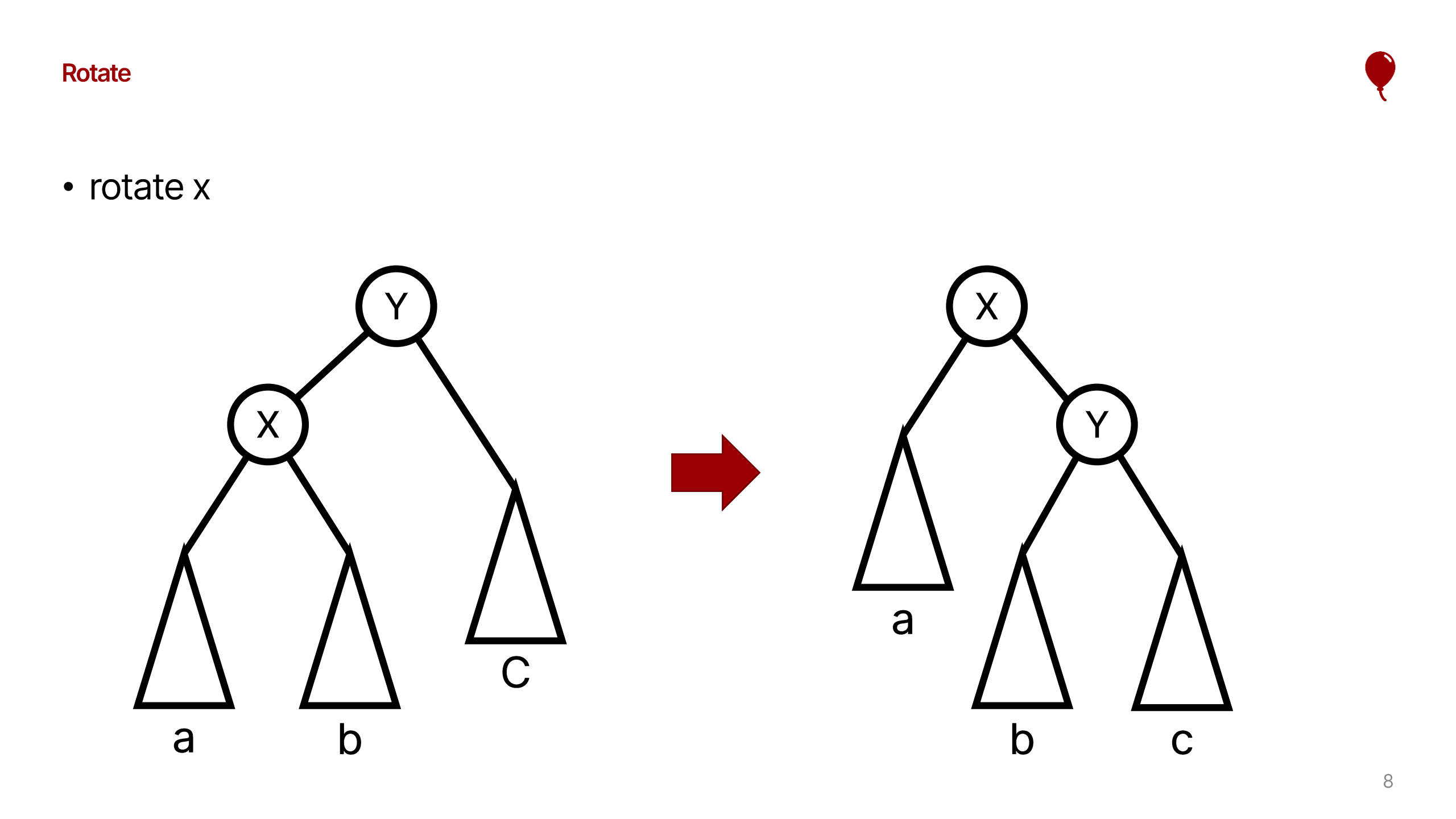
Time complexity Analysis

Basic operations

Basic Operations

- Insertion & deletion & Find
- Rotate & Splay

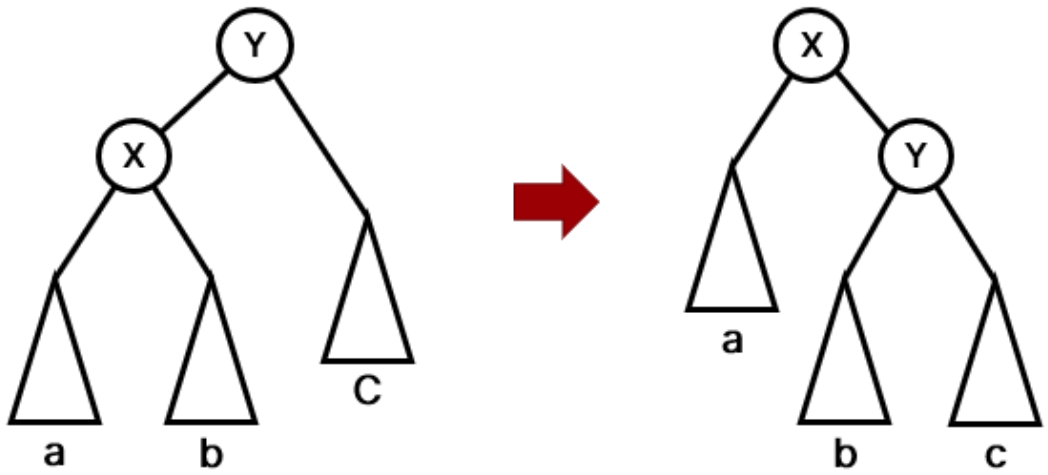
Rotate

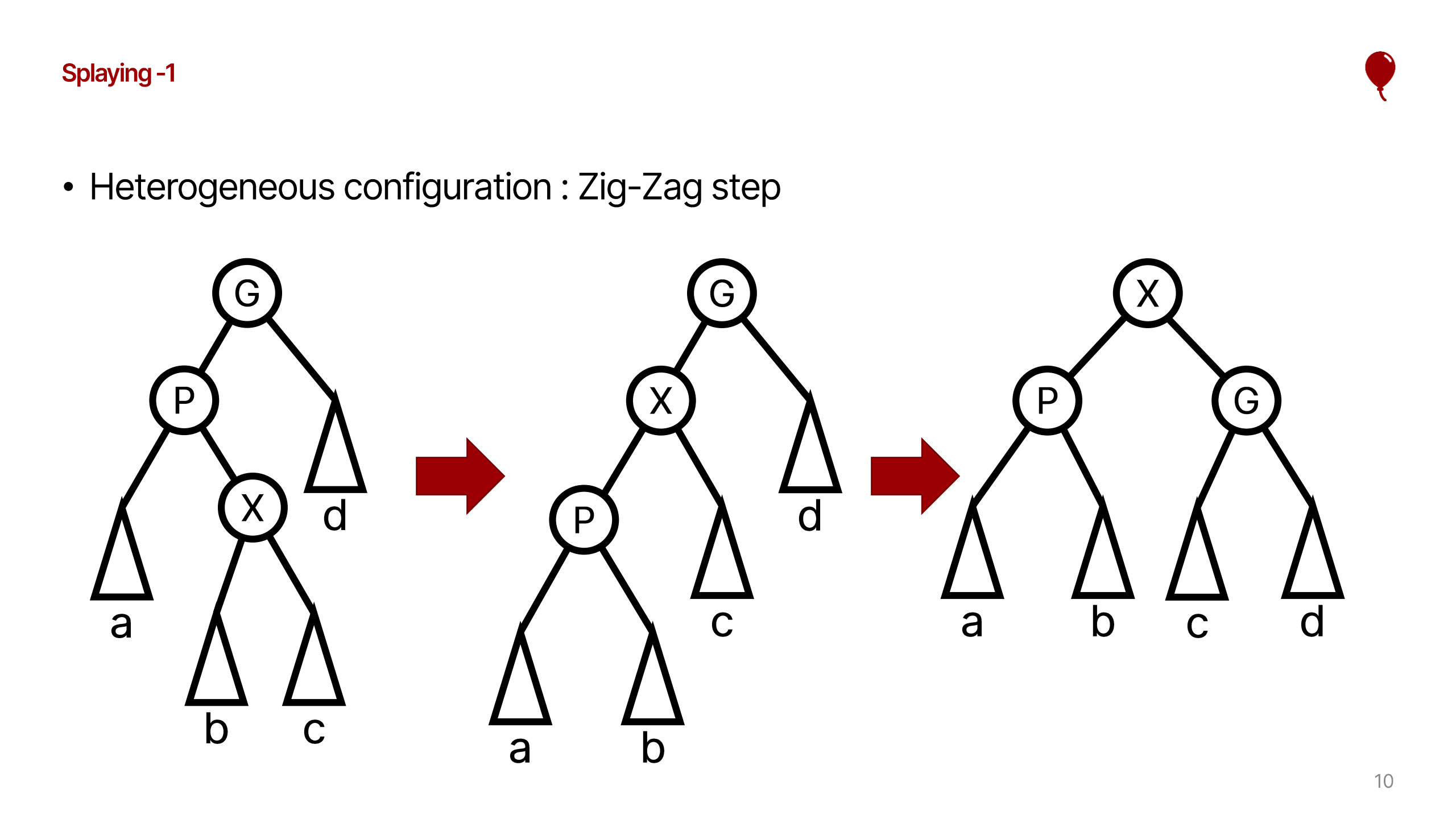


- rotate x

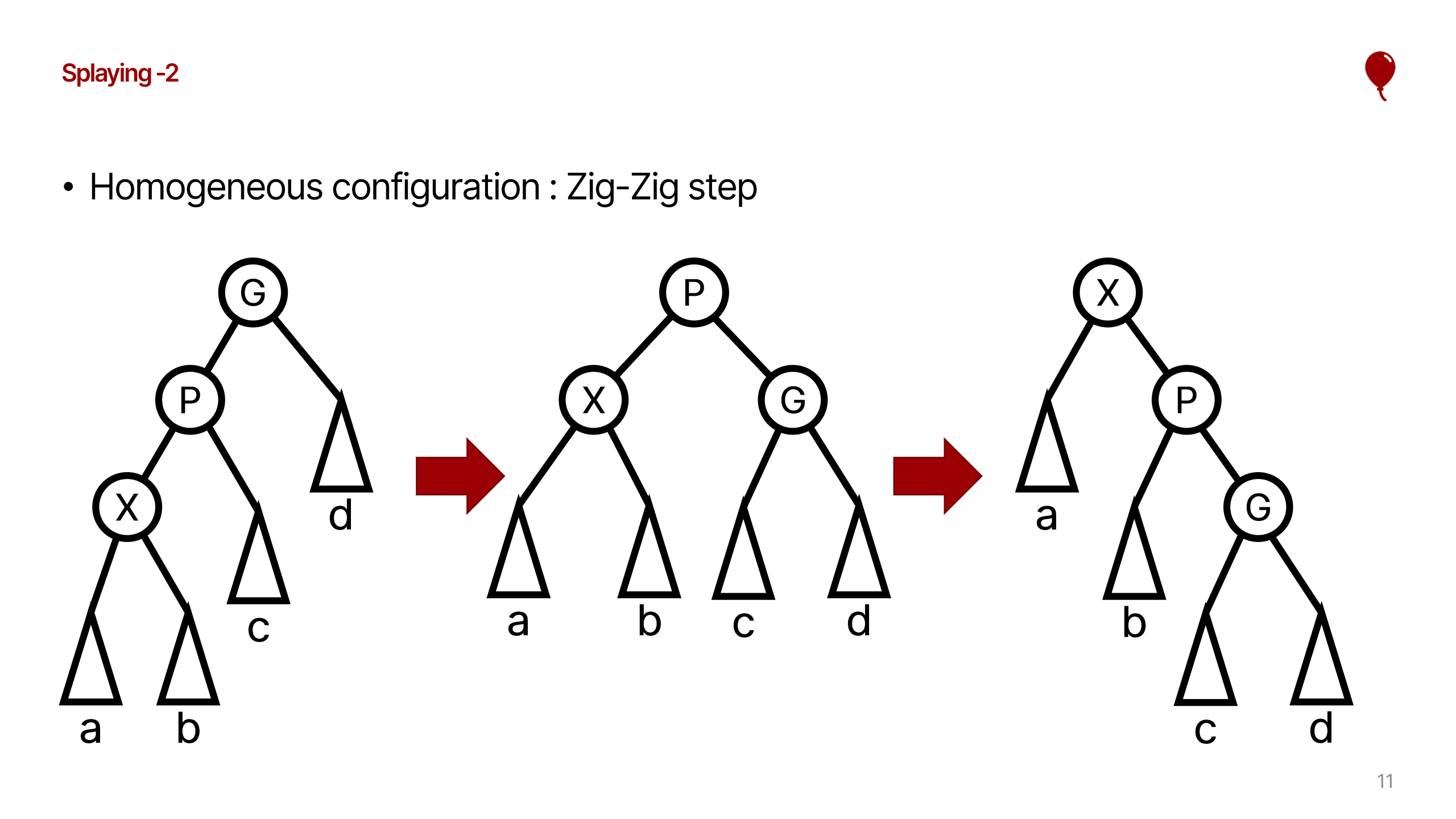
Rotate

```
44  void Rotate(node *x) {
45      node *p = x->p;
46      node *b;
47      if (x == p->l) {
48          p->l = b = x->r;
49          x->r = p;
50      }
51      else {
52          p->r = b = x->l;
53          x->l = p;
54      }
55      x->p = p->p;
56      p->p = x;
57      if (b) b->p = p;
58      (x->p ? p == x->p->l ? x->p->l : x->p->r : tree) = x;
59  }
```





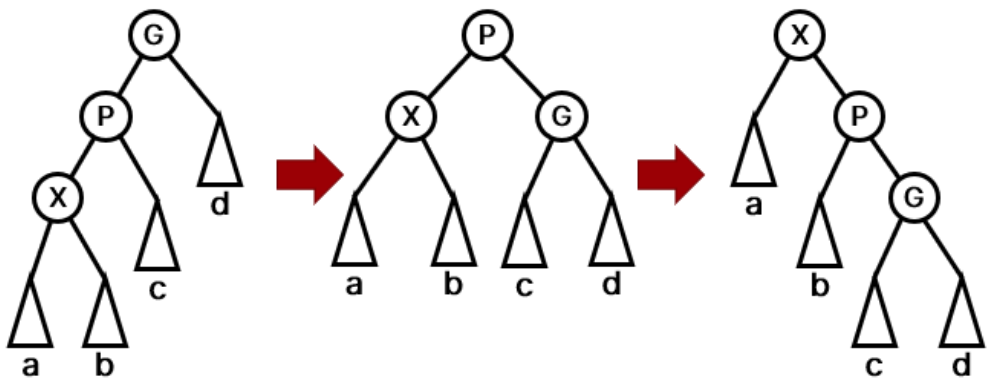
- Heterogeneous configuration : Zig-Zag step

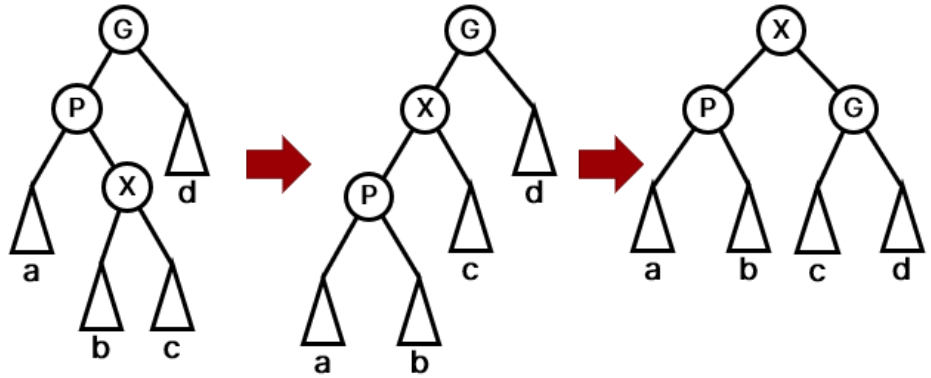


- Homogeneous configuration : Zig-Zig step

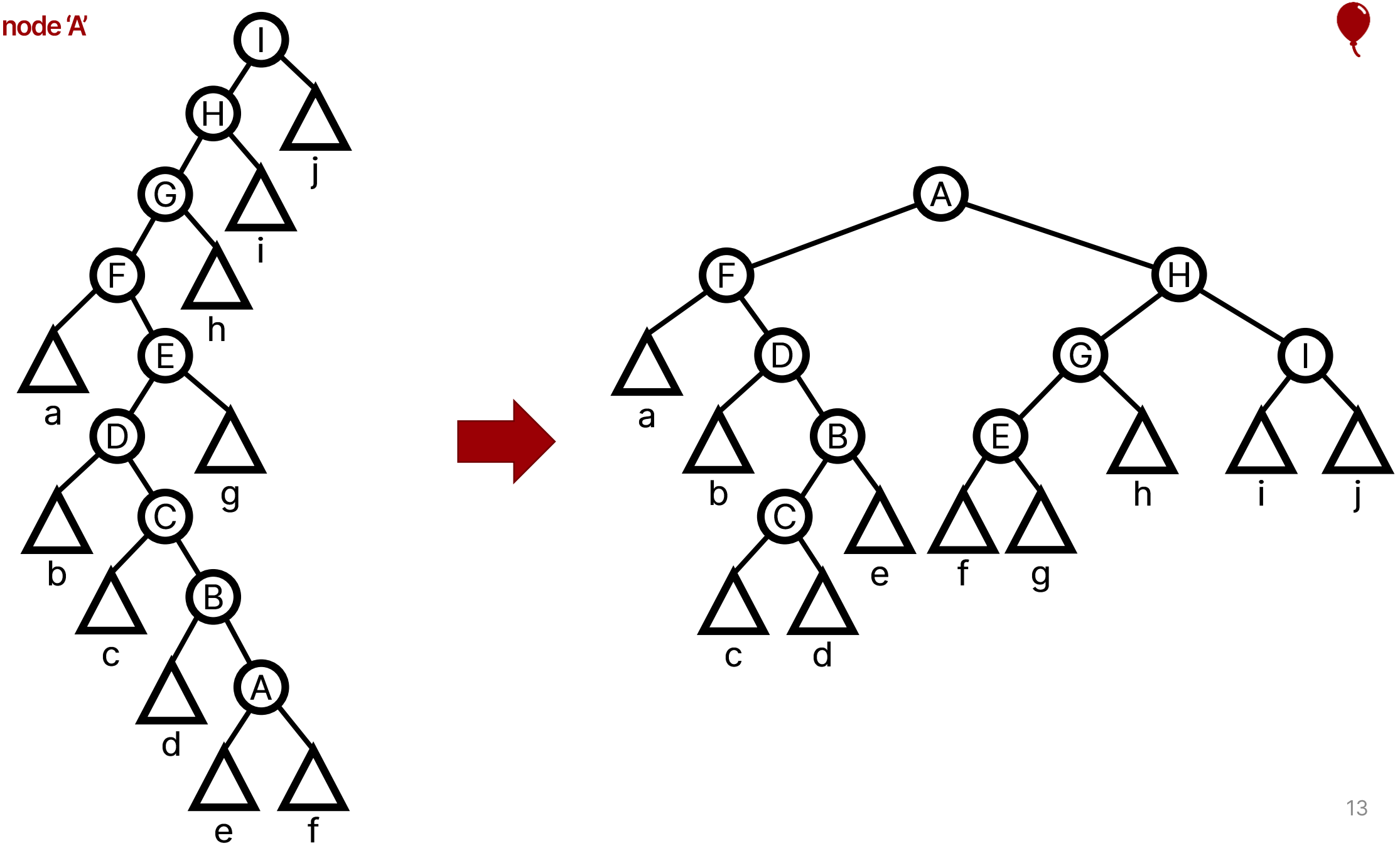
Splaying


```
61  void Splay(node *x) {
62      while (x->p) {
63          node *p = x->p;
64          node *g = p->p;
65          if (g) Rotate((x == p->l) == (p == g->l) ? p : x);
66          Rotate(x);
67      }
68  }
```





Splaying node 'A'



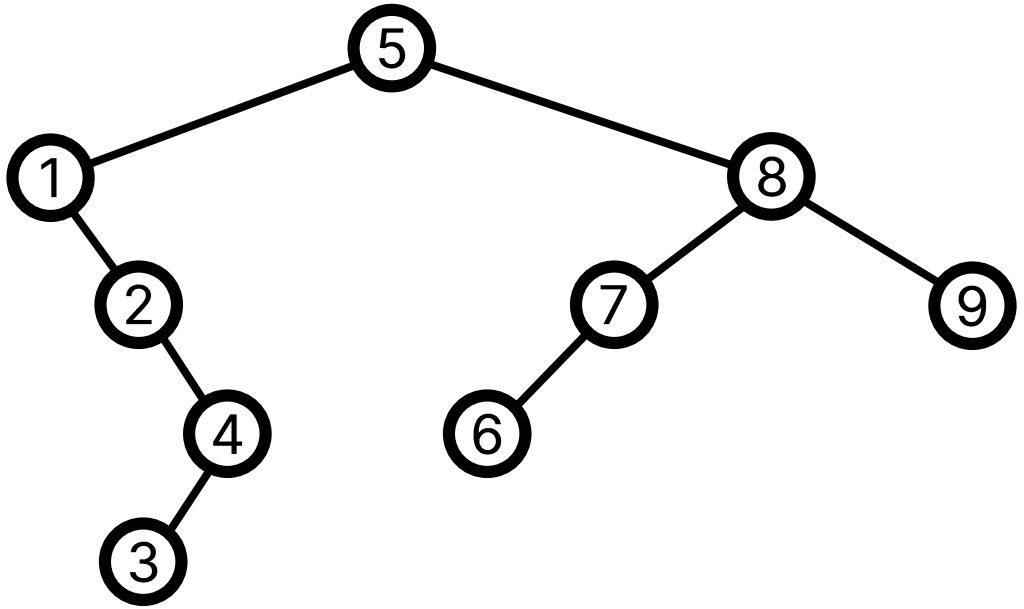
Advanced operations

Finding Kth element

- In case of finding 4th value

$cnt(x)$: the number of nodes in the subtree whose root is x

k=4



Finding Kth element

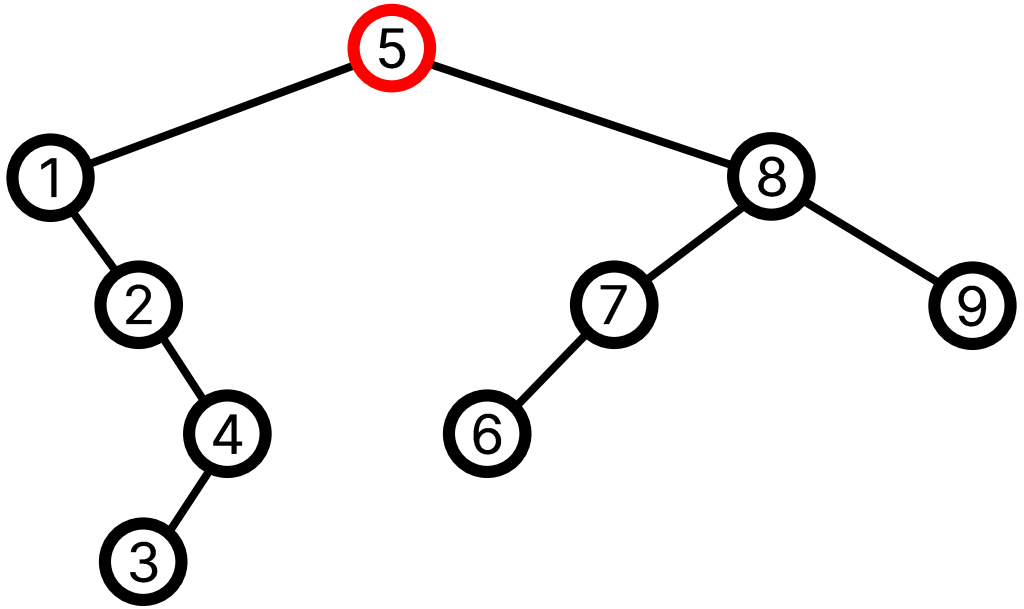
- In case of finding 4th value

$cnt(x)$: the number of nodes in the subtree whose root is x

k=4

$cnt(5 \rightarrow left) = 4$

$k <= cnt(5 \rightarrow left)$



Finding Kth element

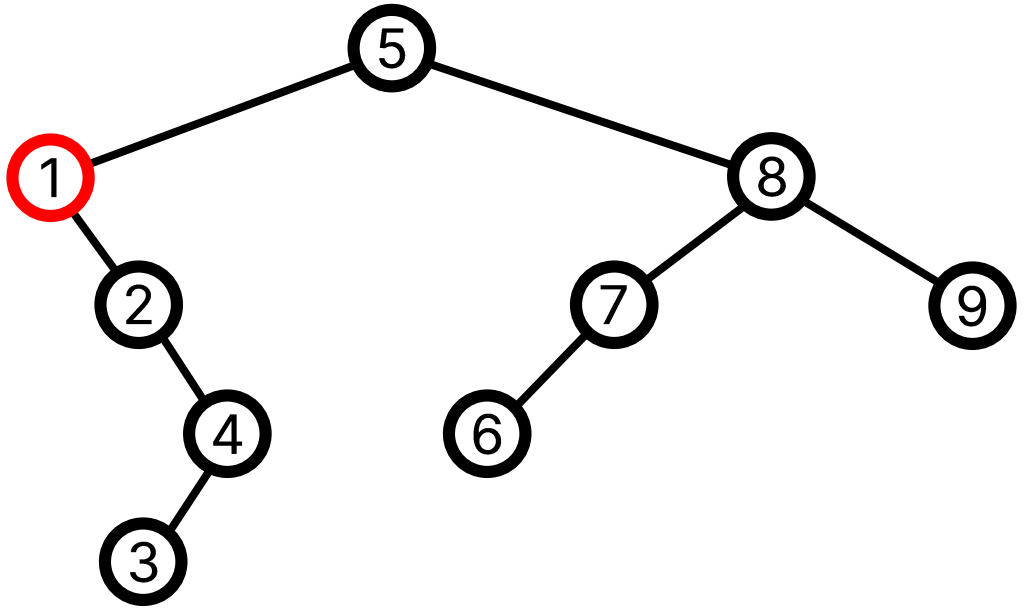
- In case of finding 4th value

$cnt(x)$: the number of nodes in the subtree whose root is x

k=4

$cnt(1 \rightarrow left) = 0$

$k > cnt(1 \rightarrow left)$



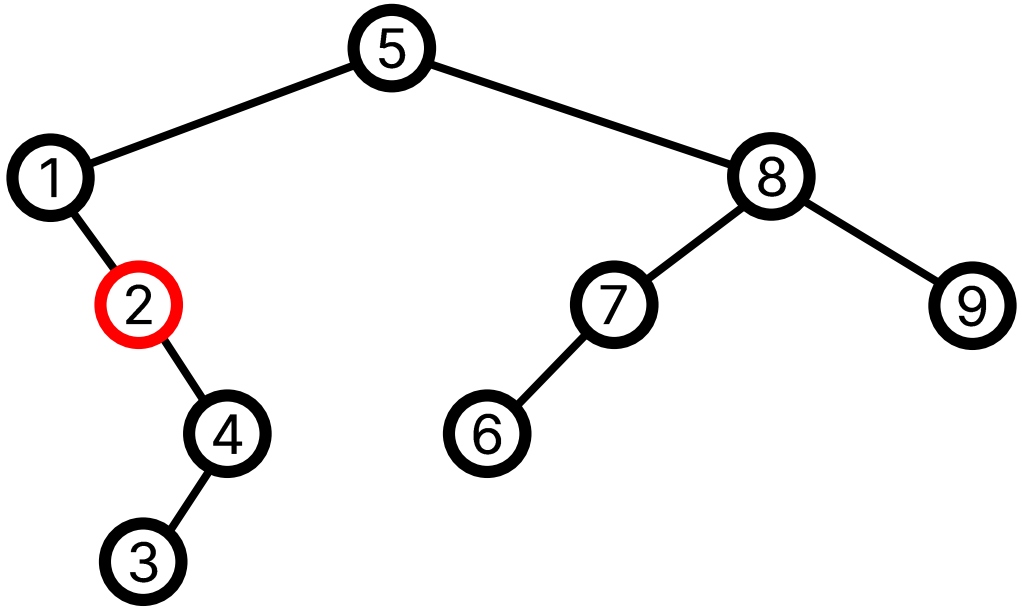
- In case of finding 4th value

$cnt(x)$: the number of nodes in the subtree whose root is x

k=3

$cnt(2 \rightarrow left) = 0$

$k > cnt(2 \rightarrow left)$



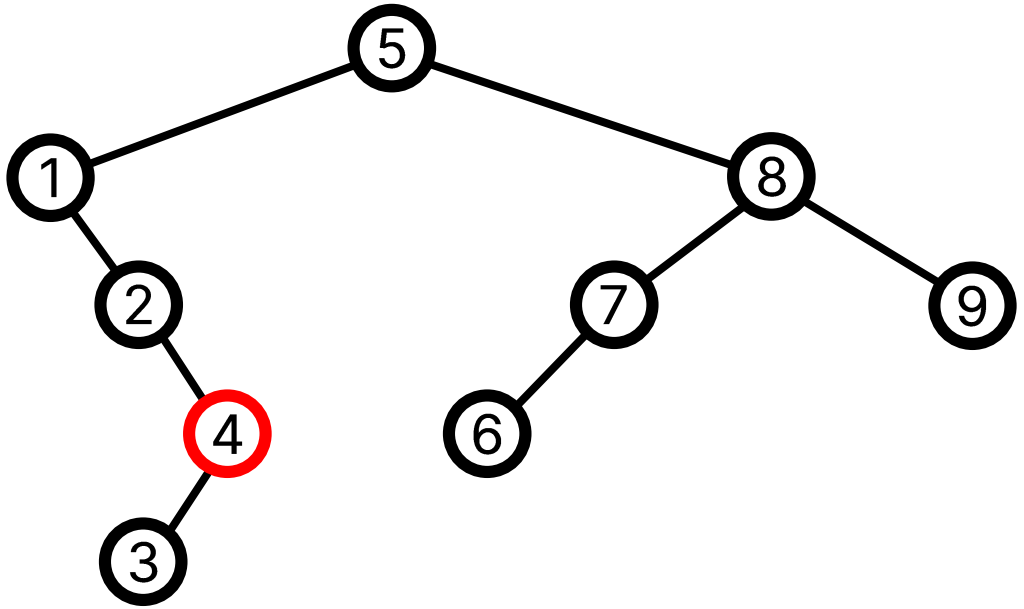
- In case of finding 4th value

$cnt(x)$: the number of nodes in the subtree whose root is x

k=2

$cnt(4 \rightarrow left) = 1$

$k - cnt(2 \rightarrow left) = 1$



Finding Kth element

```
 8  struct node {
 9      node *l, *r, *p;
10      bool inv, dummy;
11      int cnt, v;
12  } *tree;

17  void Update(node *x) {
18      x->cnt = 1;
19      if (x->l) {
20          x->cnt += x->l->cnt;
21      }
22      if (x->r) {
23          x->cnt += x->r->cnt;
24      }
25  }

37  void Rotate(node *x) {
38      node *p = x->p;
39      /* ... */
54      Update(p);
55      Update(x);
56  }
```

17~25: re-structuring이 일어났을 경우 cnt(x)를 update

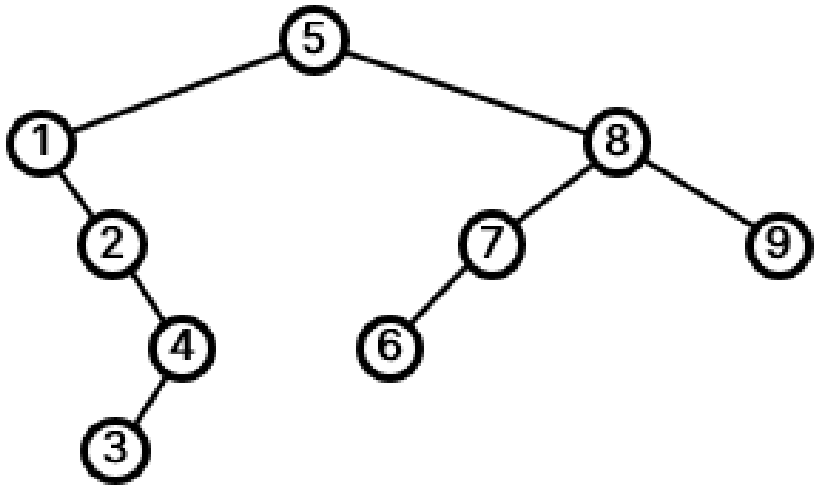
54, 55: bottom-up 형태로 update

p : previous parent of x

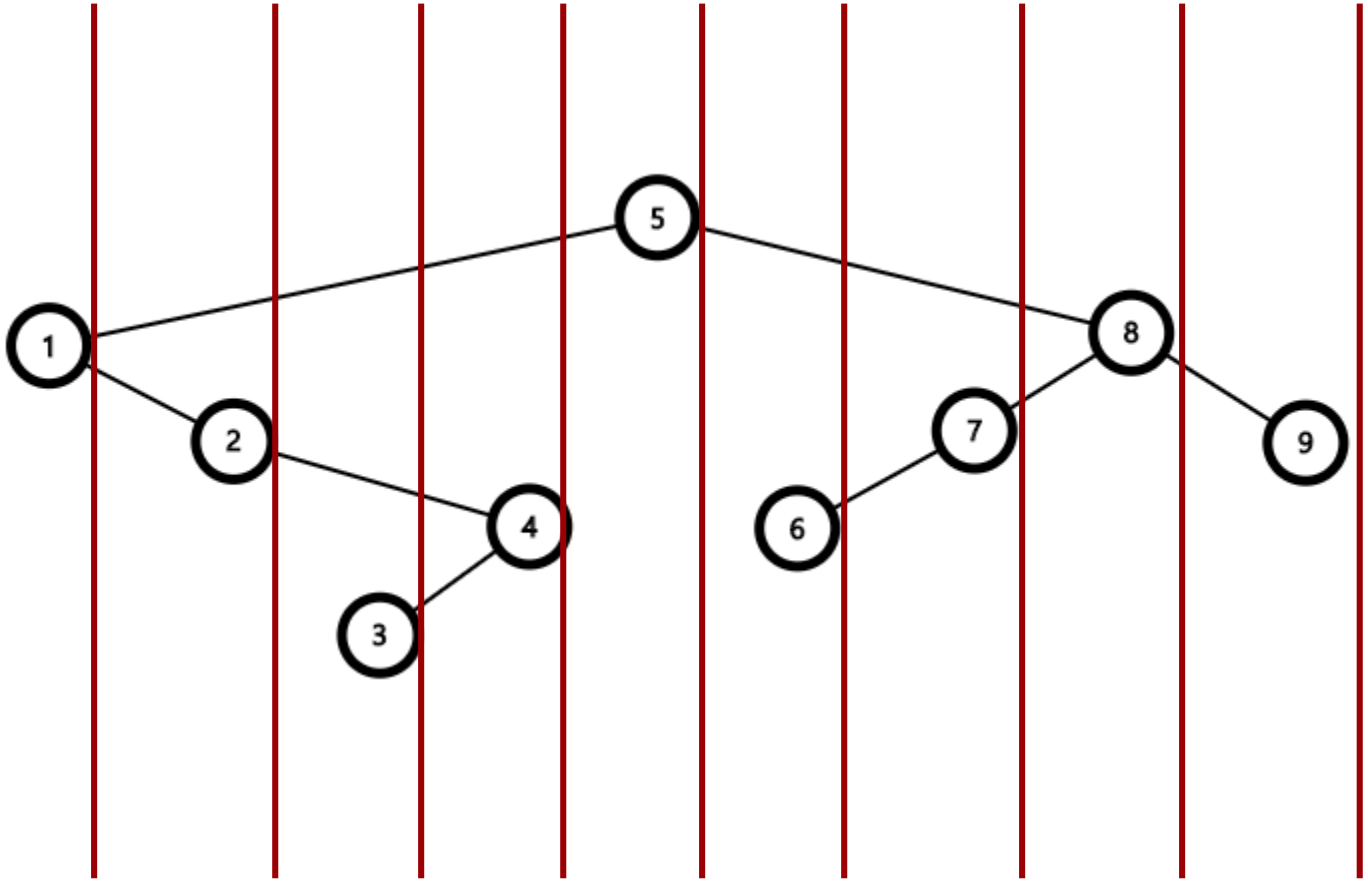
x : rotated, parent of p

Finding Kth element

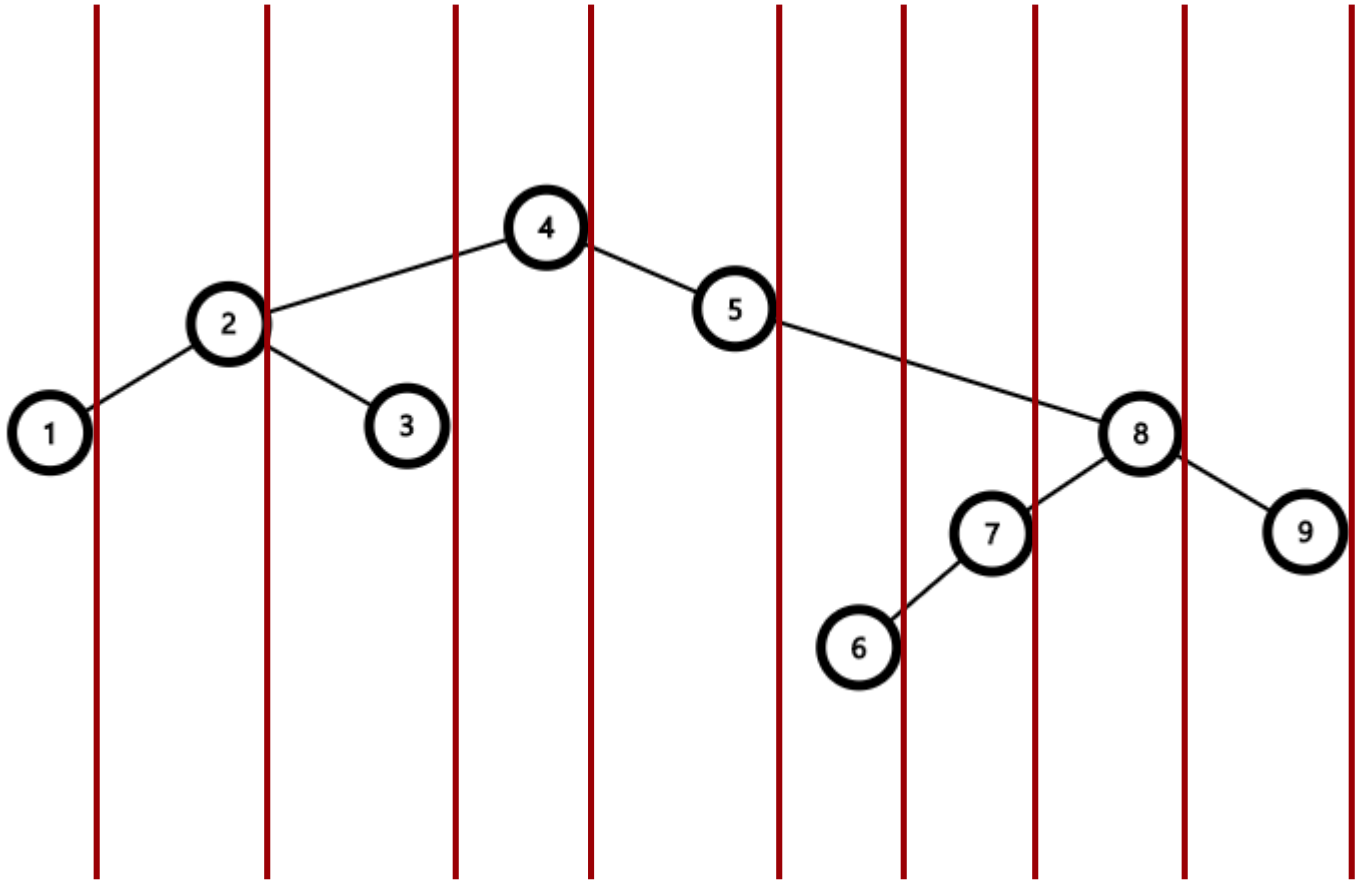
```
146  void Find_Kth(int k) {
147      node *x = tree;
148      while (1) {
149          while (x->l && x->l->cnt > k)
150              x = x->l;
151          if (x->l) k -= x->l->cnt;
152          if (!k--) break;
153          x = x->r;
154      }
155      Splay(x);
156  }
```



Splay Tree on Linear data



Splay Tree on Linear data



CF #452 (Div. 2) F. Letters Removing

Link

- 길이가 $N(1 \leq N \leq 2 \times 10^5)$인 대소문자 알파벳 & 숫자로 구성된 문자열

- $M(1 \leq M \leq 2 \times 10^5)$개의 쿼리
  - $l\ r\ c$: $[l,\ r]$사이의 문자들 중 c인 것들을 모두 제거

- 모든 쿼리가 적용된 후 남은 결과 출력

CF #452 (Div. 2) F. Letters Removing

Link

ex) N=10, M=4

s=“agtFrgF4aF”

Q1 : 2 5 g

Q2 : 4 9 F

Q3 : 1 5 4

Q4 : 1 7 a

1	2	3	4	5	6	7	8	9	10
a	g	t	F	r	g	F	4	a	F

CF #452 (Div. 2) F. Letters Removing

Link

ex) N=10, M=4

s="agtFrgF4aF"

Q1 : 2 5 g

Q2 : 4 9 F

Q3 : 1 5 4

Q4 : 1 7 a

1	2	3	4	5	6	7	8	9	10
a	g	t	F	r	g	F	4	a	F

CF #452 (Div. 2) F. Letters Removing

Link

ex) N=10, M=4

s="agtFrgF4aF"

Q1 : 2 5 g

Q2 : 4 9 F

Q3 : 1 5 4

Q4 : 1 7 a

1	2	3	4	5	6	7	8	9
a	t	F	r	g	F	4	a	F

CF #452 (Div. 2) F. Letters Removing

Link

ex) N=10, M=4

s="agtFrgF4aF"

Q1 : 2 5 g

Q2 : 4 9 F

Q3 : 1 5 4

Q4 : 1 7 a

1	2	3	4	5	6	7
a	t	F	r	g	4	a

CF #452 (Div. 2) F. Letters Removing

Link

ex) N=10, M=4

s="agtFrgF4aF"

Q1 : 2 5 g

Q2 : 4 9 F

Q3 : 1 5 4

Q4 : 1 7 a

1	2	3	4	5	6	7
a	t	F	r	g	4	a

CF #452 (Div. 2) F. Letters Removing

Link

ex) N=10, M=4

s=“agtFrgF4aF”

Q1 : 2 5 g

Q2 : 4 9 F

Q3 : 1 5 4

Q4 : 1 7 a

1	2	3	4	5
t	F	r	g	4

CF #452 (Div. 2) F. Letters Removing

Link

ex) N=10, M=4

s="agtFrgF4aF"

Q1 : 2 5 g

Q2 : 4 9 F

Q3 : 1 5 4

Q4 : 1 7 a

1	2	3	4	5	6	7	8	9	10
a	g	t	F	r	g	F	4	a	F

4^{th} : 5

9^{th} : 10

CF #452 (Div. 2) F. Letters Removing

Link

ex) N=10, M=4

s="agtFrgF4aF"

Q1 : 2 5 g

Q2 : 4 9 F

Q3 : 1 5 4

Q4 : 1 7 a

1	2	3	4	5	6	7	8	9	10
a	g	t	F	r	g	F	4	a	F

1^{st} : 1

5^{th} : 6

CF #452 (Div. 2) F. Letters Removing

Link

ex) N=10, M=4

s="agtFrgF4aF"

Q1 : 2 5 g

Q2 : 4 9 F

Q3 : 1 5 4

Q4 : 1 7 a

1	2	3	4	5	6	7	8	9	10
a	g	t	F	r	g	F	4	a	F

1^{st} : 1

7^{th} : 9

CF #452 (Div. 2) F. Letters Removing

Link

```
154     map<char, set<int>> mp;
155   int main() {
156       ios_base::sync_with_stdio(false); cin.tie(0);
157       cin>>N>>M>>s;
158       for(int i = 0; i<N; i++) {
159           mp[s[i]].insert(i+1);
160           a[i+1] = i + 1;
161       }
162       Initialize(N);
163       while(M--) {
164           int l, r;
165           char c;
166           cin>>l>>r>>c;
167           Find_Kth(l); l = tree->v;
168           Find_Kth(r); r = tree->v;
169           while(1) {
170               auto it = mp[c].lower_bound(l);
171               if(it == mp[c].end() || *it > r) break;
172               Delete(*it);
173               mp[c].erase(it);
174           }
175       }
176       print(tree);
177       return 0;
178   }
```

154: indices of character

159: 1-based index

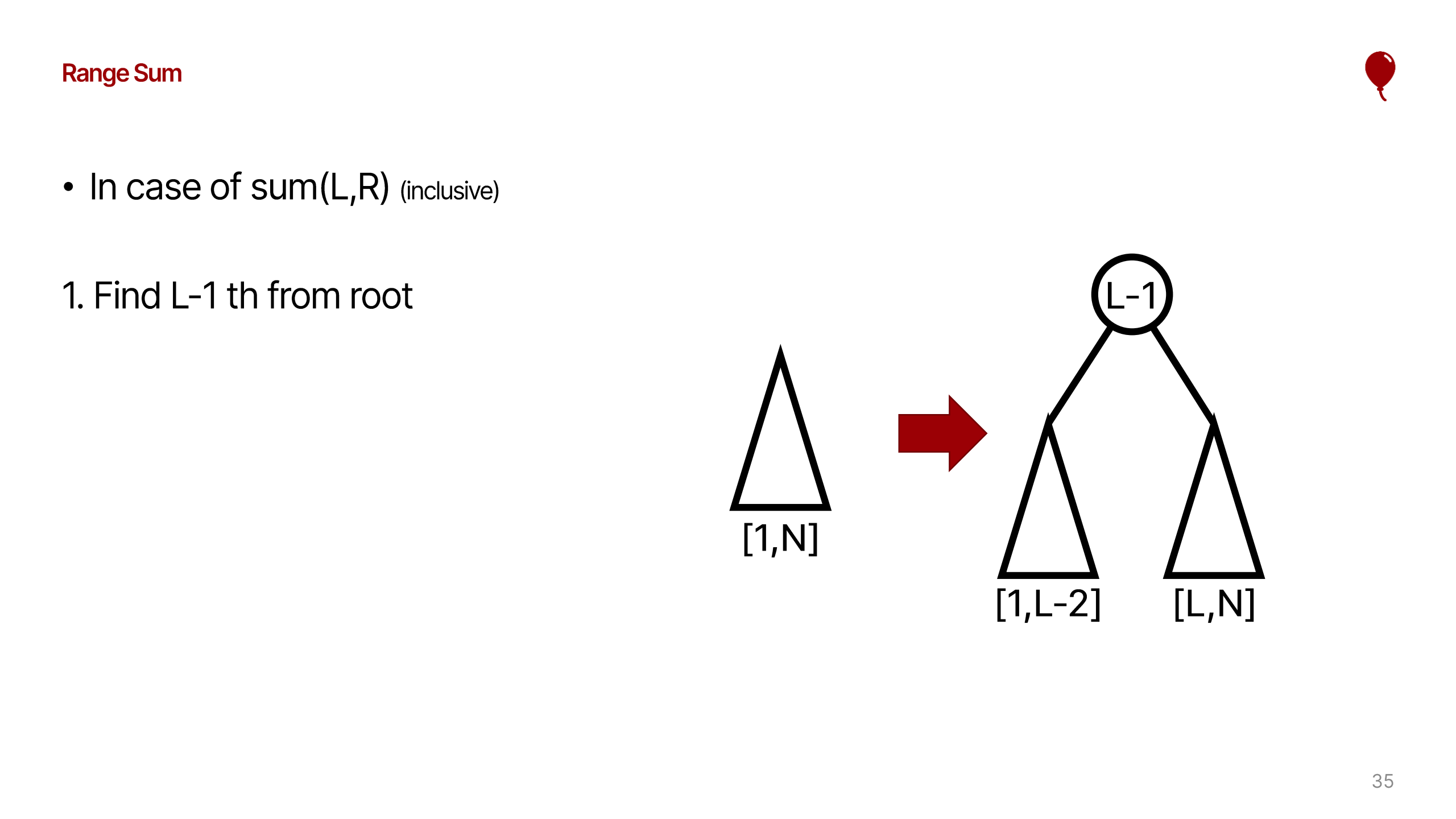
160: 1-based key value

167, 168: actual $[l, r]$ range

176: print by inorder traversal

Full source code :
https://codeforces.com/contest/899/submission/80515825

Range Sum

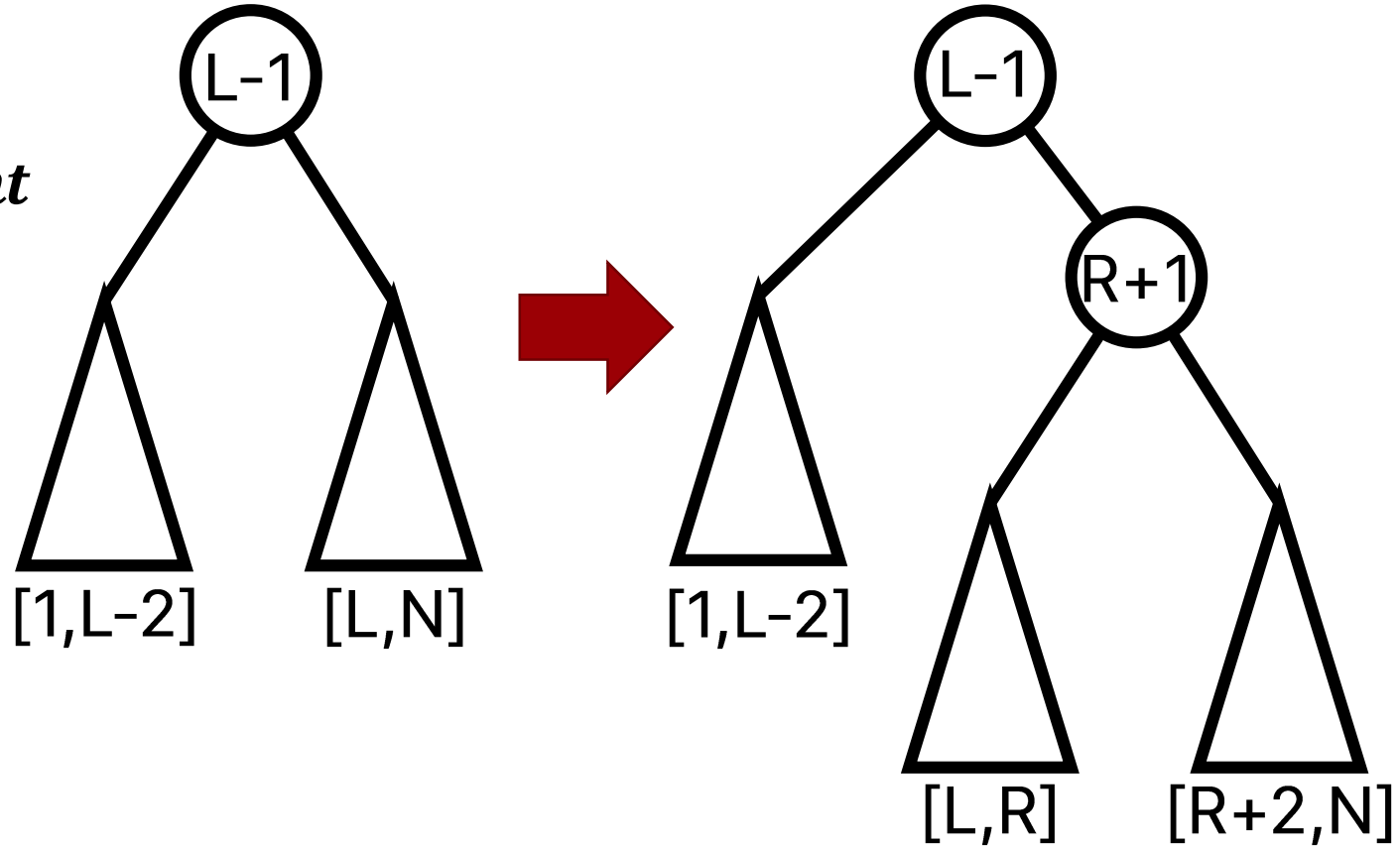


- In case of sum(L,R) (inclusive)

1. Find L-1 th from root

- In case of sum(L,R) (inclusive)

2. Find R+1 th from root
 = Find R+1-L from $(L-1) \rightarrow right$



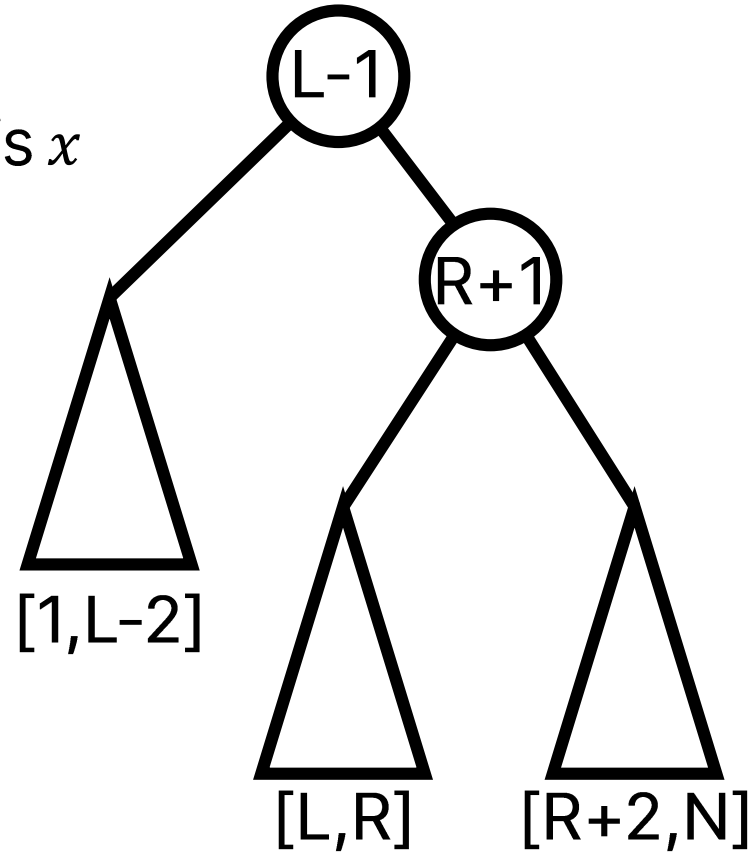
- In case of sum(L,R) (inclusive)

3.

$Sum(x)$ = Sum of nodes in the subtree whose root is x

sum(L,R) = Sum($root \rightarrow right \rightarrow left$)

```
165    void Interval(int l, int r) {
166        Find_Kth(l - 1);
167        node *x = tree;
168        tree = x->r;
169        tree->p = NULL;
170        Find_Kth(r - l + 1);
171        x->r = tree;
172        tree->p = x;
173        tree = x;
174    }
```



Lazy Propagation

- $[\,l, r\,]$ 추출을 통해 range를 하나의 노드로 취급 가능

```
15  void Lazy(node *x) {
16      x->v += x->lz;
17      if(x->l) {
18          x->l->lz += x->lz;
19          x->l->sum += x->l->cnt * x->lz;
20      }
21      if(x->r) {
22          x->r->lz += x->lz;
23          x->r->sum += x->r->cnt * x->lz;
24      }
25      x->lz = 0;
26  }
```

```
7   struct node {
8       node *l, *r, *p;
9       lint v, cnt, sum, lz;
10  } *tree;

41  void Rotate(node *x) {
42      node *p = x->p;
43      node *b;
44      Lazy(p);
45      Lazy(x);
46      if(x == p->l) { ... }

151 void Find_Kth(int k) {
152     node *x = tree;
153     Lazy(x);
154     while(1) {
155         while(x->l && x->l->cnt > k)
156             x = x->l, Lazy(x);
157         if(x->l) k -= x->l->cnt;
158         if(!k--) break;
159         x = x->r;
160         Lazy(x);
161     }
162     Splay(x);
163 }
```

Range flipping

```
 7   struct node {
 8       node *l, *r, *p;
 9       bool inv, dummy;
10       lint v, cnt, sum, mx, mn;
11   } *tree;

34   void Lazy_flip(node *x) {
35       if(!x->inv) return;
36       node *t = x->l;
37       x->l = x->r;
38       x->r = t;
39       x->inv = false;
40       if(x->l) x->l->inv = !x->l->inv;
41       if(x->r) x->r->inv = !x->r->inv;
42   }

184  void Reverse(int l, int r) {
185      Interval(l, r);
186      node *x = tree->r->l;
187      x->inv = !x->inv;
188  }
```

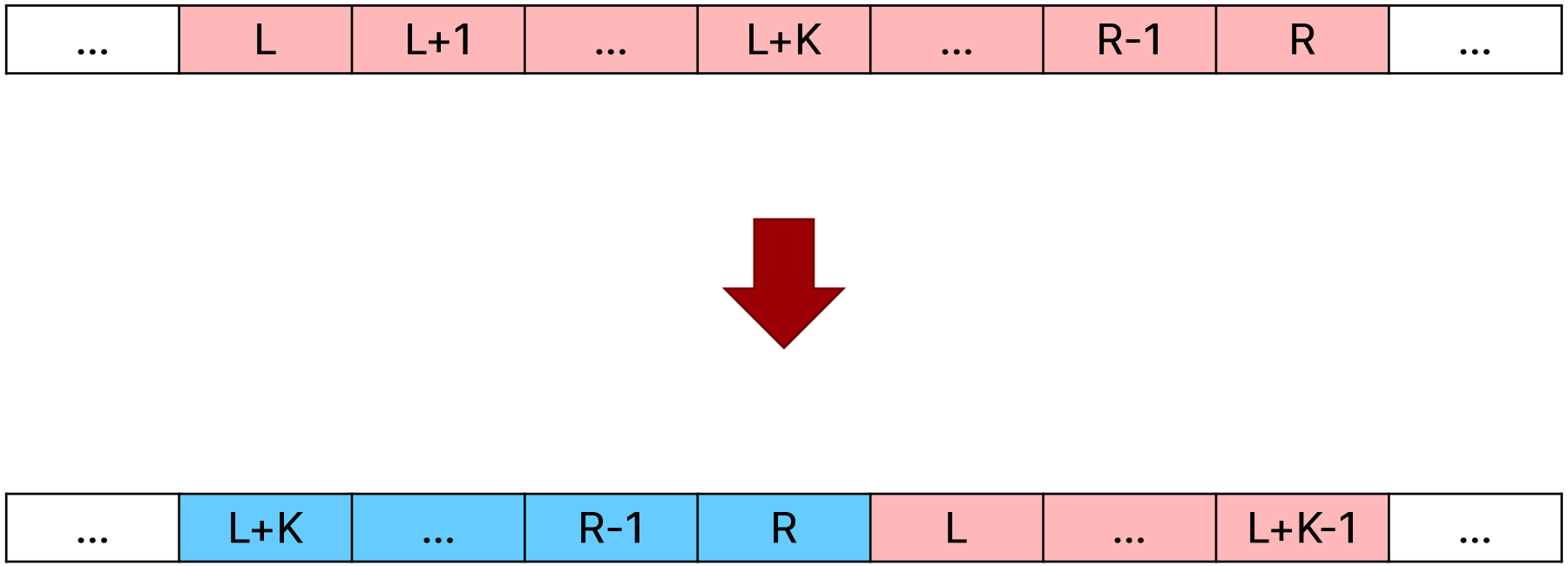
0: all node's inv initially false

36~38 : swap left subtree, right subtree of x

40,41, 187 : change inv state

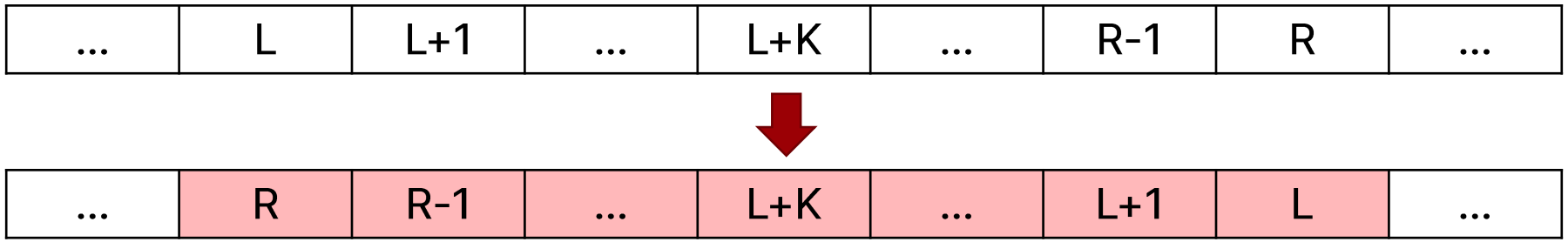
Range Shifting

- Range Shifting [L,R] by -K:



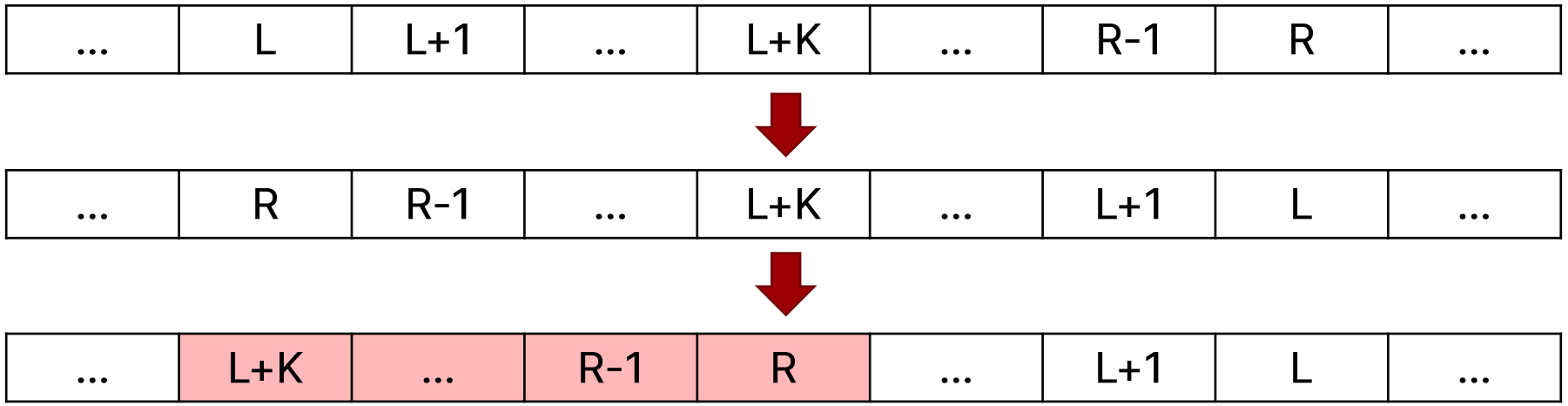
Range Shifting

- Range Shifting [L,R] by -K:



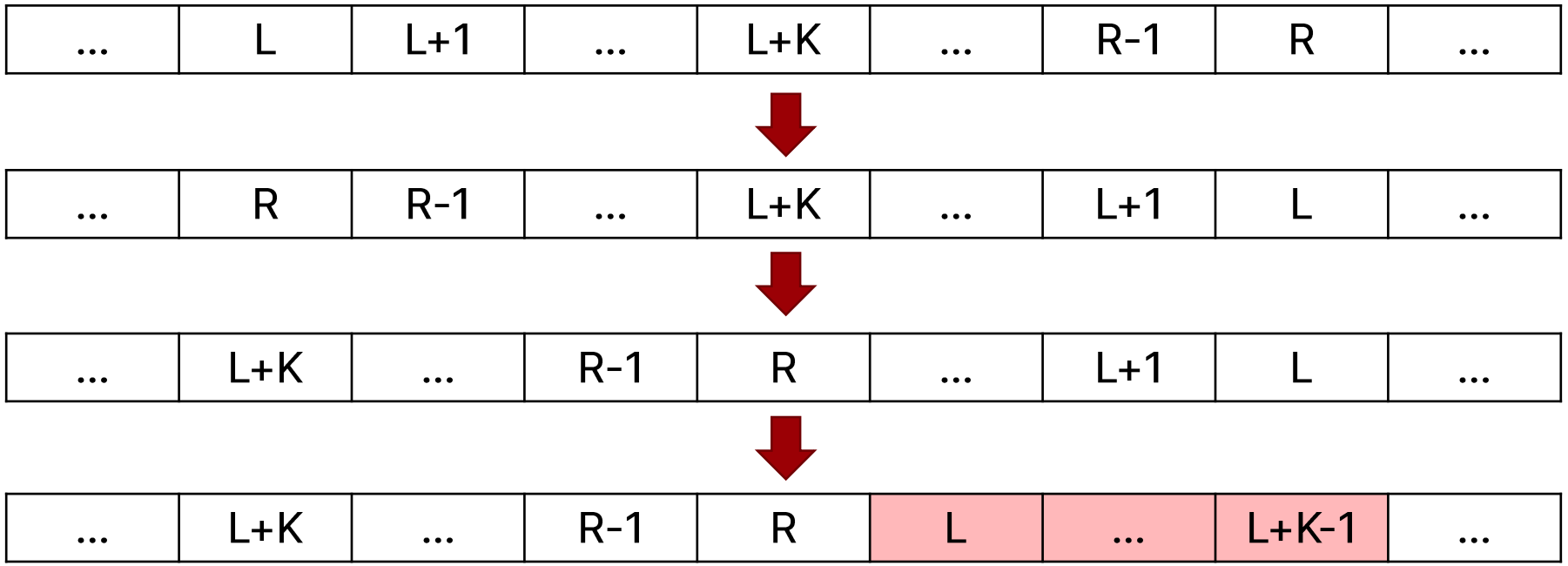
Range Shifting

- Range Shifting [L,R] by -K:



Range Shifting

- Range Shifting [L,R] by -K:



#16994 로프와 쿼리

- 알파벳 소문자로 구성된 길이 $\mathrm{N}(1 \leq \mathrm{N} < 10^5)$의 문자열 S

- $M(1 \leq M \leq 10^5)$개의 쿼리
  - 1 l r : S$[l,\ r]$를 문자열의 가장 앞으로 옮김 $(0 \leq x \leq y < N)$
  - 2 l r : S$[l,\ r]$를 문자열의 가장 뒤로 옮김 $(0 \leq x \leq y < N)$
  - 3 x : S_x 출력 $(0 \leq x < N)$

- shifting left & right를 구현.

참고 자료 :
http://www.secmem.org/blog/2019/03/09/rope/

Full source code :
http://boj.kr/3ed3808bf4e5458a8a3cad4d88135887

#13159 배열

- 크기 $N(1 \leq N \leq 3 \times 10^5)$의 정수 배열
- $Q(1 \leq Q \leq 3 \times 10^5)$개의 쿼리
  - Range Flipping
  - Range shifting
  - Range minimum & maximum query
  - Kth element
  - Target x's index

Summary

- std::map, std::set과 더불어 kth element, 구간 쿼리 처리 등을 위해 사용
- Link/Cut Tree에서 각각의 Path를 관리할 때 사용
  ($O(nlogn)$ by splay tree, others $O(nlog^2n)$)
- see also: treap
  - Antti Laaksonen, Competitive Programmer's Handbook, p253-258
  - 구종만, 알고리즘 문제해결 전략2, p708-718
- https://www.acmicpc.net/workbook/view/912
- https://solved.ac/problems/algorithms/69

자료 출처

1. https://cubelover.tistory.com/10
2. https://justicehui.github.io/hard-algorithm/2018/11/12/SplayTree1/
3. http://www.cs.cmu.edu/~sleator/papers/self-adjusting.pdf
4. Drozdek A. Data Structures and Algorithms in C++ 2nd ed, p264-270