알고리즘 종류별 풀이방법 정리(Java)

세환 신

January 2025

요 약

알고리즘 종류별 자주 쓰는 접근법의 핵심 코드를 정리해본다.

차 례

1 Binary Search

1.1 UpperIdx, LowerIdx

같은 값이 여러 번 있을 수 있는 상태에서 찾는 대상의 최소 인덱스, 최대 인덱스 구하기

Listing 1: UpperIdx

```
public static int getLowerIdx(int target){
    int st = 0;
    int en = n;

    while(st < en){
        int mid = st + (en - st) / 2;

        if(cards.get(mid) < target){
            st = mid + 1;
        }
        else{
            en = mid;
        }
    }
    return st;
}
```

Listing 2: LowerIdx

```
public static int getLowerIdx(int target){
    int st = 0;
    int en = n;

    while(st < en){
        int mid = st + (en - st) / 2;

        if(cards.get(mid) < target){
            st = mid + 1;
        }
```

```
        else{
            en = mid;
        }
    }
    return st;
}
```

참고문제 - 백준 10816번 : 숫자 카드 2

2 DFS

2.1 트리에서 가장 먼 정점 사이의 거리 구하기

트리에서 아무 노드를 기준으로 해서 BFS를 수행했을 때 결과로 나오는, 최장 거리에 해당하는 노드는 반드시 트리의 지름이 되는 두 정점 중 하나이다.

즉, 어떤 노드 a에서의 최장 거리가 되는 지점을 b라고 했을 때 트리의 지름이 각각 노드 x, y에 의해 구성된다고 했을 때 b는 x, y둘 중 하나이다. 따라서 b에서 다시 BFS를 수행하면 x, y를 모두 구할 수 있다.

t가 a, b사이, x, y사이의 공통된 중간 지점이라고 했을 때 $d(b,t) > d(t,y)$이면 x와 b사이의 거리가 x와 y사이의 거리보다 길기 때문에 트리의 지름이라는 가정에 모순이기 때문이다.

결론적으로 dfs를 두 번 수행하면 해결 가능하다.

참고문제 - 백준 1167번 : 트리의 지름

3 Dynamic Programming

3.1 LIS - 최장 증가 부분 수열

3.1.1 $O(n^2)$풀이

Listing 3: LIS1

```
//dp[i] : Length of the longest sequence among sequences with the i-th
    element as the last element of the subsequence
for(int i = 0; i < n; i++){
    for(int j = 0; j < i; j++){
        if(arr[i] > arr[j]){
            dp[i] = Math.max(dp[i], dp[j] + 1);
        }
    }
}
```

4 Disjoint Set

분리집합 문제에서는 $root$가 서로 다른 두 요소를 병합할 때 아래의 $unionRoot$를 활용하면 트리의 높이가 최소가 되는 형식으로 병합 가능하다. $findRoot$함수를 실행할 때 $parent[x] = findRoot(parent[x])$를 적용해줘서 경로압축도 같이 진행해준다. 경로압축까지 완료된 노드는 편향되지 않아서 $findRoot$함수의 실행시간을 $O(1)$로 유지할 수 있다.

Listing 4: findRoot

```
public static int findRoot(int x){
    if(x == parent[x]){
        return x;
    }

    parent[x] = findRoot(parent[x]);
    return parent[x];
```

```
}
```

Listing 5: unionRoot

```
public static void unionRoot(int x, int y){
    x = findRoot(x);
    y = findRoot(y);

    if(x != y){
        if(node_rank[x] > node_rank[y]){
            parent[y] = x;
        }
        else if(node_rank[x] < node_rank[y]){
            parent[x] = y;
        }
        else {
            parent[x] = y;
            node_rank[y] += 1;
        }
    }
}
```