Sorting

- Heap Sort
- Quick Sort
    - psuedocode

```
Quicksort(A,l,h):
    if l < h :
       p <- Partition(A,l,h)
       Quicksort(A,l,p)
       Quicksort(A,p+1,h)


RandomPartition(A,l,h):
    p <- Random(l, h)
    pivot <- A[p]
    i <- l-1
    for (j=l; j <= h-1; j++):
        if arr[j] < pivot:
            i++
            swap arr[i] and arr[j]
    swap arr[i+1] and arr[h]
    return i+1
}

Partition(A,l,h):
    pivot <- A[l]
    i <- l-1
    for (j=l; j <= h-1; j++):
        if arr[j] < pivot:
            i++
            swap arr[i] and arr[j]
    swap arr[i+1] and arr[h]
    return i+1
}
```

    - Randomization

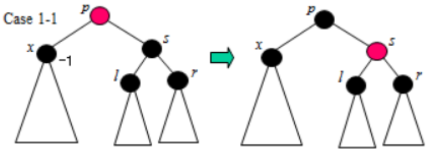
Data Structure(Skip)

- Linked List
- Stack and Queue
- Graph and Tree

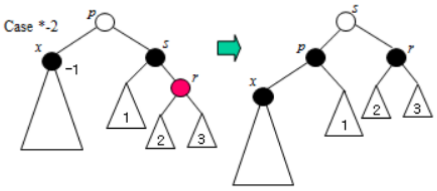
Tree

- Binary Search Tree
- Red Black Tree
    - Lemma
    - Insertion
        - Reconstructing

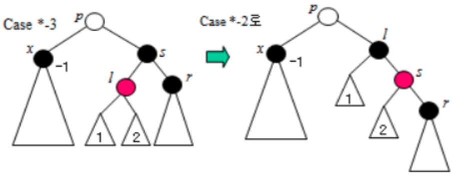
        - Recoloring
    - Deletion
      if the node `z`
        - has two children(left, right), replace it as `SUCCESSOR(z)`
        - is red node, remove it and replace its child `x`
        - is black node
            - the color of child `x` is red
              remove the node `z` and replace it as child `x`.
              And then change the child `x`'s color to black
            - the child color is black
              remove the node `z` and replace it as child `x`.
              there are another nodes as sibling `s`, `l` is left child of `s` , `r` is right child of `s`
                - if the parent color of `x` is red, `<l, r>`(Same numbering is same method to solve)
                    - Case 1-1: <black, black> - *1*
                      exchange color `p`, `s`
                      단순히 p와 s의 색상을 맞바꾼다.



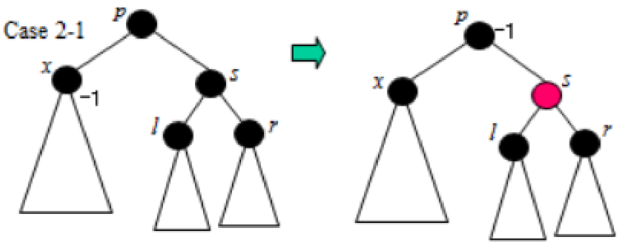
                    - Case 1-2: <*, Red> - *2*
                      reconstructing `p, s, r`
                      exchange color of `p, s` and color `r` as red
                      p를 중심으로 왼쪽으로 회전시키고, p와 s의 색상을 바꾼 다음 r의 색상을 레드에서 블랙으로 바꾼다.



                    - Case 1-3: <Red, Black> - *3*
                      s를 중심으로 오른쪽으로 회전시키고 l과 s의 색상을 맞바꾼다. Case *-2로 이동한다.

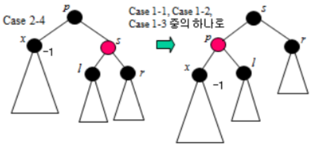


                - if the parent color of `x` is black, <s, l, r>
                    - Case 2-1: <black, black, black> - *4*
                      change `s` as red, and solve it recursively



- Case 2-2: <black, *, red> - *2*
- Case 2-3: <black, red, black> - *3*
- Case 2-4: <red, black, black> - *5*

p를 중심으로 왼쪽으로 회전시키고 p와 s의 색상을 맞바꾼다. l과 r을 경유하는 경로와 관련해서는 문제가 없다. 다만 문제가 발생한 x의 부모 노드의 색상이 블랙에서 레드로 바뀌었다. 이것은 Case 1에 해당하므로 Case 1로 이동한다.



- Height

  The height is over than $\frac{h}{2}$, so tree has internal nodes more than $2^{\frac{h}{2}} - 1$at least. So $n > 2^{\frac{h}{2}} - 1$ is $log_2 n + 1 > \frac{h}{2}$, it is same as $h < 2log_2 n + 1$.

Hash Table

- Direct Message
  - Concept
  - Limitation
- Hash Functions
  - Division Method
  - Multiplication Method
  - Universal Hashing
- Open Addressing

Graph

- BFS
- DFS
- Topological Sort
  - Concept
  - Condition
  - psuedocode

```
int N
bool visit[N]
dfs(x):
    visit[x] = true
    for child in x:
        if visit[child] is false:
            dfs(child)
    insert last of topological_sort array
```

```

topological_sort(){
    for i=0; i<n; ++i:
        if visit[i] is false:
            dfs(visit)
}
```