숨바꼭질 3

#Queue를 이용한 BFS, #DP(동적 계획법)

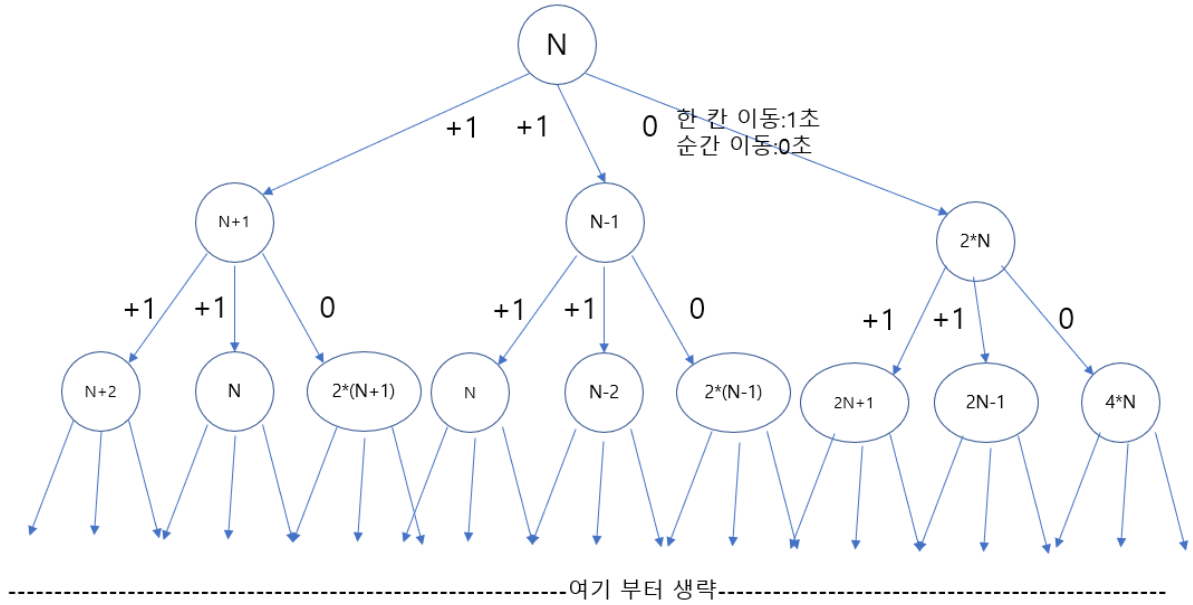
완전 탐색일 때

N은 +1, -1, 2*N 3가지 경우로 움직일 수 있다.

N점을 시작으로 위의 3 경우로 움직여 K에 도달하는 모든 경우 탐색

BUT, 완전 탐색이므로 시간 초과가 일어났다.

⇨ 이것을 해결하고자 DP를 이용한다.



[그림1. 완전 탐색 그림]

(정점의 값은 시작 위치(N)에서 현재 정점의 위치까지 이동했을 때 시간)

(ex) 트리 높이 1에서 N+1의 정점 값은 1초

시간 초과 해결책

완전 탐색을 진행하면서 위의 그림을 보면 트리 높이가 0일 때, N정점에서 탐색을 시작했는데 높이가 3일 때 N정점에서 또 다시 탐색을 시작하는 것을 볼 수 있다. 이 2개의 탐색을 비교해보면

높이 0일 때 N정점 값 ➔ 0초

높이 2일 때 N정점 값 ➔ 2초

더 위의 정점 값이 더 작으므로 BFS를 이용해서 탐색할 때 밑으로 내려가면서 발견된 중복은 무시해도 된다. 이미 먼저 발견된 정점에서 시작한 것이 최소이기 때문에

즉, 탐색하면서 중복된 사례들은 무시하고 새로 발견된 사례들만 캐치하면서 가장 먼저 발견되는 K정점의 값을 발견하면 된다.

코드

```
Int DP[10000]

int sol(int cur){

        if(cur<0 | | cur>100000)return inf;

        if(cur == k)return 0;


        //밑의 빨간줄 코드가 DP

        int &ret = dp[cur];

        //-1이 아니면 예전에 방문했고 이미 최소의 결과가 저장되어 있으
        므로 바로 return

        if(ret!=-1)return ret;

        ret = inf;


        ret = min(ret, sol(cur*2));

        ret = min(ret, sol(cur+1)+1);

        ret = min(ret, sol(cur-1)+1);


        return ret;

}
```