AJOU UNIV.
PROGRAMMING
CONTEST_2019

2019
아주대학교
프로그래밍 경시대회

풀이 슬라이드

1A/2A. APC는 왜 서브태스크 대회가 되었을까?

출제자: 홍준표

	문제번호(쉬운버전의 난이도순)							
	A	B	C	D	E	F	G	H
쉬운버전 (문제당 100점)								
어려운버전 (문제당 40점)								

1A/2A. APC는 왜 서브태스크 대회가 되었을까?

출제자: 홍준표

- 서브태스크1:
  - 현정이는 모든 문제를 풀 수 있다.
  - 현정이의 역량 L이 주어졌을 때
  - 어려운 버전을 풀 수 있다면 무조건 어려운 문제를 푸는 것이 이득
  - 아니라면, 쉬운 버전이라도 푸는게 이득!

- N개의 입력을 받으면서, if문 체크를 반복하며 점수를 계산한다.
- 시간복잡도 : O(N)

1A/2A. APC는 왜 서브태스크 대회가 되었을까?

출제자: 홍준표

- 서브태스크2:
  - 현정이는 K개의 문제만을 풀 수 있다.
  - K개 중 어려운 버전을 최대한 많이 푸는게 이득!
  - 풀 수 있는 어려운 버전을 다 풀고, 아직 문제를 더 풀 수 있다면 아직 풀지 못한 문제의 쉬운 버전을 더 푼다.
- N개의 입력을 받으면서, if문 체크를 반복하며 어려운 버전을 풀 수 있는 문제(A), 쉬운 버전만 풀 수 있는 문제(B) 의 개수를 센다.
- A 중 K의 제한 안에 풀 수 있는 만큼 푼다. K′ = K - min(K, A)
- 더 풀 수 있다면, B 에서 풀 수 있는 만큼 푼다. min(K′, B)

- 시간복잡도 : O(N)

1B/2B. 세훈이의 선물가게

출제자: 김현정

- 서브태스크1: 두 사람은 주문이 들어오면 한 번에 모든 포장을 끝낸다.
  - 입력 순서에 따라 각자 개수만큼 버퍼에 쌓는다.
  - 하나 쌓을 때마다 선물 번호를 하나씩 증가시킨다.
  - 시간복잡도: O(sum(m))

```
int acnt = 0, bcnt = 0, cur_gift = 1;
for(int i = 0; i < N; i++){
    int t, m;
    char c;
    scanf("%d %c %d", &t, &c, &m);

    for(int j = 0; j < m; j++){
        if(c == 'B') a[acnt++] = cur_gift++;
        else b[bcnt++] = cur_gift++;
    }
}
```

1B/2B. 세훈이의 선물가게

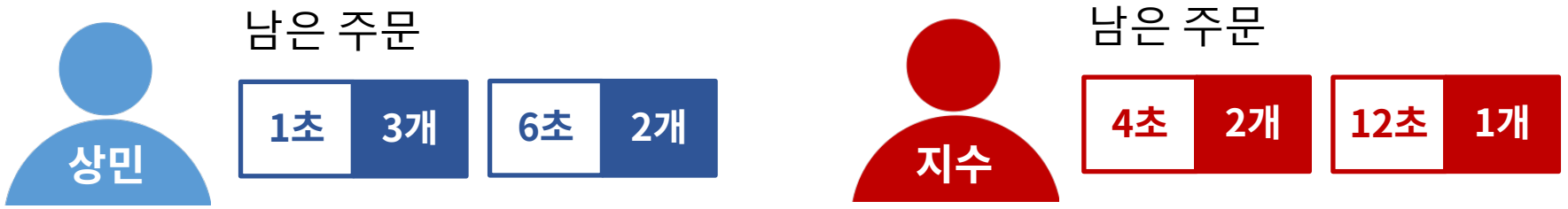
출제자: 김현정

- 서브태스크2: 두 사람은 선물 하나를 포장할 때 시간이 걸린다.
  - 주문 및 포장 시간에 따라 번갈아 선물을 가져간다.
  - 두 사람의 남은 주문 중 더 빨리 시작하는 포장 시간을 찾고
  - 1번 선물부터 차례로 해당 아르바이트생의 버퍼에 쌓는다.

1B/2B. 세훈이의 선물가게

출제자: 김현정

- 서브태스크2: 두 사람은 선물 하나를 포장할 때 시간이 걸린다.
  - 주문 및 포장 시간에 따라 번갈아 선물을 가져간다.
  - 두 사람의 남은 주문 중 더 빨리 시작하는 포장 시간을 찾고
  - 1번 선물부터 차례로 해당 아르바이트생의 버퍼에 쌓는다.

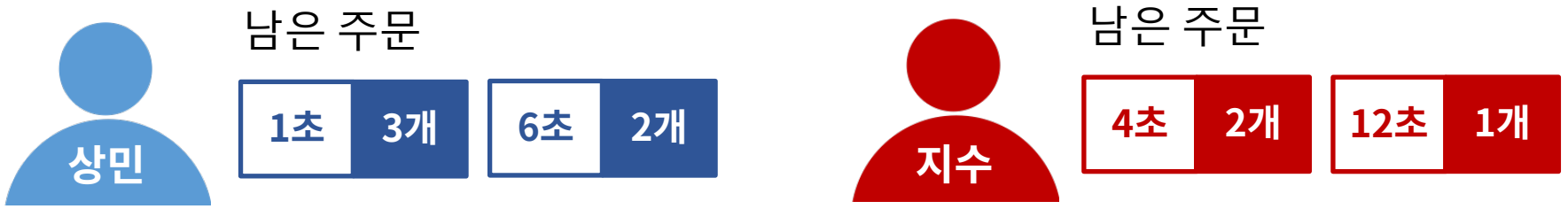


두 사람의 주문을 각자
(시간, 개수)의 형태로 큐에 담는다.

1B/2B. 세훈이의 선물가게

출제자: 김현정

- 서브태스크2: 두 사람은 선물 하나를 포장할 때 시간이 걸린다.
  - 주문 및 포장 시간에 따라 번갈아 선물을 가져간다.
  - 두 사람의 남은 주문 중 더 빨리 시작하는 포장 시간을 찾고
  - 1번 선물부터 차례로 해당 아르바이트생의 버퍼에 쌓는다.



1번 선물부터 차례로 두 사람의 남은 주문 큐에서
가장 앞에 있는 시간이 더 빠른 쪽의 버퍼에 쌓는다.

1B/2B. 세훈이의 선물가게

출제자: 김현정

- 서브태스크2: 두 사람은 선물 하나를 포장할 때 시간이 걸린다.
  - 주문 및 포장 시간에 따라 번갈아 선물을 가져간다.
  - 두 사람의 남은 주문 중 더 빨리 시작하는 포장 시간을 찾고
  - 1번 선물부터 차례로 해당 아르바이트생의 버퍼에 쌓는다.

1번 선물부터 차례로 두 사람의 남은 주문 큐에서
가장 앞에 있는 시간이 더 빠른 쪽의 버퍼에 쌓는다.

1B/2B. 세훈이의 선물가게

출제자: 김현정

- 서브태스크2: 두 사람은 선물 하나를 포장할 때 시간이 걸린다.
  - 주문 및 포장 시간에 따라 번갈아 선물을 가져간다.
  - 두 사람의 남은 주문 중 더 빨리 시작하는 포장 시간을 찾고
  - 1번 선물부터 차례로 해당 아르바이트생의 버퍼에 쌓는다.

이번 선물을 가져간 친구의 가장 앞 주문에서
시간을 포장시간만큼 증가시키고 개수를 하나 줄인다.

1B/2B. 세훈이의 선물가게

출제자: 김현정

- 서브태스크2: 두 사람은 선물 하나를 포장할 때 시간이 걸린다.
  - 주문 및 포장 시간에 따라 번갈아 선물을 가져간다.
  - 두 사람의 남은 주문 중 더 빨리 시작하는 포장 시간을 찾고
  - 1번 선물부터 차례로 해당 아르바이트생의 버퍼에 쌓는다.

이번 선물을 가져간 친구의 가장 앞 주문에서
시간을 포장시간만큼 증가시키고 개수를 하나 줄인다.

1B/2B. 세훈이의 선물가게

출제자: 김현정

- 서브태스크2: 두 사람은 선물 하나를 포장할 때 시간이 걸린다.
  - 주문 및 포장 시간에 따라 번갈아 선물을 가져간다.
  - 두 사람의 남은 주문 중 더 빨리 시작하는 포장 시간을 찾고
  - 1번 선물부터 차례로 해당 아르바이트생의 버퍼에 쌓는다.

1B/2B. 세훈이의 선물가게

출제자: 김현정

- 서브태스크2: 두 사람은 선물 하나를 포장할 때 시간이 걸린다.
  - 주문 및 포장 시간에 따라 번갈아 선물을 가져간다.
  - 두 사람의 남은 주문 중 더 빨리 시작하는 포장 시간을 찾고
  - 1번 선물부터 차례로 해당 아르바이트생의 버퍼에 쌓는다.

1B/2B. 세훈이의 선물가게

출제자: 김현정

- 서브태스크2: 두 사람은 선물 하나를 포장할 때 시간이 걸린다.
  - 주문 및 포장 시간에 따라 번갈아 선물을 가져간다.
  - 두 사람의 남은 주문 중 더 빨리 시작하는 포장 시간을 찾고
  - 1번 선물부터 차례로 해당 아르바이트생의 버퍼에 쌓는다.

1B/2B. 세훈이의 선물가게

출제자: 김현정

- 서브태스크2: 두 사람은 선물 하나를 포장할 때 시간이 걸린다.
  - 주문 및 포장 시간에 따라 번갈아 선물을 가져간다.
  - 두 사람의 남은 주문 중 더 빨리 시작하는 포장 시간을 찾고
  - 1번 선물부터 차례로 해당 아르바이트생의 버퍼에 쌓는다.

~ 건너뜀 ~

같은 로직으로 4번 선물까지 처리한 상태

1B/2B. 세훈이의 선물가게

출제자: 김현정

- 서브태스크2: 두 사람은 선물 하나를 포장할 때 시간이 걸린다.
  - 주문 및 포장 시간에 따라 번갈아 선물을 가져간다.
  - 두 사람의 남은 주문 중 더 빨리 시작하는 포장 시간을 찾고
  - 1번 선물부터 차례로 해당 아르바이트생의 버퍼에 쌓는다.

앞 원소의 개수가 0개가 되면 해당 시간을 기록하고 큐에서 제거한다.
새롭게 가장 앞에 온 원소의 시간을 둘 중 더 나중인 시간으로 갱신한다.

1B/2B. 세훈이의 선물가게

출제자: 김현정

- 서브태스크2: 두 사람은 선물 하나를 포장할 때 시간이 걸린다.
  - 주문 및 포장 시간에 따라 번갈아 선물을 가져간다.
  - 두 사람의 남은 주문 중 더 빨리 시작하는 포장 시간을 찾고
  - 1번 선물부터 차례로 해당 아르바이트생의 버퍼에 쌓는다.

앞 원소의 개수가 0개가 되면 해당 시간을 기록하고 큐에서 제거한다.
새롭게 가장 앞에 온 원소의 시간을 둘 중 더 나중인 시간으로 갱신한다.

1B/2B. 세훈이의 선물가게

출제자: 김현정

- 서브태스크2: 두 사람은 선물 하나를 포장할 때 시간이 걸린다.
  - 주문 및 포장 시간에 따라 번갈아 선물을 가져간다.
  - 두 사람의 남은 주문 중 더 빨리 시작하는 포장 시간을 찾고
  - 1번 선물부터 차례로 해당 아르바이트생의 버퍼에 쌓는다.

두 사람의 비교 시간이 같다면
이번 선물은 상민이가 가져간다.

1B/2B. 세훈이의 선물가게

출제자: 김현정

- 서브태스크2: 두 사람은 선물 하나를 포장할 때 시간이 걸린다.
  - 주문 및 포장 시간에 따라 번갈아 선물을 가져간다.
  - 두 사람의 남은 주문 중 더 빨리 시작하는 포장 시간을 찾고
  - 1번 선물부터 차례로 해당 아르바이트생의 버퍼에 쌓는다.

두 사람의 비교 시간이 같다면
이번 선물은 상민이가 가져간다.

1B/2B. 세훈이의 선물가게

출제자: 김현정

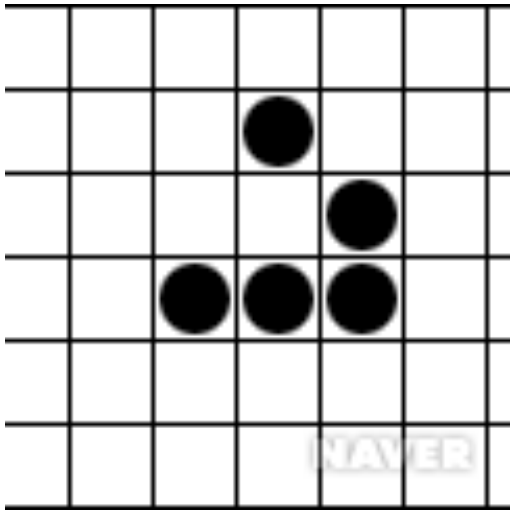
- 서브태스크2: 두 사람은 선물 하나를 포장할 때 시간이 걸린다.
  - 주문 및 포장 시간에 따라 번갈아 선물을 가져간다.
  - 두 사람의 남은 주문 중 더 빨리 시작하는 포장 시간을 찾고
  - 1번 선물부터 차례로 해당 아르바이트생의 버퍼에 쌓는다.

남은 주문이 없을 때까지 반복한다.
시간복잡도: O(sum(m))

2C. 생명 게임

출제자: 홍준표

- 서브태스크1:

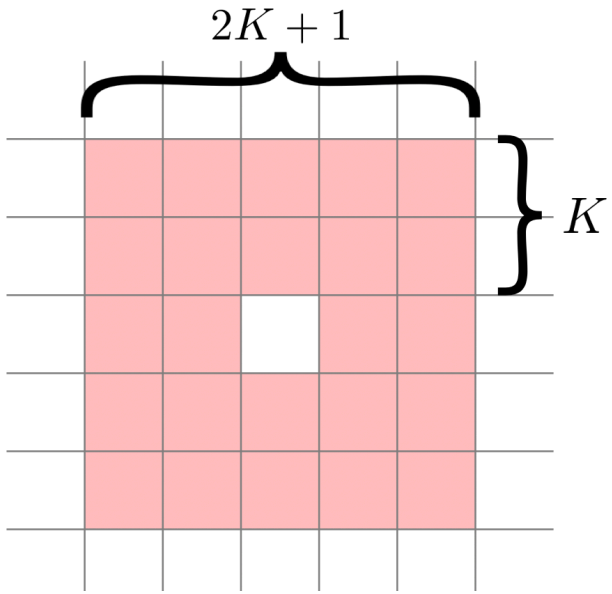


  - 단순한 시뮬레이션
  - 주변의 생명 개수를 세며 다음 상태를 기록
- 시간복잡도 : O(NMT x 8)

2C. 생명 게임

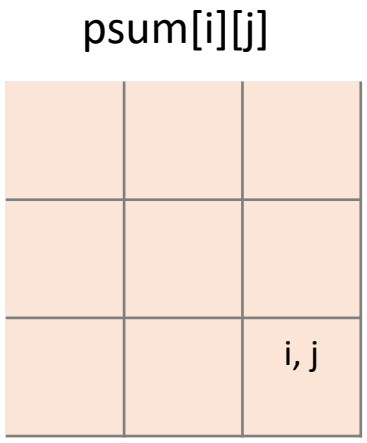
출제자: 홍준표

- 서브태스크2:
  - 빠르게 넓은 범위의 생명 개수를 세어야함.
  - 단순히 반복문을 돌게 되면
  - O(NMT x K^2) 로 시간초과



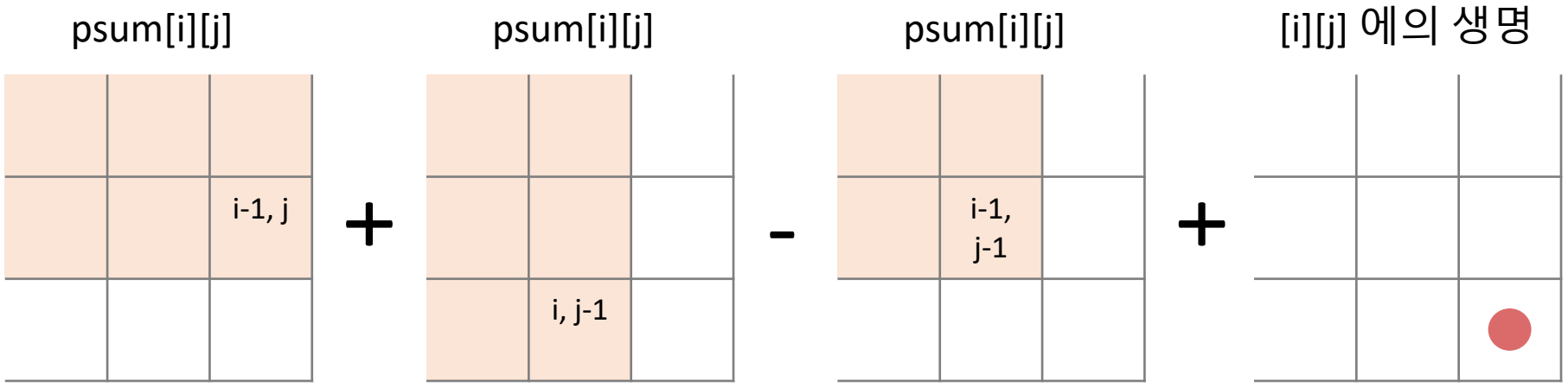
2C. 생명 게임

출제자: 홍준표



=

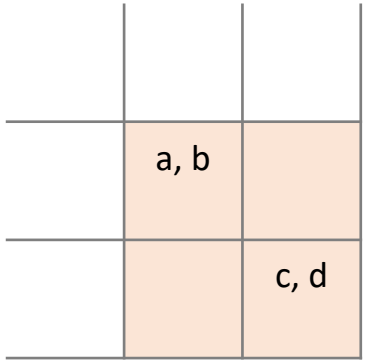
- 2차원 배열 psum을 정의
- psum[i][j] = [1~i][1~j] 까지의 생명의 개수
- psum[i][j] = psum[i-1][j] + psum[i][j-1]
  – psum[i-1][j-1] + ([i][j]에의 생명)
- psum을 완성하는 데 드는 시간복잡도 : O(NM)



2C. 생명 게임

출제자: 홍준표

[a-c][b-d] 의 생명 개수

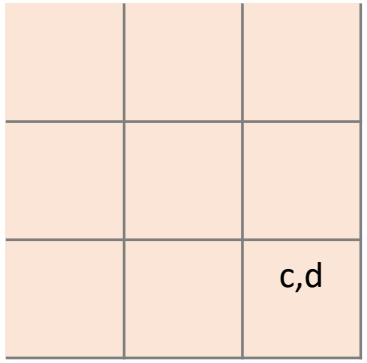


=

- 만든 psum을 활용해서 어떻게 가져오지?
- [a~c][b~d] 의 생명 개수 =

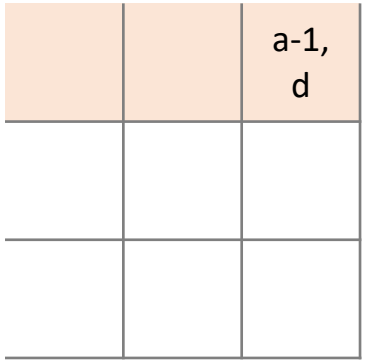
  psum[c][d] – psum[a-1][d] – psum[c][b-1] + psum[a-1][b-1]
- 값을 가져오는 데 드는 시간복잡도 : O(1)

psum[c][d]



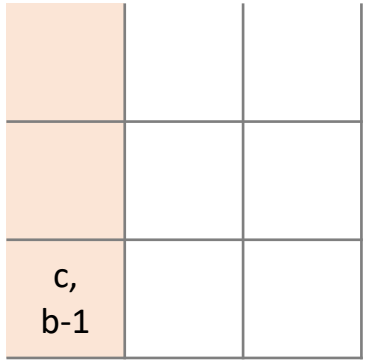
-

psum[a-1][d]



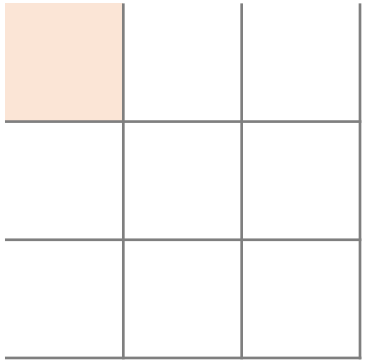
-

psum[c][b-1]



+

psum[a-1][b-1]

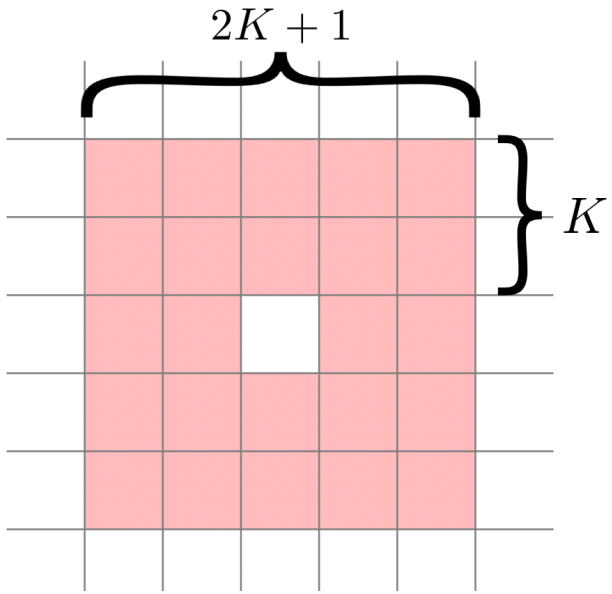


2C. 생명 게임

출제자: 홍준표

- 이를 활용해서 매 시간마다 psum을 구하고, 상태를 만들면 문제 해결!

- 시간복잡도 : O(NMT)



1C/2D. 묘수풀이: 모독

출제자: 김현정

- 서브태스크1: 현정이의 전장에는 최대 하나의 하수인이 존재한다.
  - N=0: 현재 전장에서 모독각이 성립하는지 확인한다.
  - N=1: 현정이의 전장의 하수인이 상대 전장의 어떤 하수인을 공격할 지 결정
    - [1, M]번의 하수인을 공격한 뒤, 모독각이 성립하는지 확인한다.
    - 상대 하수인을 공격하지 않는다는 선택지도 있다.

  - 위 상황 중 하나라도 가능하면 로그를 출력한다.
  - 모독을 항상 마지막에 써도 가능/불가능이 바뀌지 않음이 보장된다.
  - 시간복잡도: O(M)

1C/2D. 묘수풀이: 모독

출제자: 김현정

- 서브태스크2: 현정이 전장에 N, 상대 전장에 M 만큼의 하수인이 존재한다.
  - 현정이의 [1, N]번 하수인이 각자 공격할 상대 하수인을 결정한다.
  - 각 하수인은 공격하지 않거나, 상대 하수인 [1, M]번 중 하나를 공격할 수 있다.
  - 전체 $(M+1)^N$가지 상태에 대해 공격을 진행하고, 모독각을 확인한다.

1C/2D. 묘수풀이: 모독

출제자: 김현정

- 서브태스크2: 현정이 전장에 N, 상대 전장에 M 만큼의 하수인이 존재한다.
  - 현정이의 [1, N]번 하수인이 각자 공격할 상대 하수인을 결정한다.
  - 각 하수인은 공격하지 않거나, 상대 하수인 [1, M]번 중 하나를 공격할 수 있다.
  - 전체 $(M+1)^N$가지 상태에 대해 공격을 진행하고, 모독각을 확인한다.

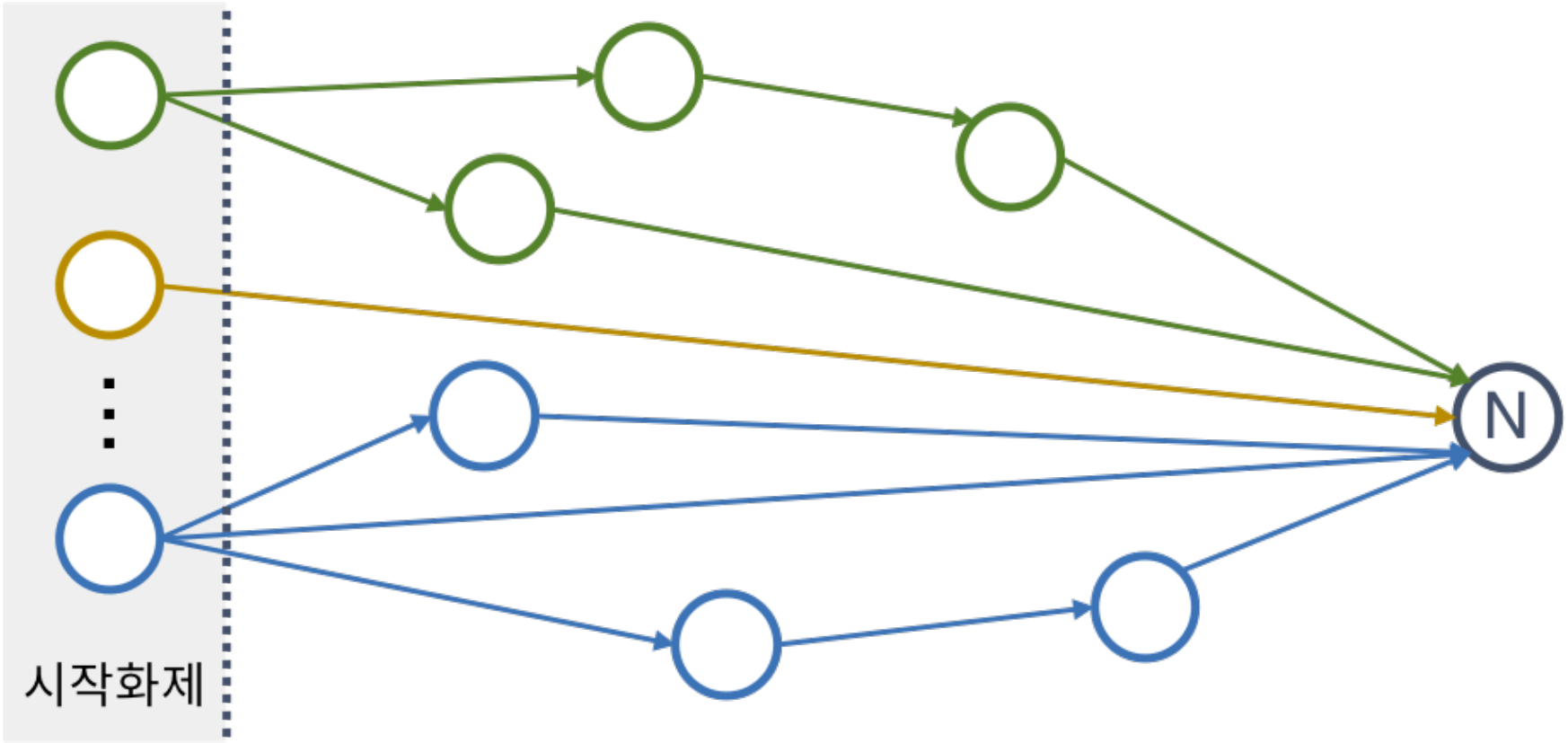
- 주의할 점: 전체 공격 상대를 결정 후 공격력이 작은 하수인부터 공격한다.
  - 현정이의 n1, n2가 상대방의 m1을 모두 공격해야만 모독각이 나오는 경우
  - n1의 공격력이 n2보다 크고, m1의 생명력이 n1의 공격력보다 작을 때
  - n1이 먼저 m1을 공격하면 m1이 죽기 때문에 n2가 m1을 공격할 수 없다.

  - 시간복잡도: $O((N+M) \times (M+1)^N)$

1D. 그래서 팩 주냐?

출제자: 김현정

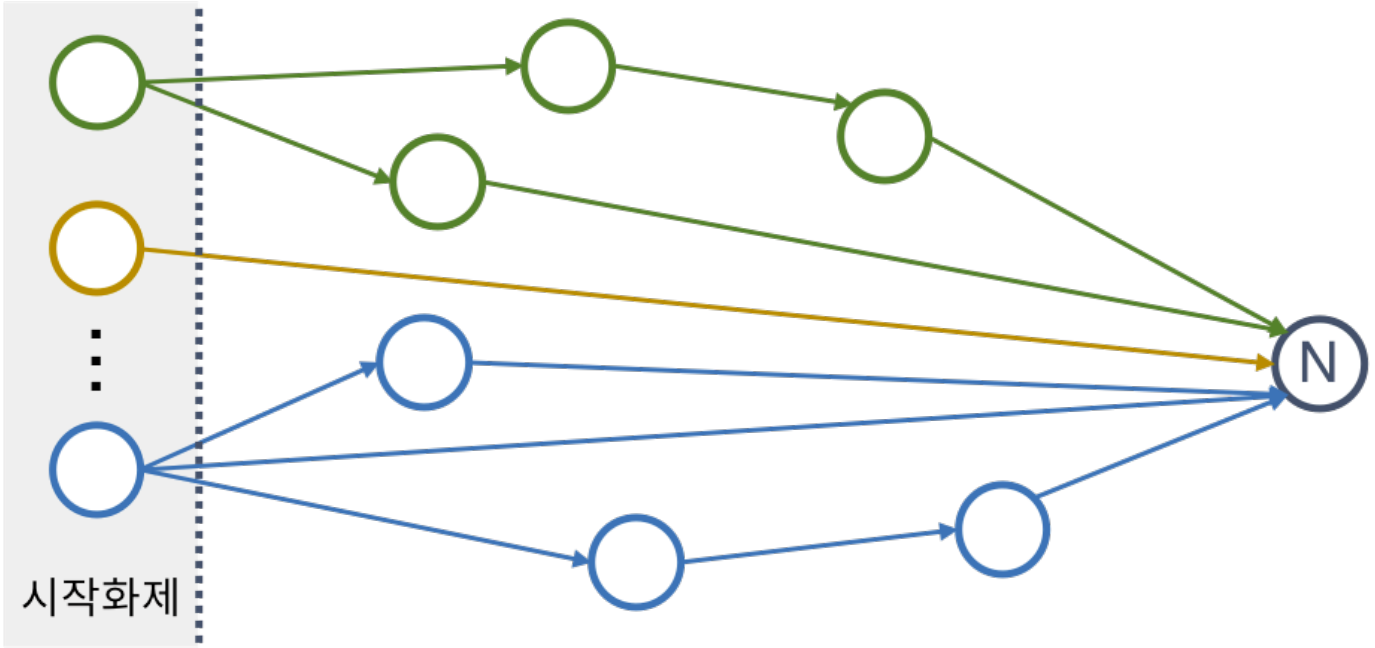
- 서브태스크1: 입력으로 주어지는 그래프의 형태는 아래와 같다.



1D. 그래서 팩 주냐?

출제자: 김현정

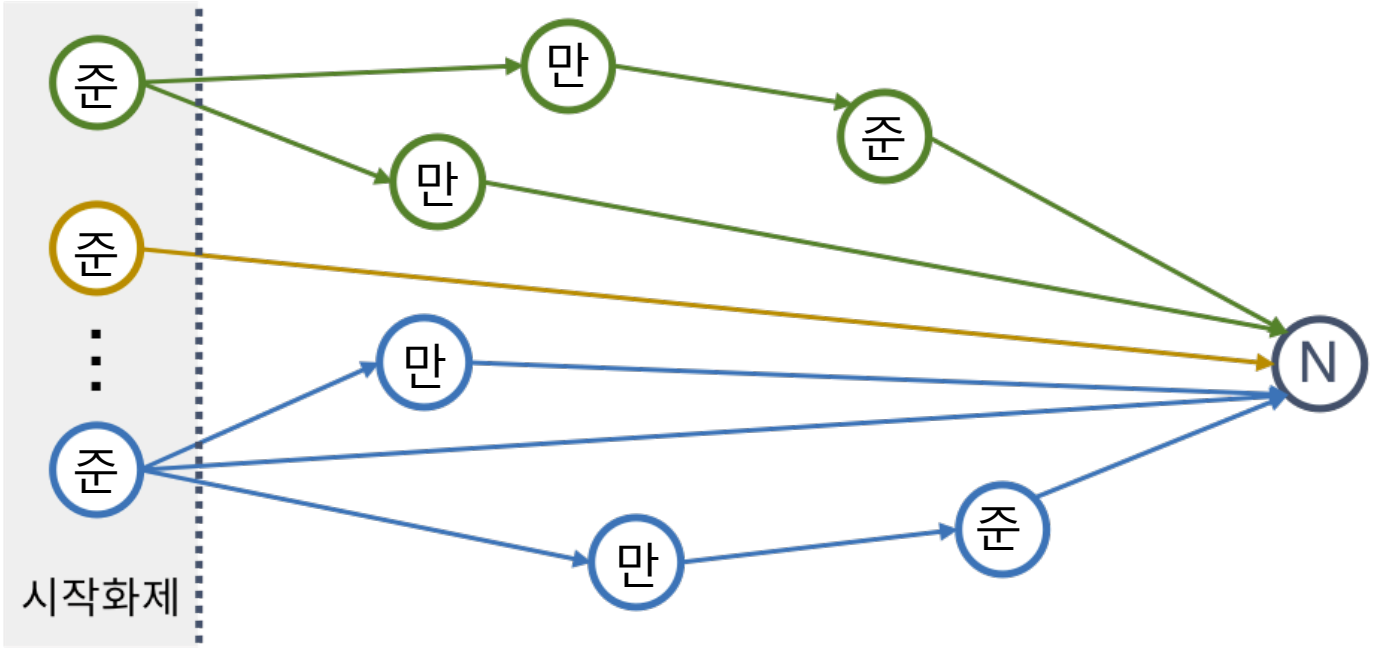
- 서브태스크1: 입력으로 주어지는 형태가 제한적이다.
  - 두 번째 화제 다음부터는 선택지가 하나밖에 없다.
  - 따라서 첫 번째 화제들에서 만영이가 고를 수 있는 두 번째 화제를 제한하는 횟수의 최솟값만 고려하면 된다.



1D. 그래서 팩 주냐?

출제자: 김현정

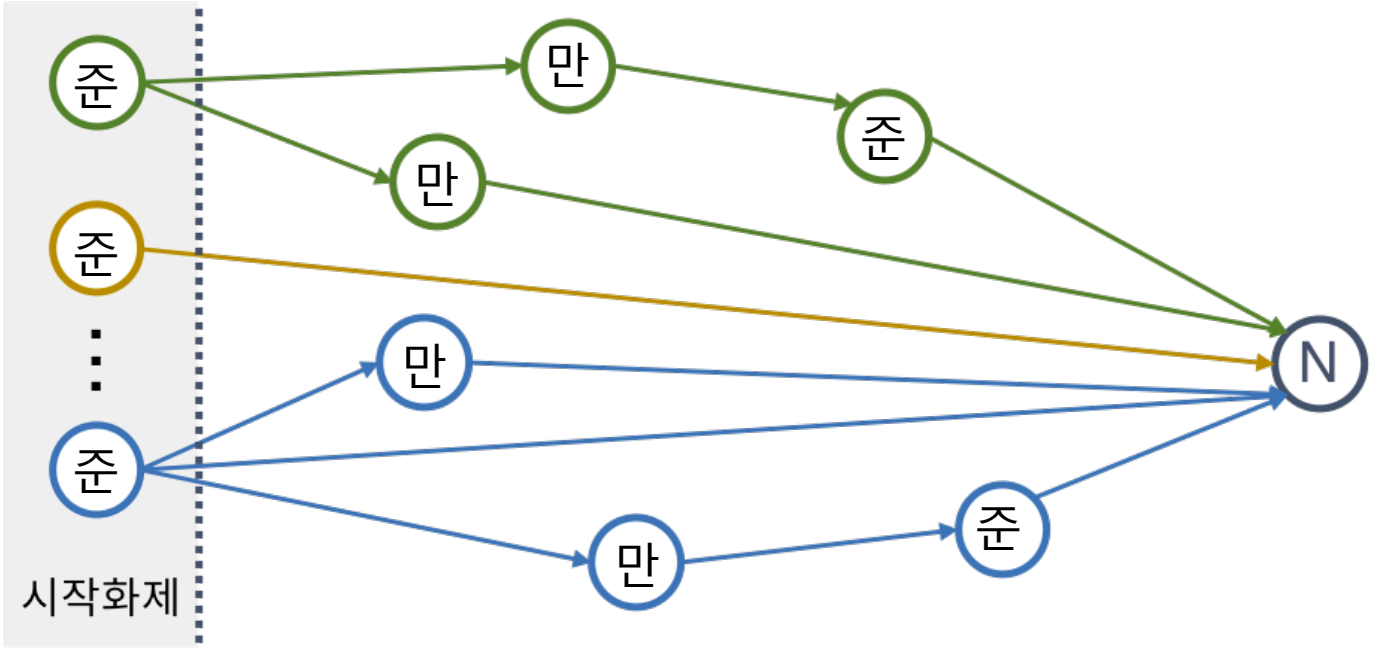
- 서브태스크1: 입력으로 주어지는 형태가 제한적이다.
  - 준표의 입장에서 만영이가 골라야하는 두 번째 화제는 홀수 경로 위의 화제여야 한다.
  - 준-만-준-만-준 처럼 화제의 개수가 홀수여야 준표가 N번 화제를 고를 수 있다.



1D. 그래서 팩 주냐?

출제자: 김현정

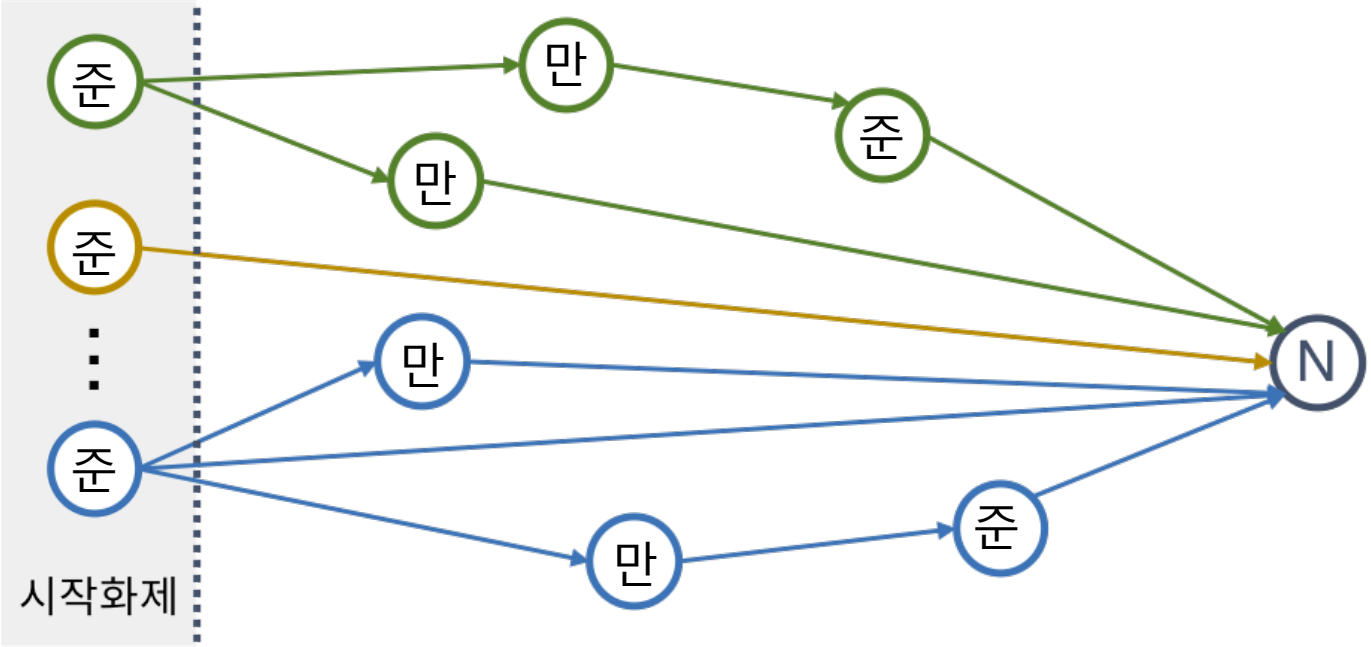
- 서브태스크1: 입력으로 주어지는 형태가 제한적이다.
  - 각 시작화제에서 뻗어나가는 경로 중 짝수경로와 홀수경로의 수를 센다.
  - 해당 시작화제를 준표가 골랐을때 만영이에게 정색하는 횟수는 홀수경로가 존재한다면 짝수경로의 개수만큼이다.



1D. 그래서 팩 주냐?

출제자: 김현정

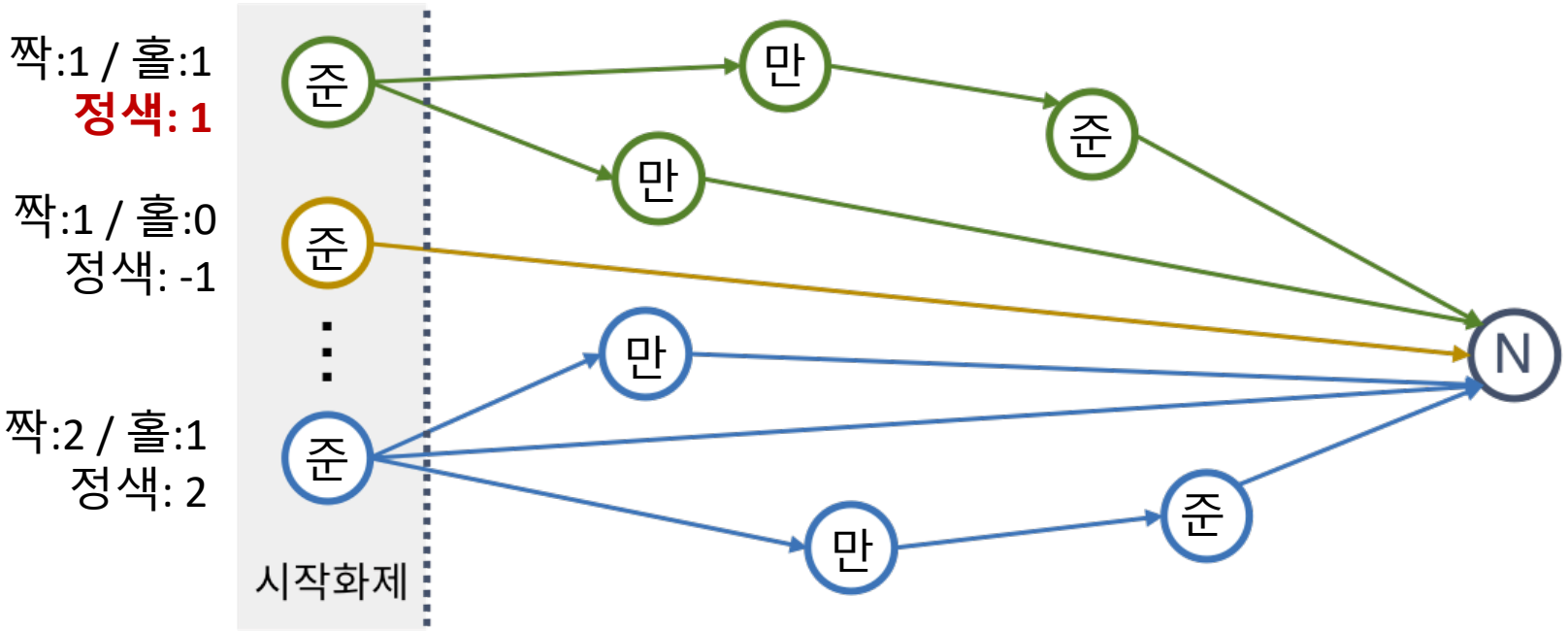
- 서브태스크1: 입력으로 주어지는 형태가 제한적이다.
  - 홀수경로가 존재하는 시작화제 중 짝수경로가 가장 적은 것이 답이 된다.
  - 존재하는 홀수경로가 없다면 -1이 답이 된다.



1D. 그래서 팩 주냐?

출제자: 김현정

- 서브태스크1: 입력으로 주어지는 형태가 제한적이다.
  - 홀수경로가 존재하는 시작화제 중 짝수경로가 가장 적은 것이 답이 된다.
  - 존재하는 홀수경로가 없다면 -1이 답이 된다.
  - 시간복잡도: O(E)



1D. 그래서 팩 주냐?

출제자: 김현정

- 서브태스크2:
  - 각 노드의 최소 정색 횟수를 구하기 위해서는 이후 노드들의 최소 정색 횟수를 알아야한다.
  - 시작화제부터 DFS 탐색을 하며 N번부터 리턴 될 때 아래 테이블을 채운다.
  - DP[vertex][turn]: vertex 화제를 turn(0: 준표 / 1: 만영)인 사람이 골랐을 때 준표가 이후 N번 화제를 고르기 위해 해야하는 최소 정색 횟수

1D. 그래서 팩 주냐?

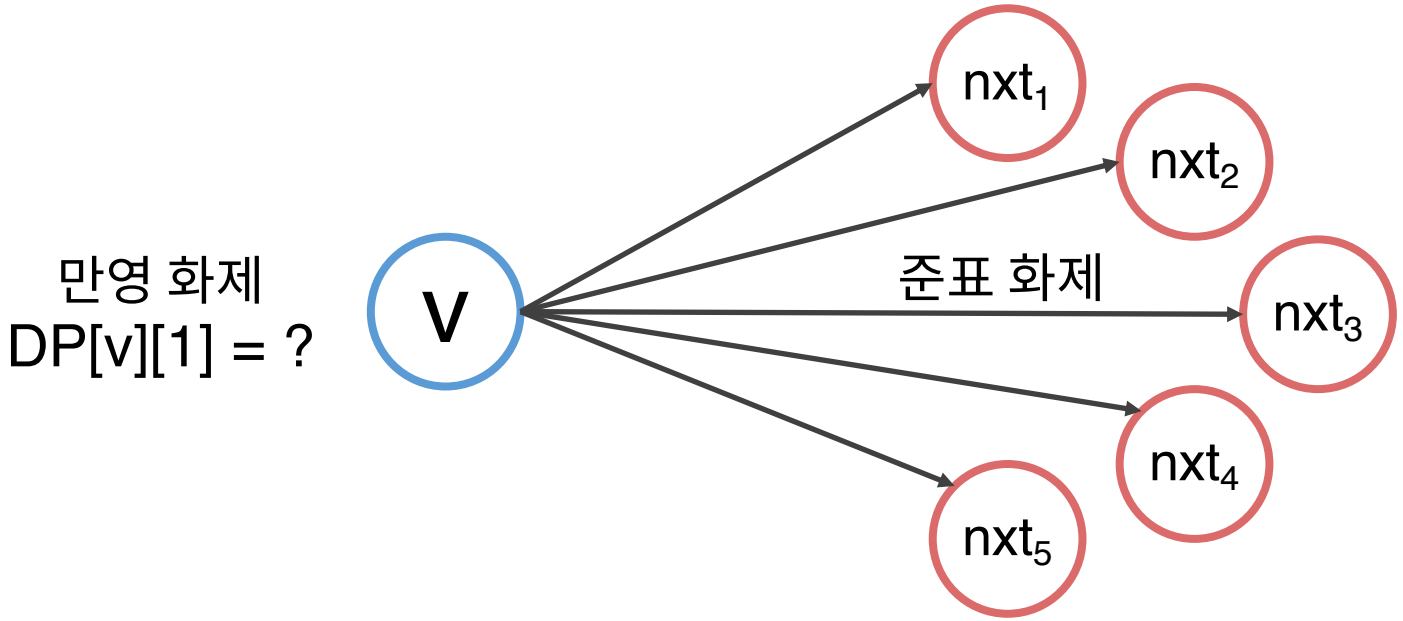
출제자: 김현정

- 서브태스크2:
  - DP[v][0]: v 화제를 준표가 골랐을 때 이후 필요한 최소 정색 횟수
  - DP[v][1]: v 화제를 만영이가 골랐을 때 이후 준표가 하게 될 최소 정색 횟수
  - DP[N][0] = 0, DP[N][1] = INF

1D. 그래서 팩 주냐?

출제자: 김현정

- 서브태스크2:
  - DP[v][0]: v 화제를 준표가 골랐을 때 이후 필요한 최소 정색 횟수
  - DP[v][1]: v 화제를 만영이가 골랐을 때 이후 준표가 하게 될 최소 정색 횟수



$DP[nxt_1][0] = 0$

$DP[nxt_2][0] = 2$

$DP[nxt_3][0] = 3$

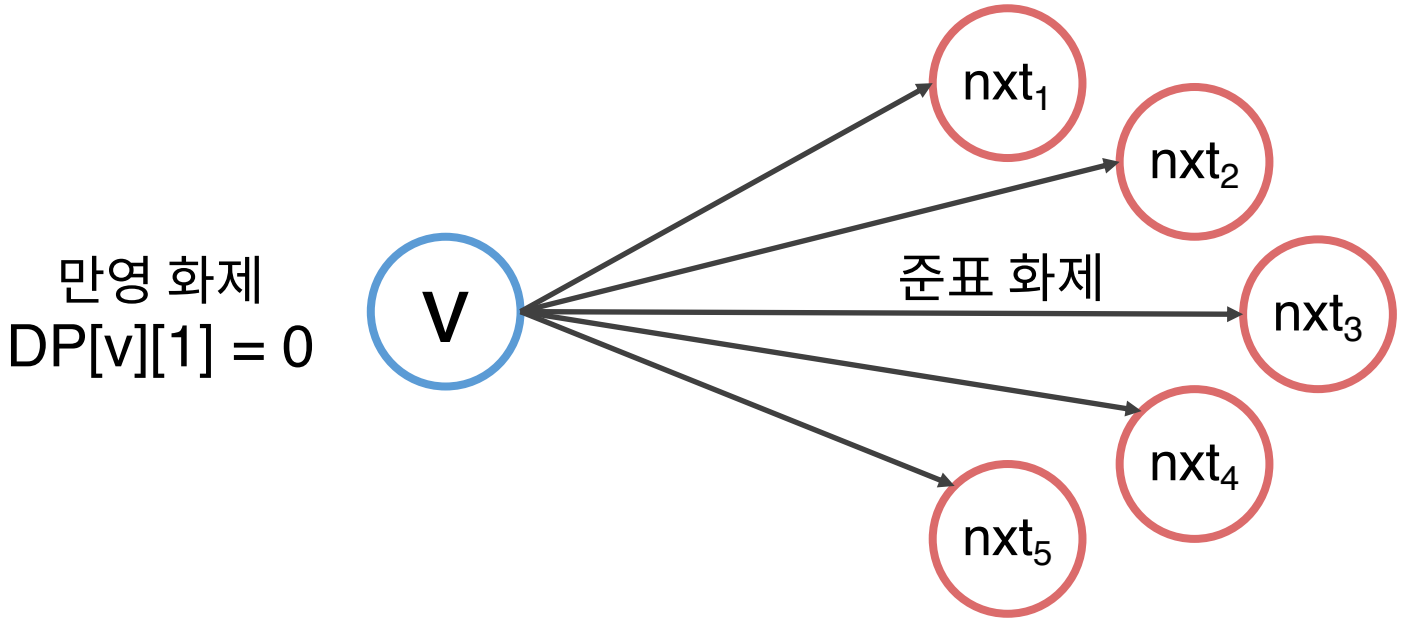
$DP[nxt_4][0] = 3$

$DP[nxt_5][0] = INF$

1D. 그래서 팩 주냐?

출제자: 김현정

- 서브태스크2:
  - v가 만영이의 화제이므로 다음 화제 선택권은 준표에게 있다.
  - 준표는 다음 화제 중 정색 횟수가 가장 적은 화제를 고른다.
  - DP[v][1] = $\min_{\mathbf{nxt} \in \text{next of } \mathbf{v}}$(DP[nxt][0])



$DP[nxt_1][0] = 0$

$DP[nxt_2][0] = 2$

$DP[nxt_3][0] = 3$

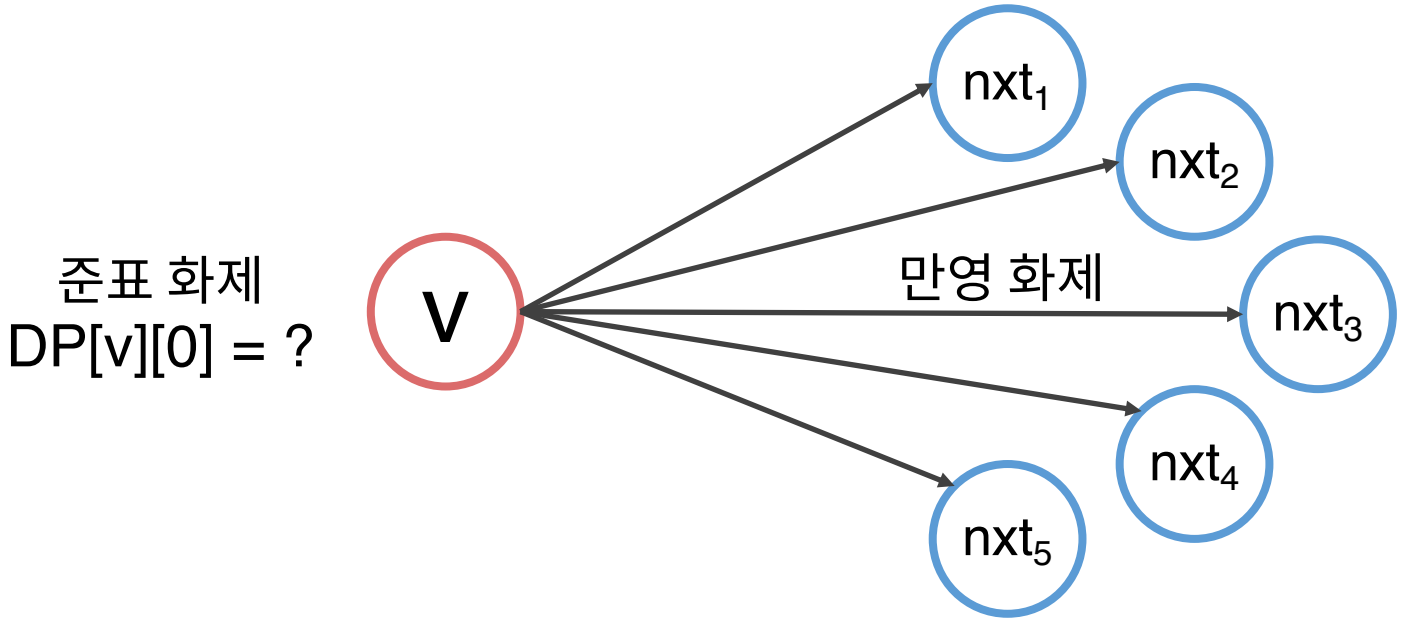
$DP[nxt_4][0] = 3$

$DP[nxt_5][0] = INF$

1D. 그래서 팩 주냐?

출제자: 김현정

- 서브태스크2:
  - DP[v][0]: v 화제를 준표가 골랐을 때 이후 필요한 최소 정색 횟수
  - DP[v][1]: v 화제를 만영이가 골랐을 때 이후 준표가 하게 될 최소 정색 횟수



$DP[nxt_1][1] = 0$

$DP[nxt_2][1] = 1$

$DP[nxt_3][1] = 1$

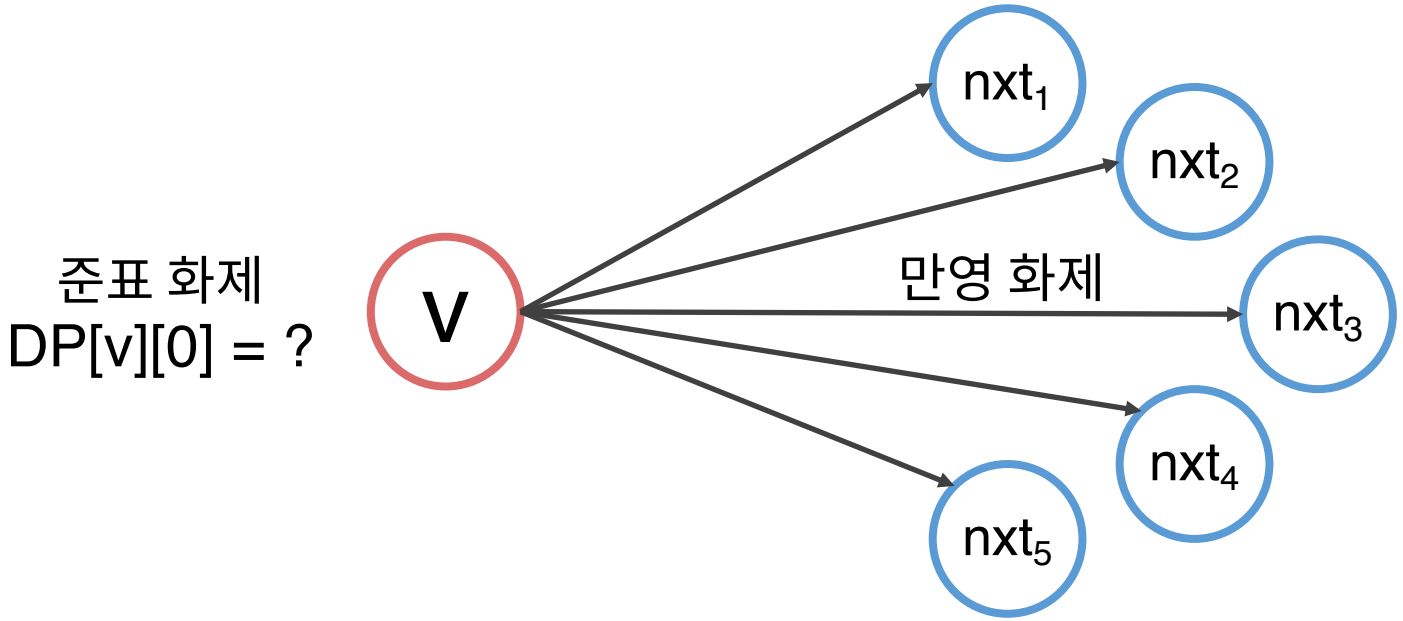
$DP[nxt_4][1] = 3$

$DP[nxt_5][1] = INF$

1D. 그래서 팩 주냐?

출제자: 김현정

- 서브태스크2:
  - v는 준표가 고른 화제이므로 다음 화제는 만영이가 고른다.
  - 만영이는 준표가 가장 많이 정색하는 방향으로 화제를 고른다.
  - 준표는 자신이 가장 적게 정색하는 방향으로 만영이의 선택을 제한한다.



$DP[nxt_1][1] = 0$

$DP[nxt_2][1] = 1$

$DP[nxt_3][1] = 1$

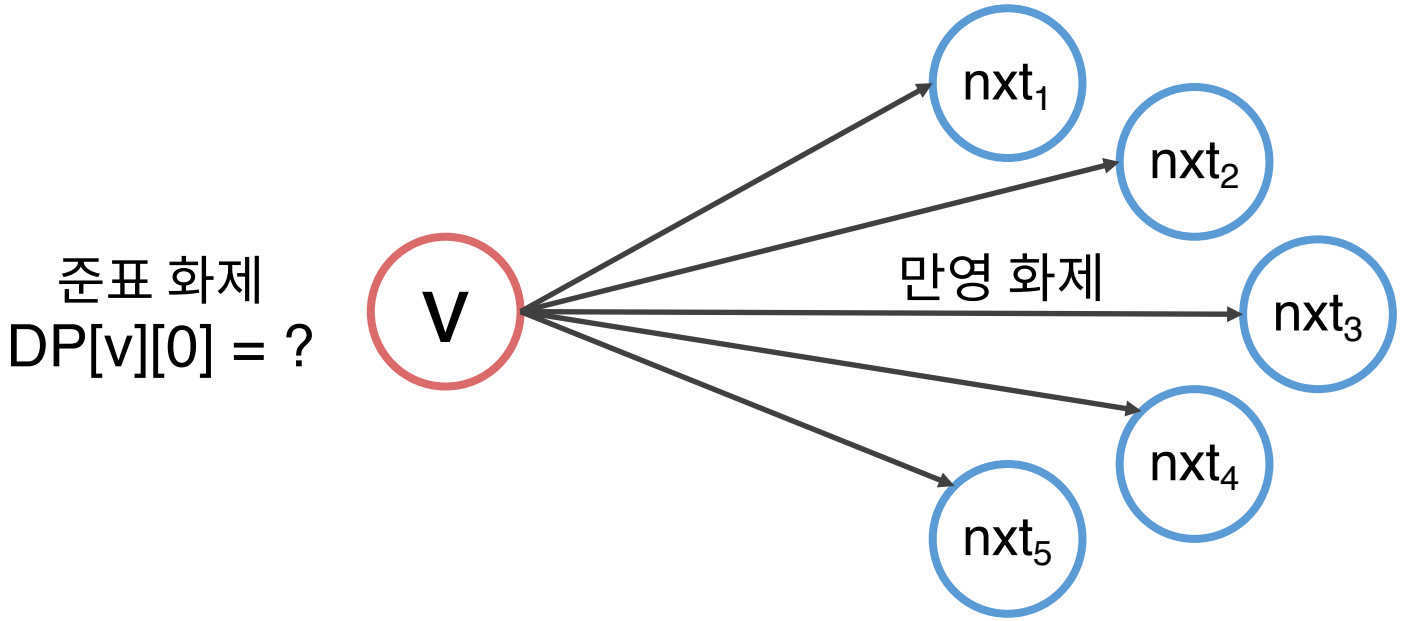
$DP[nxt_4][1] = 3$

$DP[nxt_5][1] = INF$

1D. 그래서 팩 주냐?

출제자: 김현정

- 서브태스크2:
  - 만영이는 정색 횟수가 큰 순서부터 화제를 고른다.
  - 준표 정색 0번 + 만영이가 다음 화제로 nxt_5를 선택
  - 준표 정색 1번 + 만영이가 다음 화제로 nxt_4를 선택



$DP[nxt_1][1] = 0$

$DP[nxt_2][1] = 1$

$DP[nxt_3][1] = 1$

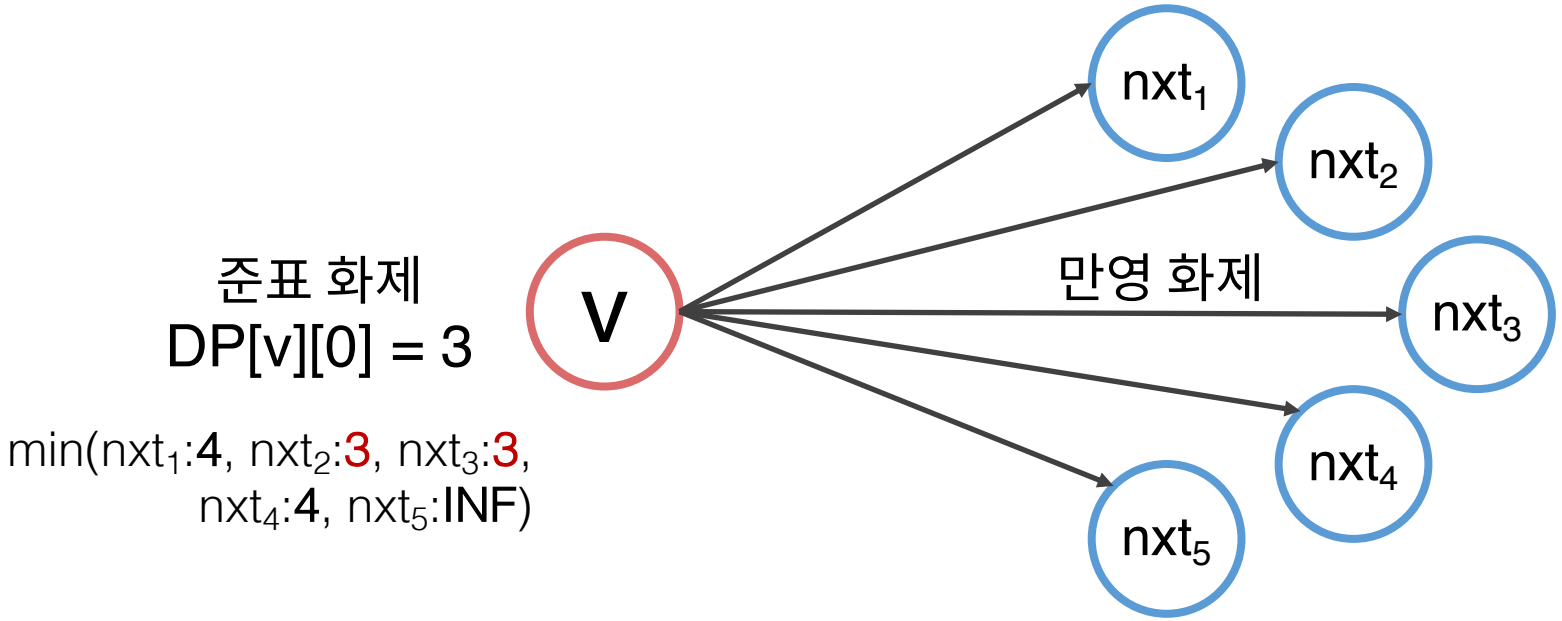
$DP[nxt_4][1] = 3$

$DP[nxt_5][1] = INF$

1D. 그래서 팩 주냐?

출제자: 김현정

- 서브태스크2:
  - 준표는 정색을 2회 하면 만영이가 다음 화제로 nxt_2나 nxt_3를 고르게 할 수 있다.
  - 준표는 정색을 4회 해야 만영이가 다음 화제로 nxt_1을 고르게 할 수 있다.
  - $DP[v][0] = \min_{\mathbf{nxt} \in \text{next of } \mathbf{v}}(DP[nxt][1] + \text{count}_{\mathbf{u} \in \text{next of } \mathbf{v}}(DP[nxt][1] < DP[u][1]))$



1D. 그래서 팩 주냐?

출제자: 김현정

- 서브태스크2:
  - DP[v][0]: v 화제를 준표가 골랐을 때 이후 필요한 최소 정색 횟수
  - DP[v][1]: v 화제를 만영이가 골랐을 때 이후 준표가 하게 될 최소 정색 횟수
  - DP[N][0] = 0, DP[N][1] = INF

- $DP[v][1] = \min_{\mathbf{nxt} \in \text{next of } \mathbf{v}}(DP[nxt][0])$
- $DP[v][0] = \min_{\mathbf{nxt} \in \text{next of } \mathbf{v}}(DP[nxt][1] + \text{count}_{\mathbf{u} \in \text{next of } \mathbf{v}}(DP[nxt][1] < DP[u][1]))$

- 시간복잡도: O(ElogE)

1E. 아름다운 만영로

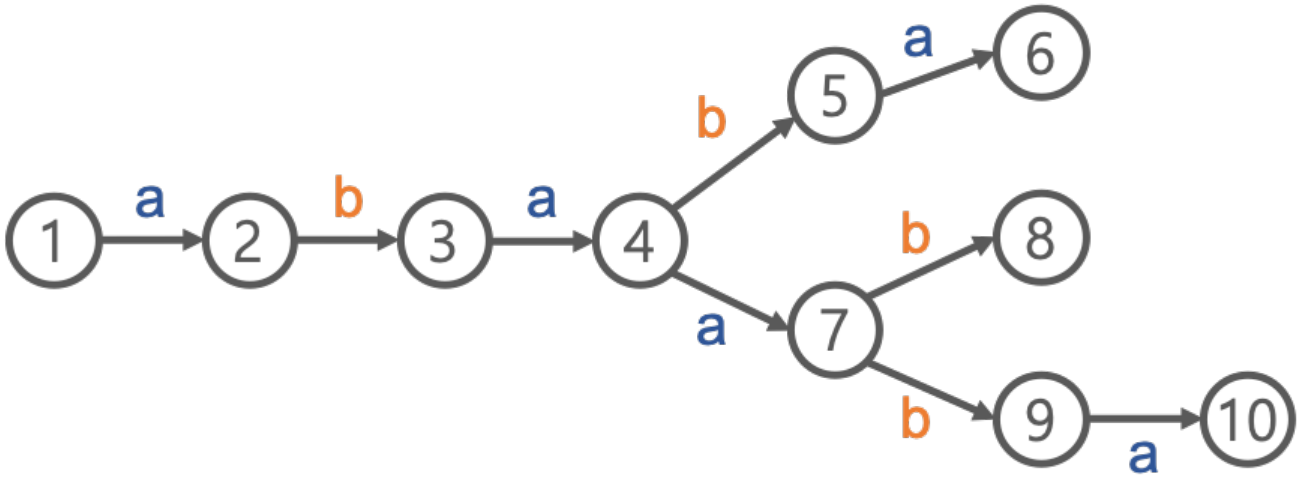
출제자: 한만영

- Sub1
- 루트가 있는 트리
- 언제나 부모에서 자식 방향으로만 진행
- 특정한 패턴의 경로 탐색

1E. 아름다운 만영로

출제자: 한만영

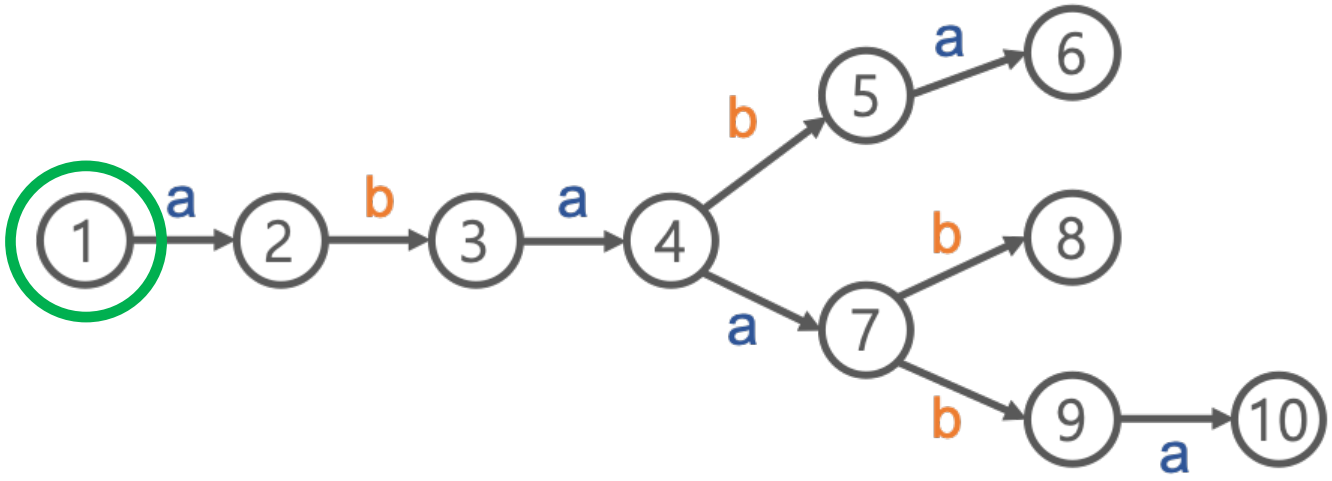
- Sub1
- 루트에서부터 재귀적으로 탐색을 진행합니다.
- 여태까지 지나온 꽃을 기억하는 배열을 만듭니다.



1E. 아름다운 만영로

출제자: 한만영

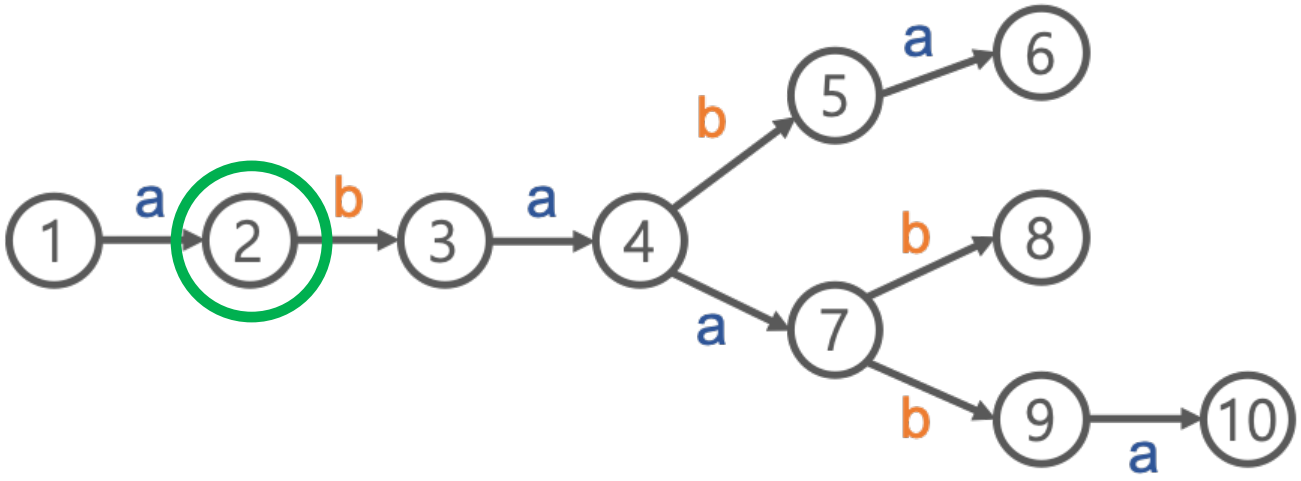
- Sub1
- 지나온 꽃: []
- 현재 위치: 1



1E. 아름다운 만영로

출제자: 한만영

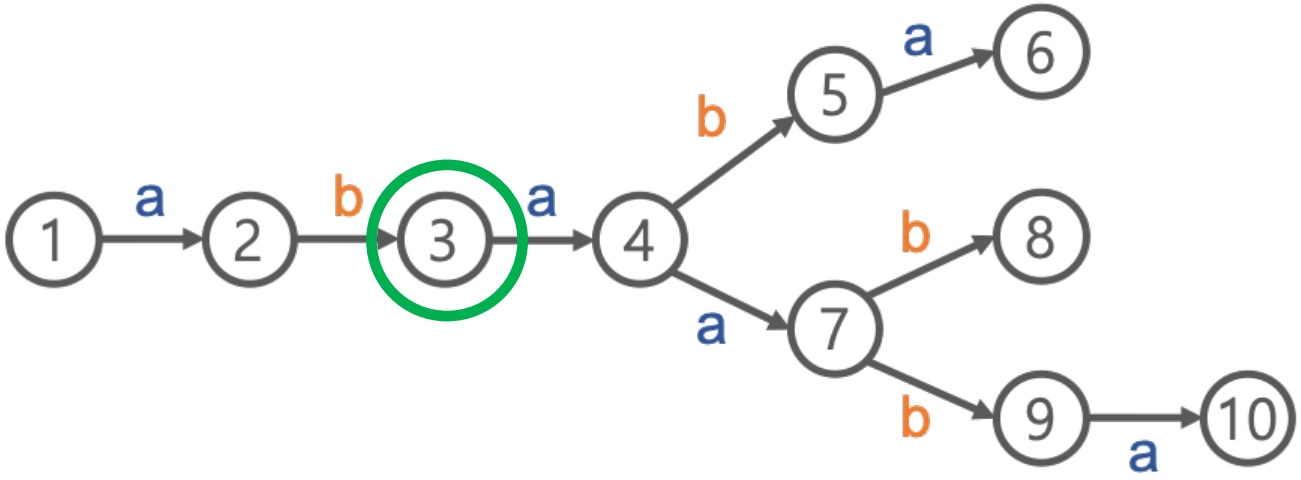
- Sub1
- 지나온 꽃: [a]
- 현재 위치: 2



1E. 아름다운 만영로

출제자: 한만영

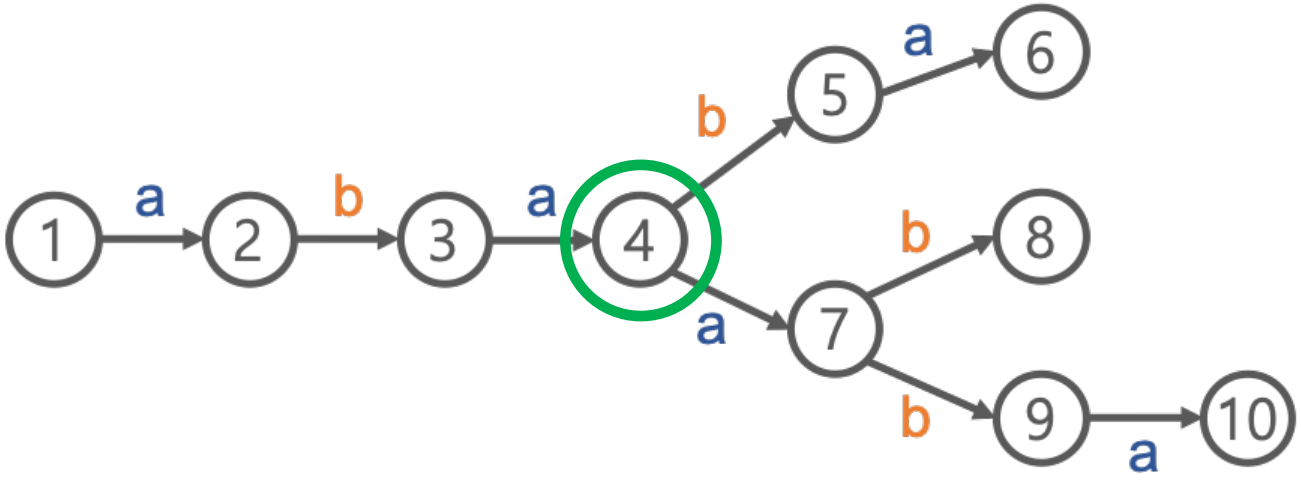
- Sub1
- 지나온 꽃: [a, b]
- 현재 위치: 3
- [a, b] 가 패턴 “ab”와 같으므로 1개의 경로를 확인할 수 있다.



1E. 아름다운 만영로

출제자: 한만영

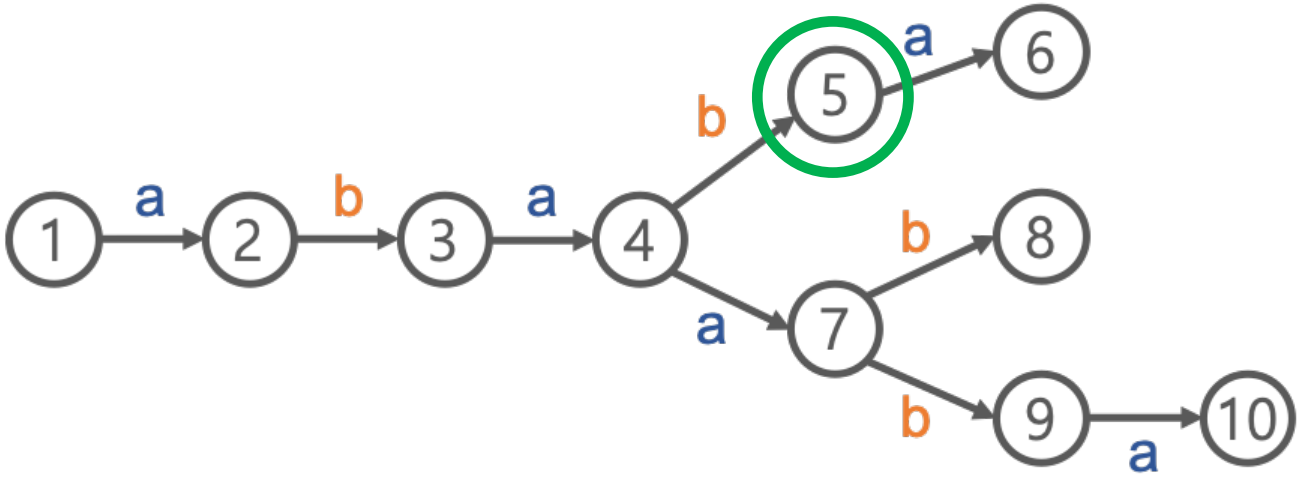
- Sub1
- 지나온 꽃: [a, b, a]
- 현재 위치: 4
- […, b, a] 가 “ab”와 일치 하지 않음으로 경로를 발현하지 못했다.



1E. 아름다운 만영로

출제자: 한만영

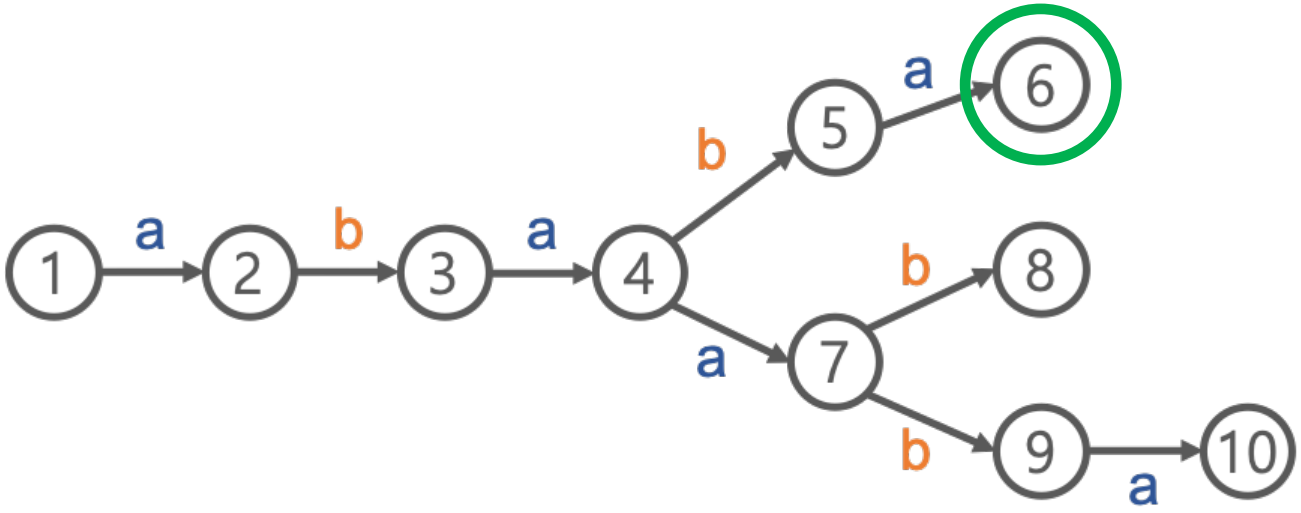
- Sub1
- 지나온 꽃: [a, b, a, b]
- 현재 위치: 5
- […, a, b] 가 패턴 “ab”와 같으므로 1개의 경로를 확인할 수 있다.



1E. 아름다운 만영로

출제자: 한만영

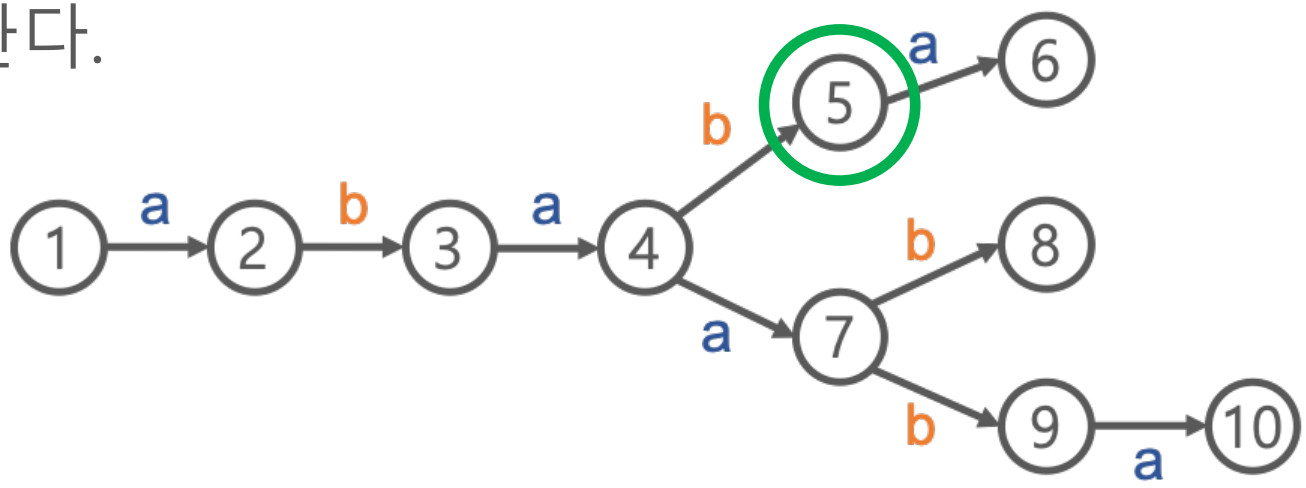
- Sub1
- 지나온 꽃: [a, b, a, b, a]
- 현재 위치: 6
- […, b, a] 가 “ab”와 일치 하지 않음으로 경로를 발현하지 못했다.



1E. 아름다운 만영로

출제자: 한만영

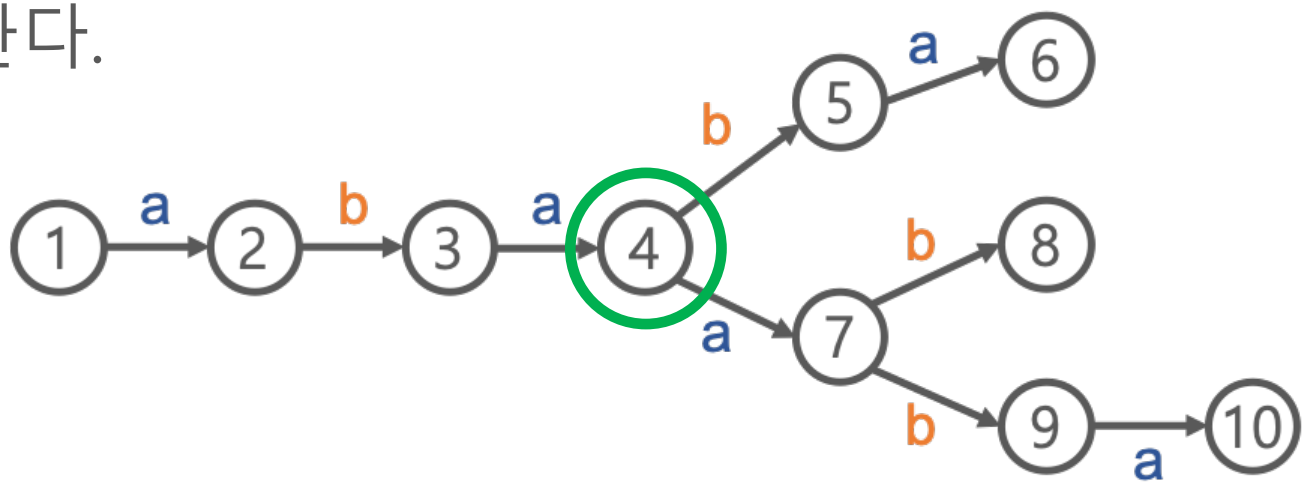
- Sub1
- 지나온 꽃: [a, b, a, b]
- 현재 위치: 5
- 더이상 진행 할 수 없으므로 복귀한다. 복귀할때마다 배열에서 꽃을 하나씩 제거한다.



1E. 아름다운 만영로

출제자: 한만영

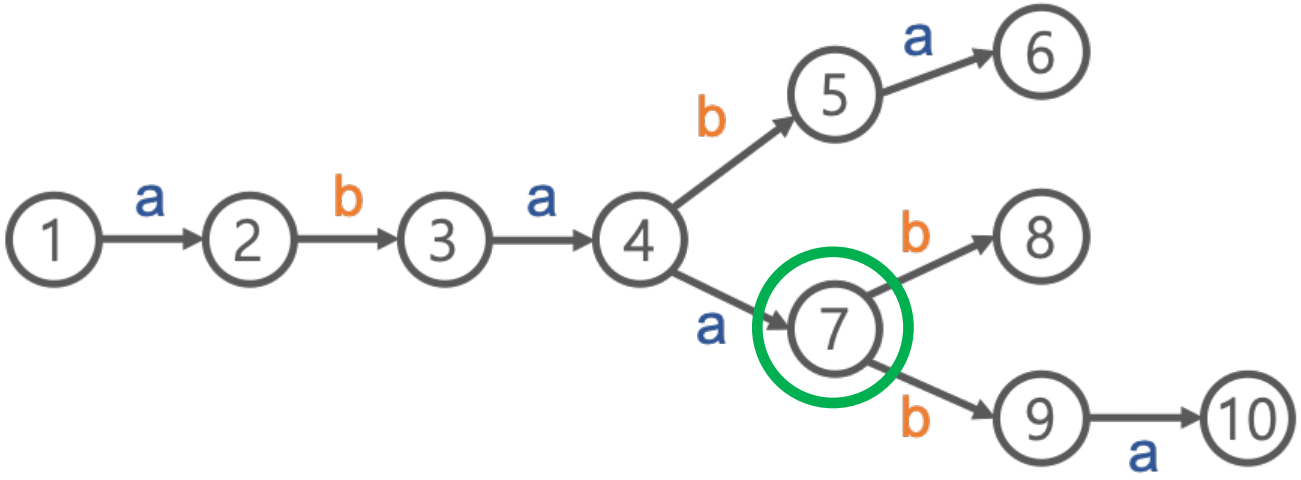
- Sub1
- 지나온 꽃: [a, b, a]
- 현재 위치: 4
- 더이상 진행 할 수 없으므로 복귀한다. 복귀할때마다 배열에서 꽃을 하나씩 제거한다.



1E. 아름다운 만영로

출제자: 한만영

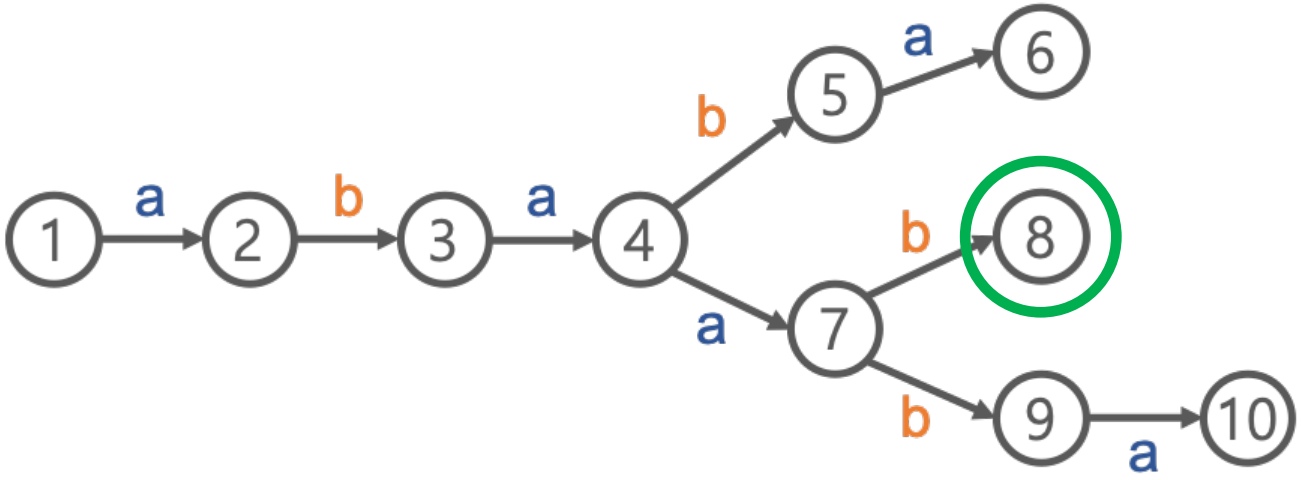
- Sub1
- 지나온 꽃: [a, b, a, a]
- 현재 위치: 7
- […, a, a] 가 “ab”와 일치 하지 않음으로 경로를 발현하지 못했다.



1E. 아름다운 만영로

출제자: 한만영

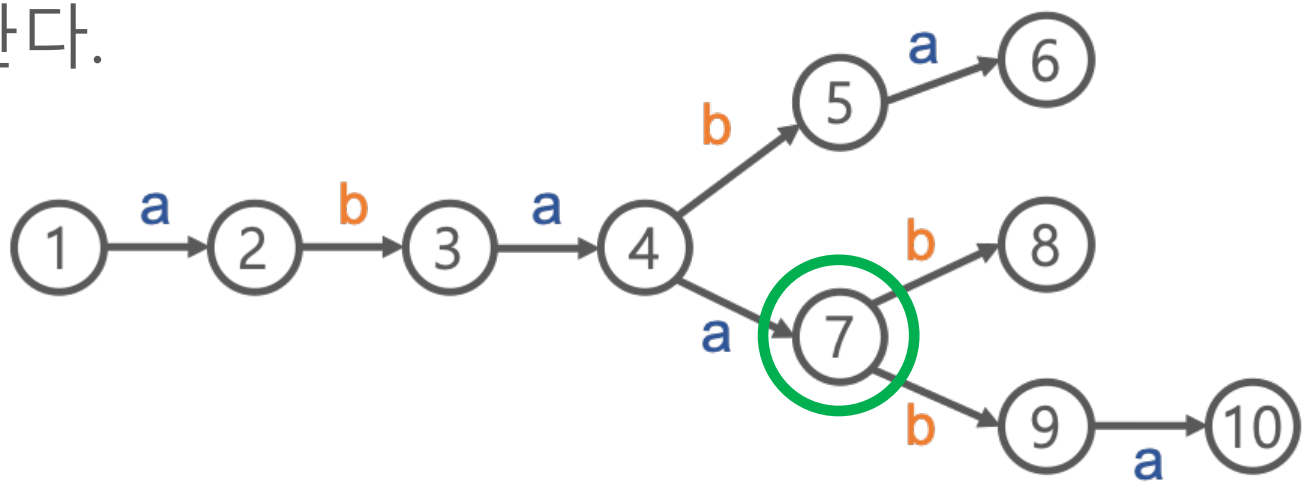
- Sub1
- 지나온 꽃: [a, b, a, a, b]
- 현재 위치: 8
- […, a, b] 가 패턴 “ab”와 같으므로 1개의 경로를 확인할 수 있다.



1E. 아름다운 만영로

출제자: 한만영

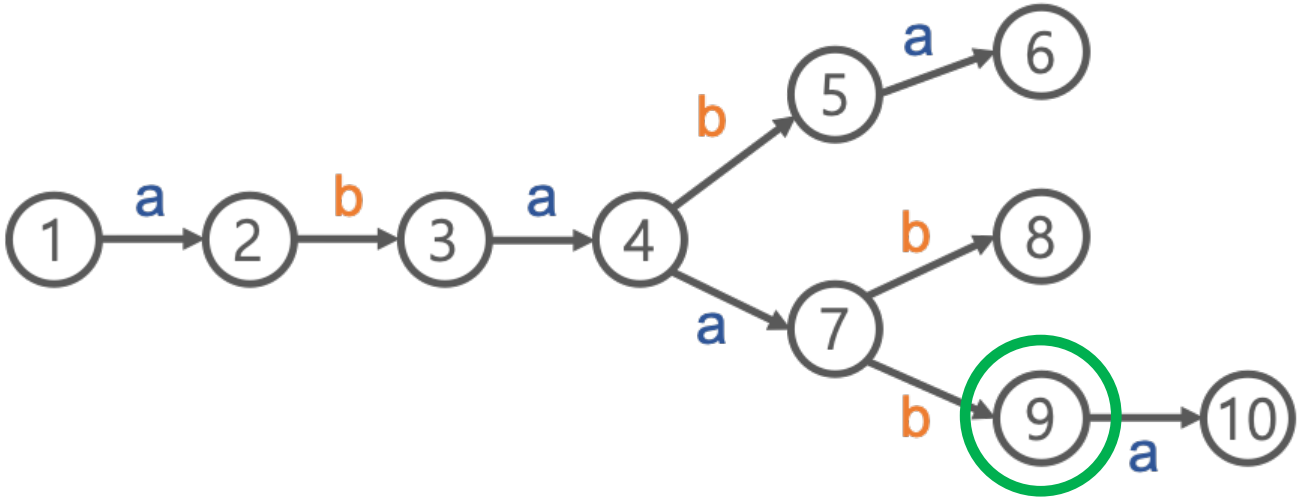
- Sub1
- 지나온 꽃: [a, b, a, a]
- 현재 위치: 7
- 더이상 진행 할 수 없으므로 복귀한다. 복귀할때마다 배열에서 꽃을 하나씩 제거한다.



1E. 아름다운 만영로

출제자: 한만영

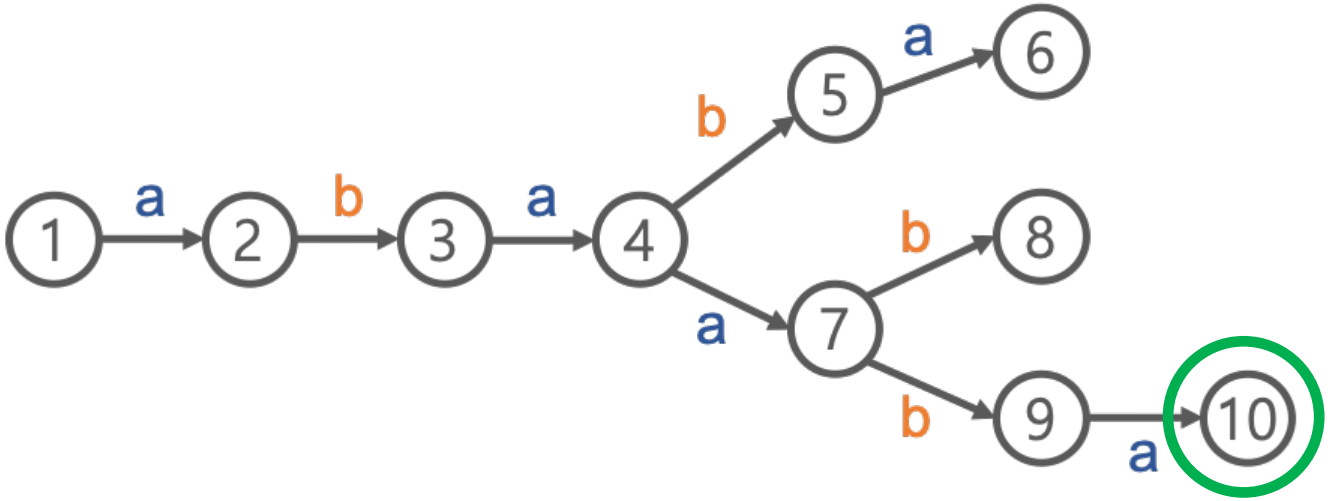
- Sub1
- 지나온 꽃: [a, b, a, a, b]
- 현재 위치: 9
- […, a, b] 가 패턴 “ab”와 같으므로 1개의 경로를 확인할 수 있다.



1E. 아름다운 만영로

출제자: 한만영

- Sub1
- 지나온 꽃: [a, b, a, a, b, a]
- 현재 위치: 10
- […, b, a] 가 “ab”와 일치 하지 않음으로 경로를 발현하지 못했다.



1E. 아름다운 만영로

출제자: 한만영

- Sub1
- 시간 복잡도
  - 각 노드로 방문함: N
  - 방문할 때마다 문자열 비교를 실행: |S|
  - O(N x |S|)

- 문자열을 비교하는 시간을 줄일 수 있다면?

1E. 아름다운 만영로

출제자: 한만영

- Sub2
- Rabin-Karp알고리즘
  - 매번 꽃을 배열에 넣을 때 마다 해쉬값을 만들어 낸다
    - Ex: “abab” => 4852
  - 해쉬가 다르다면 두 문자열이 같을 가능성(해쉬 충돌)은 매우 낮다
  - 해쉬의 비교는 정수비교임으로 매우 빠르다

1E. 아름다운 만영로

출제자: 한만영

- Sub2
- Rabin-Karp알고리즘
  - 매번 꽃을 배열에 넣을 때 마다 해쉬값을 만들어 낸다
    - O(1)
  - 해쉬 비교(정수 비교)
    - O(1)

1E. 아름다운 만영로

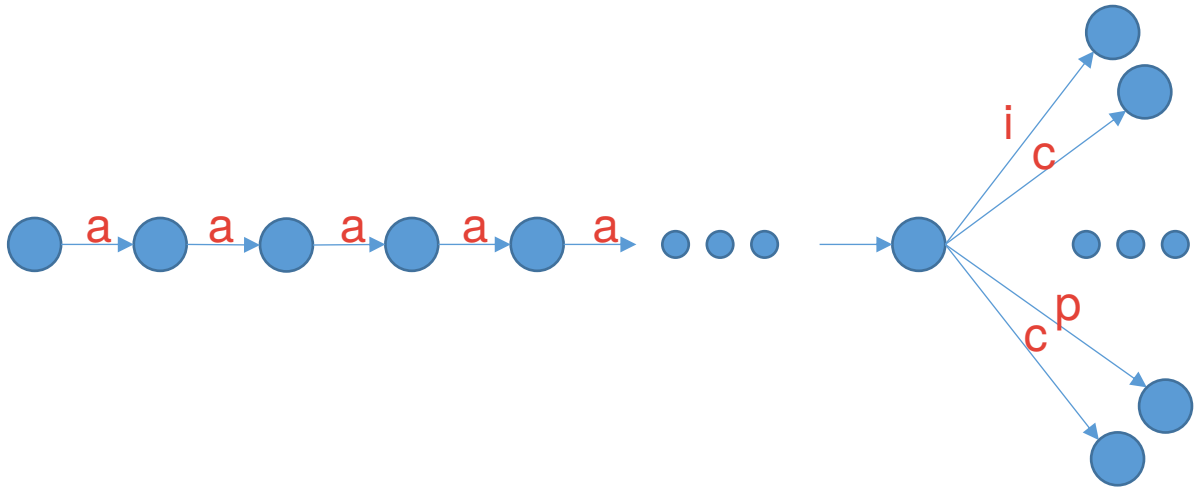
출제자: 한만영

- Sub2
- 시간 복잡도
  - 각 노드로 방문함: N
  - 방문할 때마다 문자열 비교를 해쉬로 실행: 1
  - O(N)

1E. 아름다운 만영로

출제자: 한만영

- Sub2
- KMP로 문자열 비교를 한다면 어떨까?



1F/2E. 아싸 너!

출제자: 홍준표

- 서브태스크1:
  - N 이 7일 때 상태의 개수는 ?
  - … 7! = 5040
  - 5040 개의 상태에 대한 BFS로 가장 짧은 swap 횟수를 구할 수 있다
- 상태의 정의는 ?
  - 벡터의 해싱
  - 7자리의 수로 표현 가능 (1,3,2,5,4,7,6 -> 1325476 < 1e7)

- 시간복잡도 : O(N x N!)

1F/2E. 아싸 너!

출제자: 홍준표

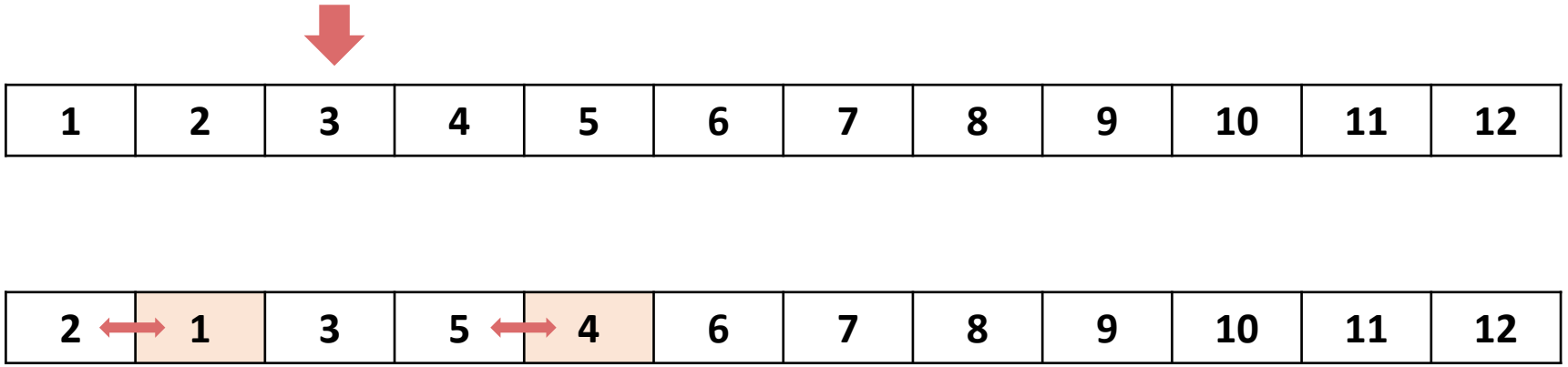
- 서브태스크1:
  - 사실은.. 될때까지 랜덤하게 시행해도 100만회를 잘 넘지 않음

- 시간복잡도 : O(1e6)

1F/2E. 아싸 너!

출제자: 홍준표

- 서브태스크2:

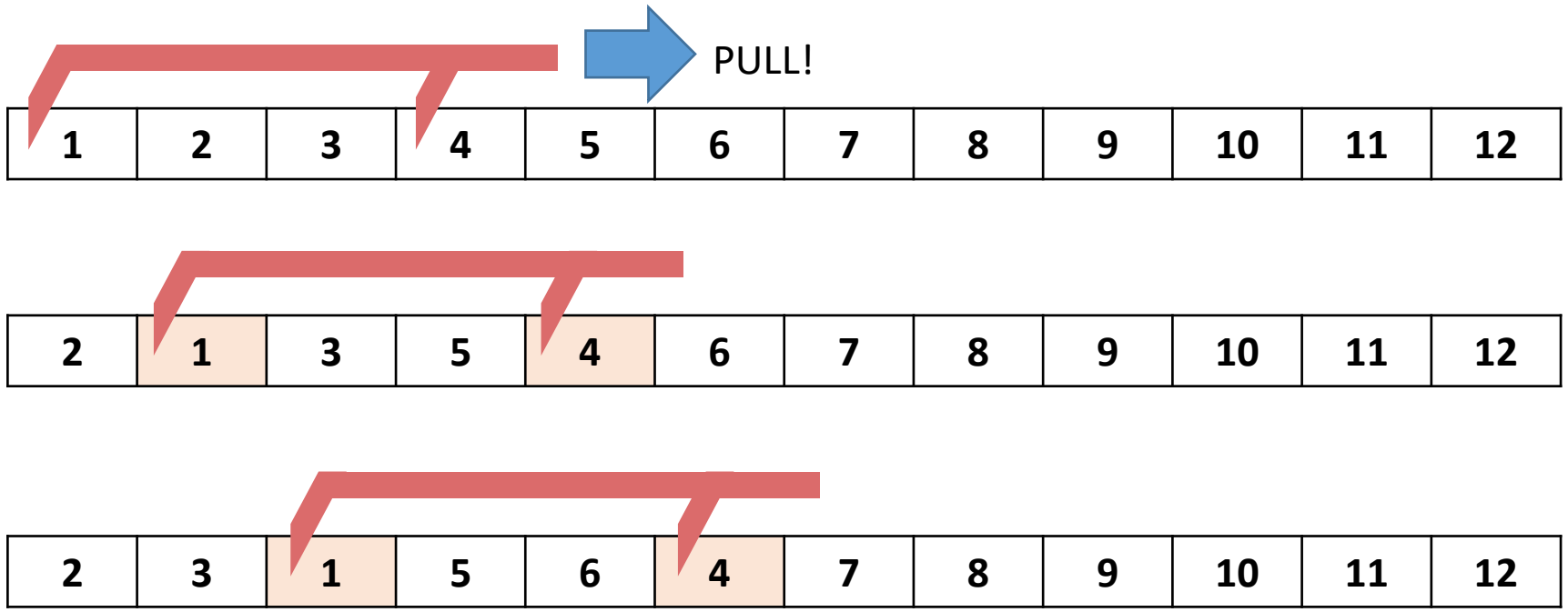


- 연산을 재정의 해보자

1F/2E. 아싸 너!

출제자: 홍준표

- 서브태스크2:



- 원 위에서 한바퀴 돌리면 두 칸 돌린 원순열이 됨.
  - N이 홀수일 때는 언제나 가능,
  - N이 짝수일 때는 짝수 칸 돌린 원순열을 만들 수 있음.

1F/2E. 아싸 너!

출제자: 홍준표

- N이 짝수일 때 홀수 칸 돌린 원순열은 만들 수 없을까 ?

1	2	3	4	5	6	7	8

- Swap : 인접한 배열을 교환
- 게임에서는 한 턴에 Swap 연산이 두 번 발생한다. 언제나 짝수번

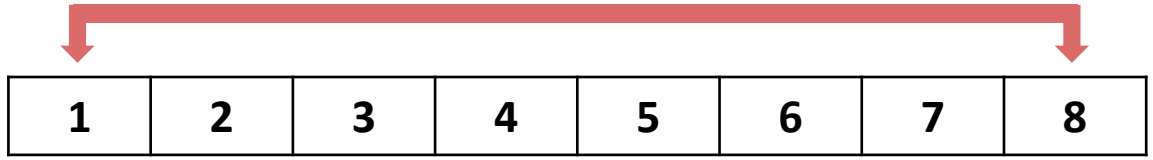
2	3	4	5	6	7	8	1

- 로 만들기 위해선 N-1 번의 Swap 이 필요하다. N-1 : 홀수

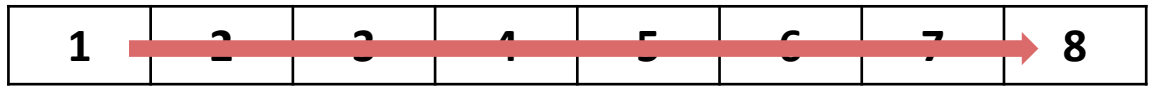
1F/2E. 아싸 너!

출제자: 홍준표

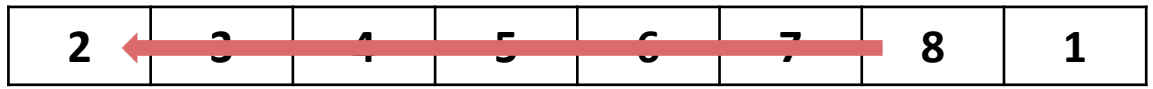
- 잠깐! 배열이 아니라 원이잖아요



- 이것만 할 수 있으면 원이라고 생각 할 수 있음



N-1 번



N-2 번

총 2N – 3번 : 홀수

1F/2E. 아싸 너!

출제자: 홍준표

- 마지막으로 필요한 증명 …

1	2	3	4	5	6	7	8

- 같은 상태로 돌아오기 위해 필요한 Swap 연산의 횟수는 ?
- 제자리에 2번 하면 돌아오는 것은 자명 : 짝수번
- 홀수번 Swap 해서 돌아오는 것이 가능할까 ?
  - 상태에 대해 이분 그래프가 만들어짐
    - 여기선 증명을 생략..
- 시간복잡도 : O(N^2)

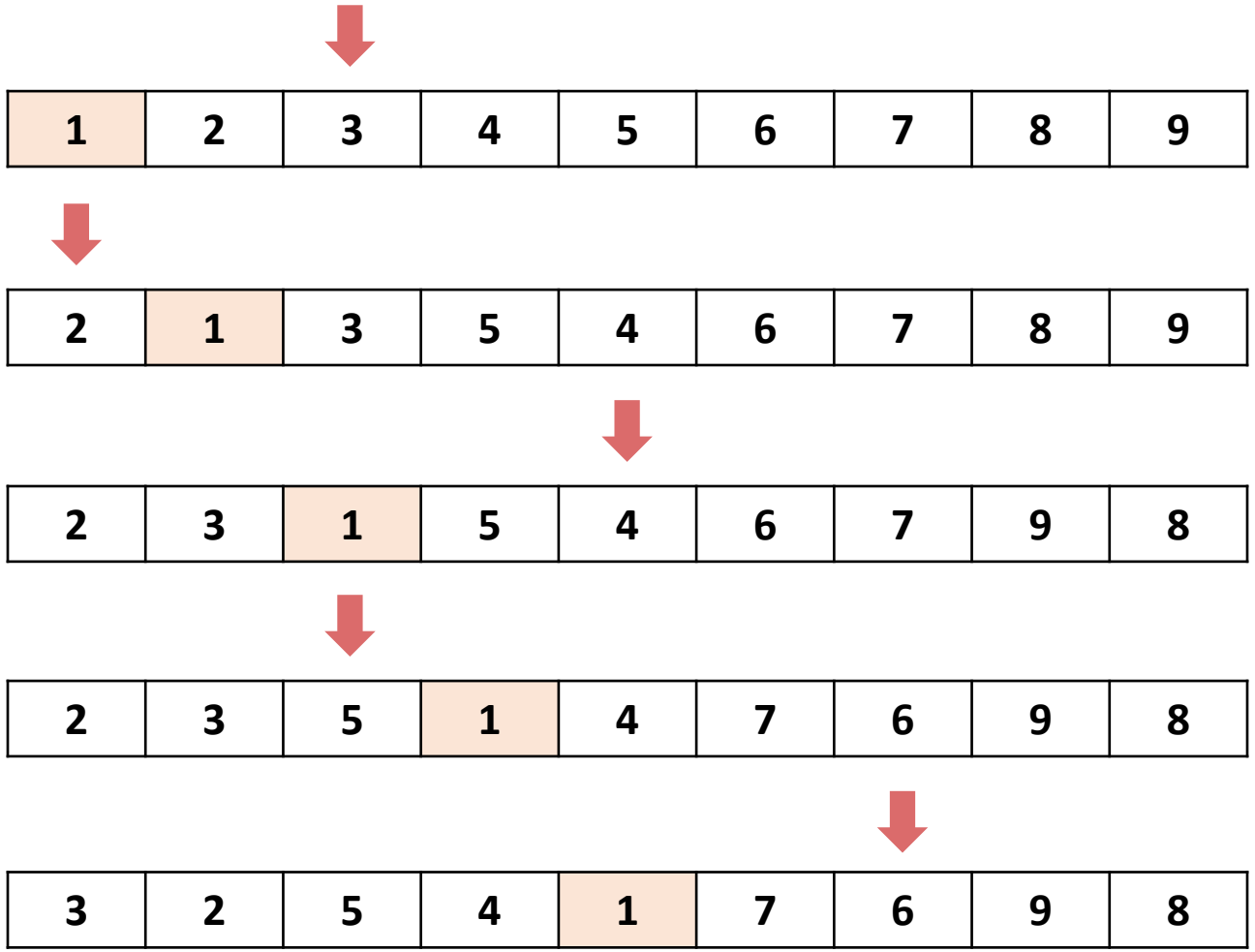
1F/2E. 아싸 너!

출제자: 홍준표

- 서브태스크3:
  - 반절로 줄여야함
  - 연산을 한바퀴 돌려서 두칸 돌아간 원순열을 만들 수 있었음
  - 한바퀴 돌려서 한칸 돌아간 원순열은 만들 수 없을까?
  - 하나만 정해서 옮겨보자

1F/2E. 아싸 너!

출제자: 홍준표



- 이렇게 하나만 보며 연산을 계속하면 한칸만을 돌릴 수 있음.
- 시간복잡도 : (N^2)

2F. 문자열 장식

출제자: 한만영

- 문자열에서 정해진 모든 패턴이 등장하는 부분문자열을 찾는 문제입니다.

2F. 문자열 장식

출제자: 한만영

- Sub1
- 우선 문자열이 등장하는 위치를 모두 찾습니다.
  - 앞에서부터 하나씩 옮기면서 같은지 확인합니다.
- S = “bananabananabanana”
- P = {“an”, “anab”, “ab”, “nana”}
  - bananabananabanana
  - bananabananabanana
  - bananabananabanana
  - bananabananabanana

2F. 문자열 장식

출제자: 한만영

- Sub1
- bananabananabanana
- bananababananababanana
- bananabananabanana
- bananabananabanana

- 맨 앞에 구간을 잡습니다.
  - [ ] b{a{n{an}{a}b}}{a{n{an}{a}b}}{a{nan}a}

2F. 문자열 장식

출제자: 한만영

- Sub1
- 맨 앞에 구간을 잡습니다.
  - [ ] b{a{n{an}{a}b}}{a{n{an}{a}b}}{a{nan}a}
- 모든 패턴을 모을 때 까지 뒤로 확장합니다.
  - [b{a{n{an}{a}b}}] {a{n{an}{a}b}}{a{nan}a} : (7자)
- 모든 패턴은 내부에 유지하면서 앞을 축소합니다.
  - b [{a{n{an}{a}b}}] {a{n{an}{a}b}}{a{nan}a} : (6자)

2F. 문자열 장식

출제자: 한만영

- Sub1
- 모든 패턴은 내부에 유지하면서 앞을 축소합니다.
  - b [{a{n{an}{a}b}}] {a{n{an}{a}b}}{a{nan}a} : (6자)
- 뒤로 1칸 확장합니다.
  - b [{a{n{an}{a}b}}{a] {n{an}{a}b}}{a{nan}a} : (7자)
- 모든 패턴을 내부에 유지하면서 앞을 축소합니다.
  - b [{a{n{an}{a}b}}{a] {n{an}{a}b}}{a{nan}a} : (7자)
- 반복합니다.

2F. 문자열 장식

출제자: 한만영

- Sub1
- 반복합니다.
  - b [{a{n{an}{a}b}}{a] {n{an}{a}b}}{a{nan}a} : (7자)
  - b [{a{n{an}{a}b}}{a{n] {an}{a}b}}{a{nan}a} : (8자)
  - …
  - b [{a{n{an}{a}b}}{a{n{an] }{a}b}} {a{nan}a} : (12자)
  - b{a{n{an}{a}b}} [{a{n{an}{a}b}}] {a{nan}a} : (6자)
  - …
  - b{a{n{an}{a}b}} [{a{n{an}{a}b}}{a{nan}a}] : (11자)

2F. 문자열 장식

출제자: 한만영

- Sub1
- 구간이 가장 짧았던 순간을 기록하면 답이 6임을 알 수 있습니다.

- 시간복잡도
  - 패턴 위치 확인
    - O(|S| x sum(|P_i|))
  - 모든 패턴을 포함하는 최단 부분문자열 탐색
    - O(|S| x N)
- 패턴 위치 확인에 드는 시간이 대부분을 차지!!

2F. 문자열 장식

출제자: 한만영

- Sub2
- 패턴의 등장 위치를 빠르게 확인해야한다
  - KMP 알고리즘, Rabin-Karp 알고리즘, Z 알고리즘, Aho-Corasick 알고리즘

- 시간복잡도
  - 패턴 위치 확인
    - O(|S| + sum($|P_i|$))
  - 모든 패턴을 포함하는 최단 부분문자열 탐색
    - O(|S| x N)

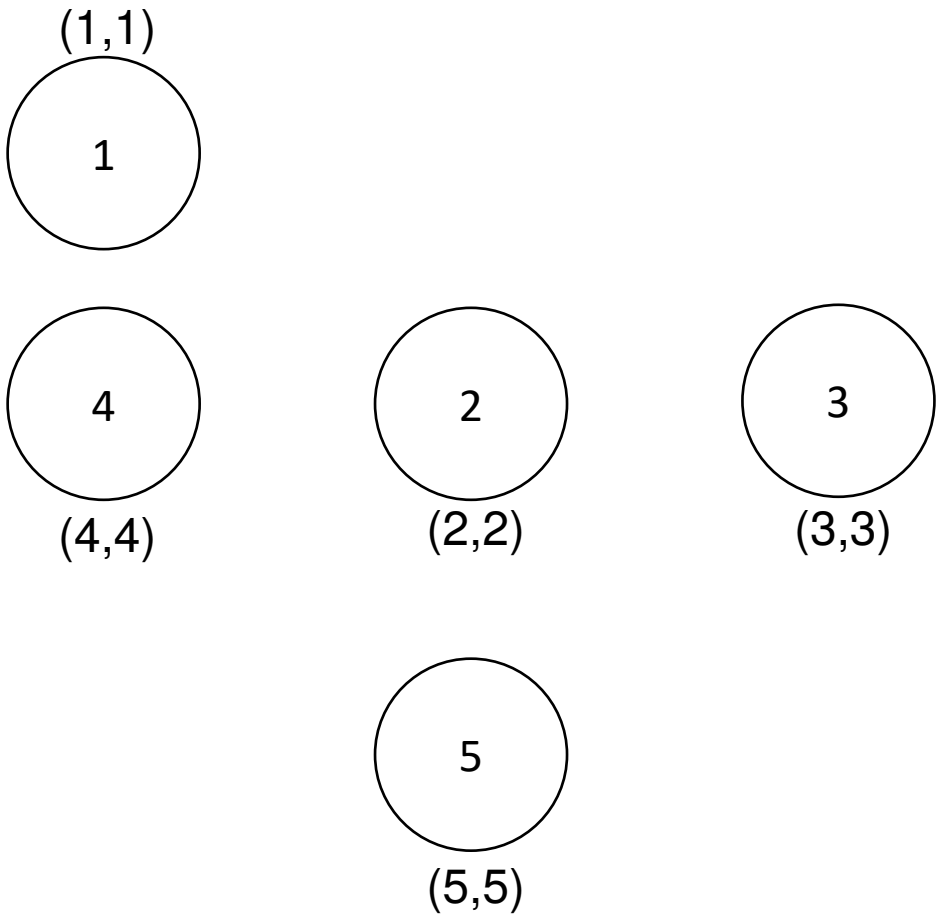
1G. 문제집 만들기

출제자: 김준서

- 주어진 조건에서 x~y 범위의 최소값과 최대값을 찾는 문제

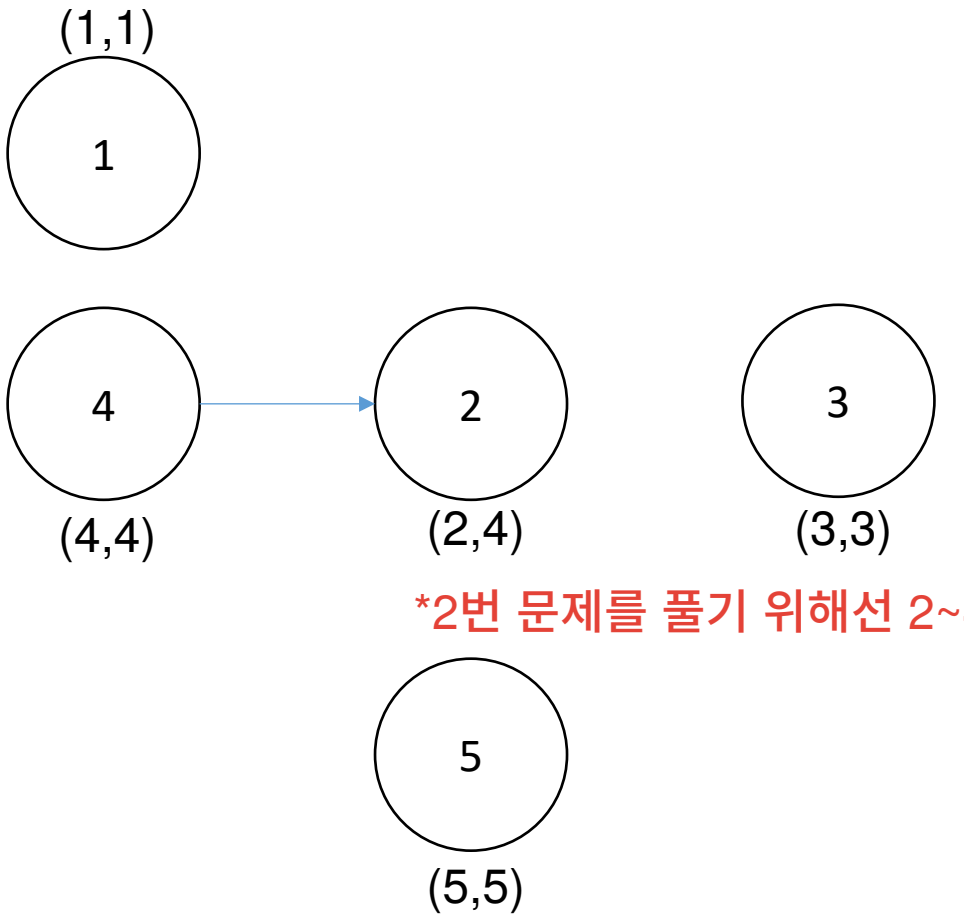
1G. 문제집 만들기

출제자: 김준서



1G. 문제집 만들기

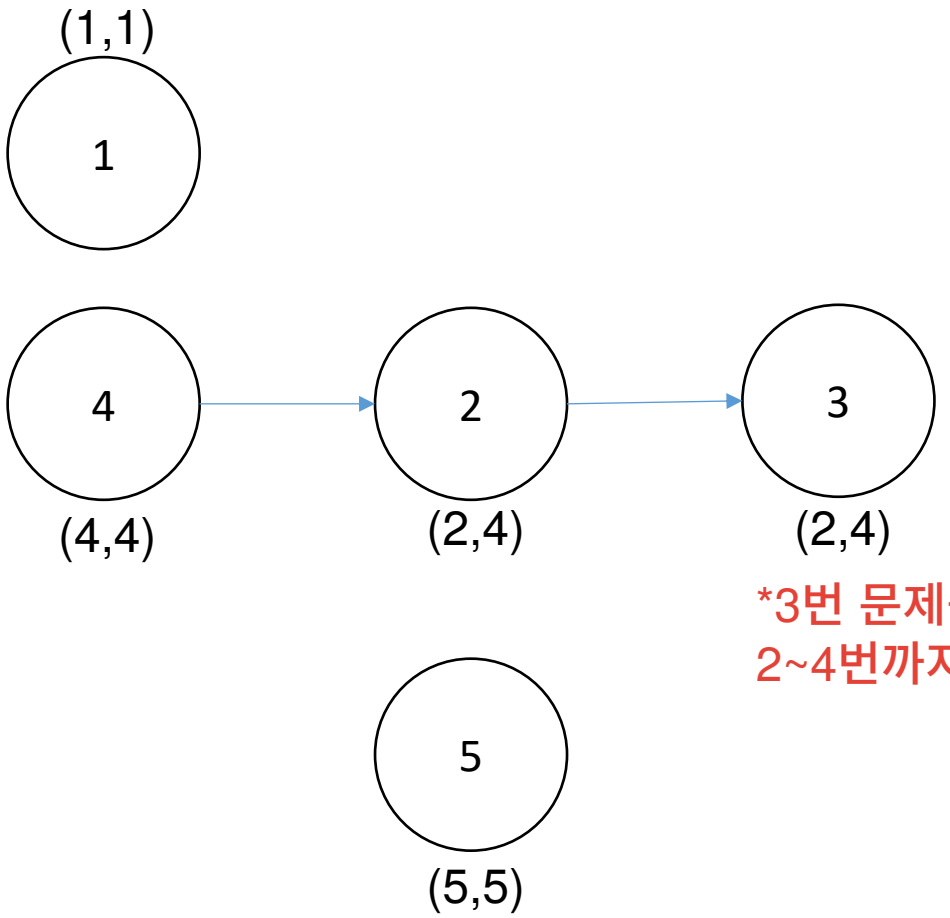
출제자: 김준서



*2번 문제를 풀기 위해선 2~4번까지의 범위가 필요

1G. 문제집 만들기

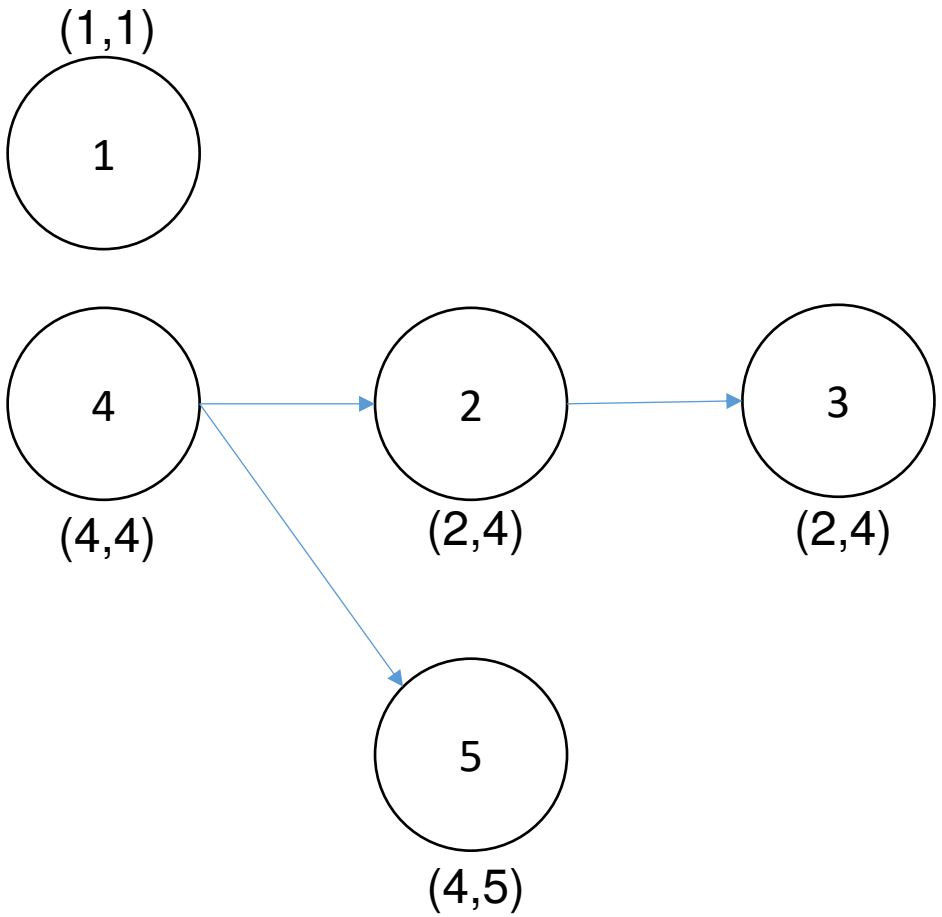
출제자: 김준서



*3번 문제를 풀기 위해선
2~4번까지의 문제가 필요

1G. 문제집 만들기

출제자: 김준서



*5번 문제를 풀기 위해선 4~5번까지의 문제가 필요

1G. 문제집 만들기

출제자: 김준서

- sub1 solution_O(NQ)
- 1) 위상 정렬을 사용하여 각 정점에서 현재 정점 번호에 해당하는 문제를 풀기 위해 필요한 정점 번호의 최소값, 최대값을 저장한다.

- 2) Q개의 작업횟수에 대해 x부터 y까지 반복문을 돌면서 최소값(min)과 최대값(max)을 구한다. 만약, min값과 x값이 같고, 동시에 max값과 y값이 같다면 YES를 출력한다. 아니라면, NO를 출력한다.

1G. 문제집 만들기

출제자: 김준서

- sub2 solution_O(QlogN)
- 1) 위상 정렬을 사용하여 각 정점에서 현재 정점 번호에 해당하는 문제를 풀기 위해 필요한 문제 번호의 최소값, 최대값을 저장한다.

- 2) Q개의 작업횟수에 대해 세그먼트 트리를 이용하여 x부터 y까지의 최소값(min)과 최대값(max)을 구한다. 만약, min값과 x값이 같고, 동시에 max값과 y값이 같다면 YES를 출력한다. 아니라면, NO를 출력한다.

1G. 문제집 만들기

출제자: 김준서

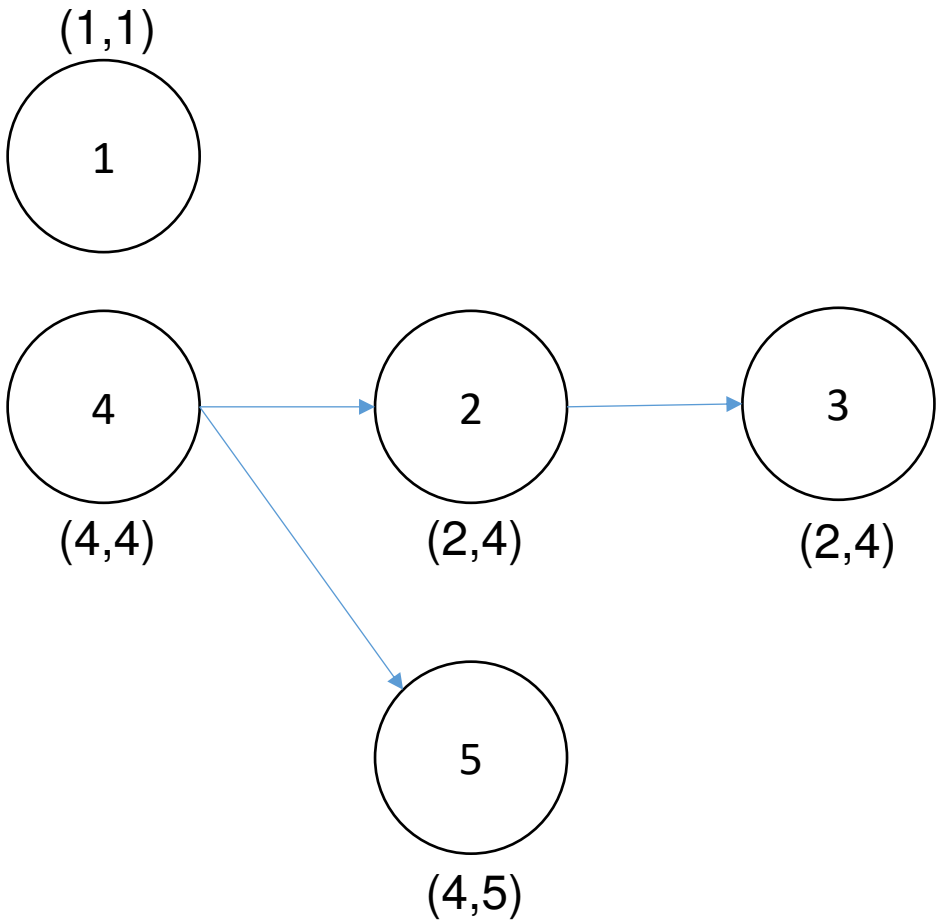
sub3 solution_O((N+Q)logN)

- 1) 각 노드에서 현재 정점 번호에 해당하는 문제를 풀기 위해 필요한 문제들을 저장한다.

(단, 연결된 바로 이전 정점의 정보들만 저장한다.)

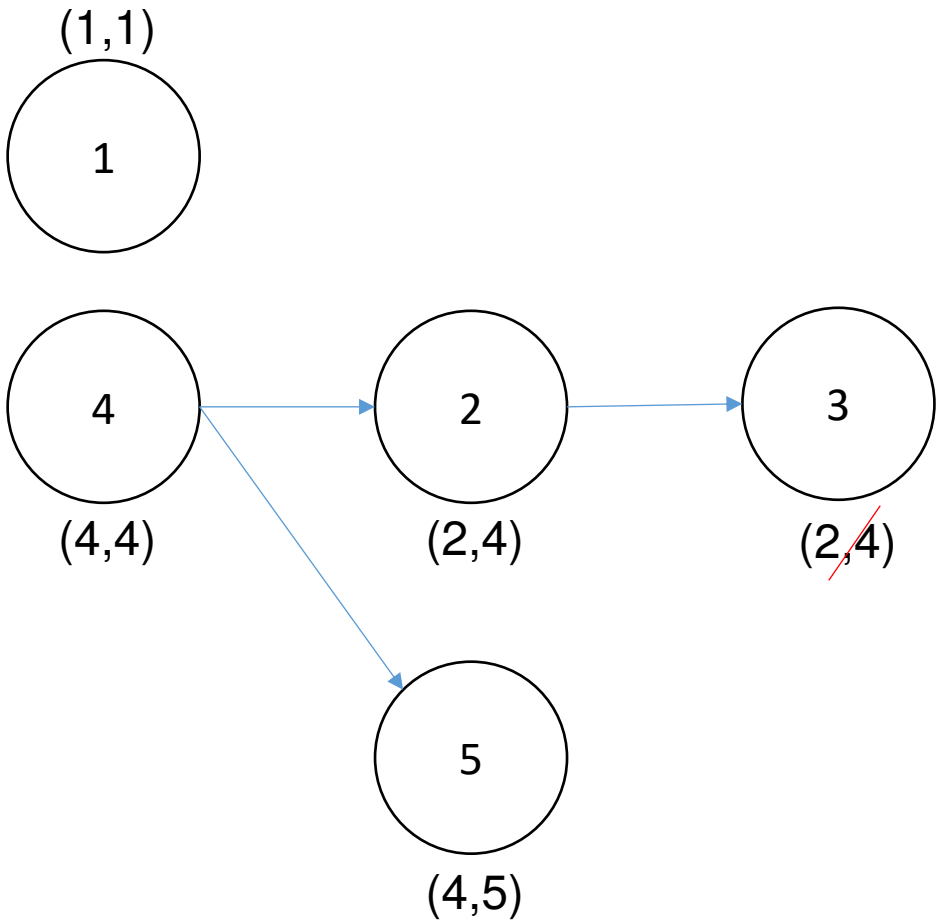
1G. 문제집 만들기

출제자: 김준서



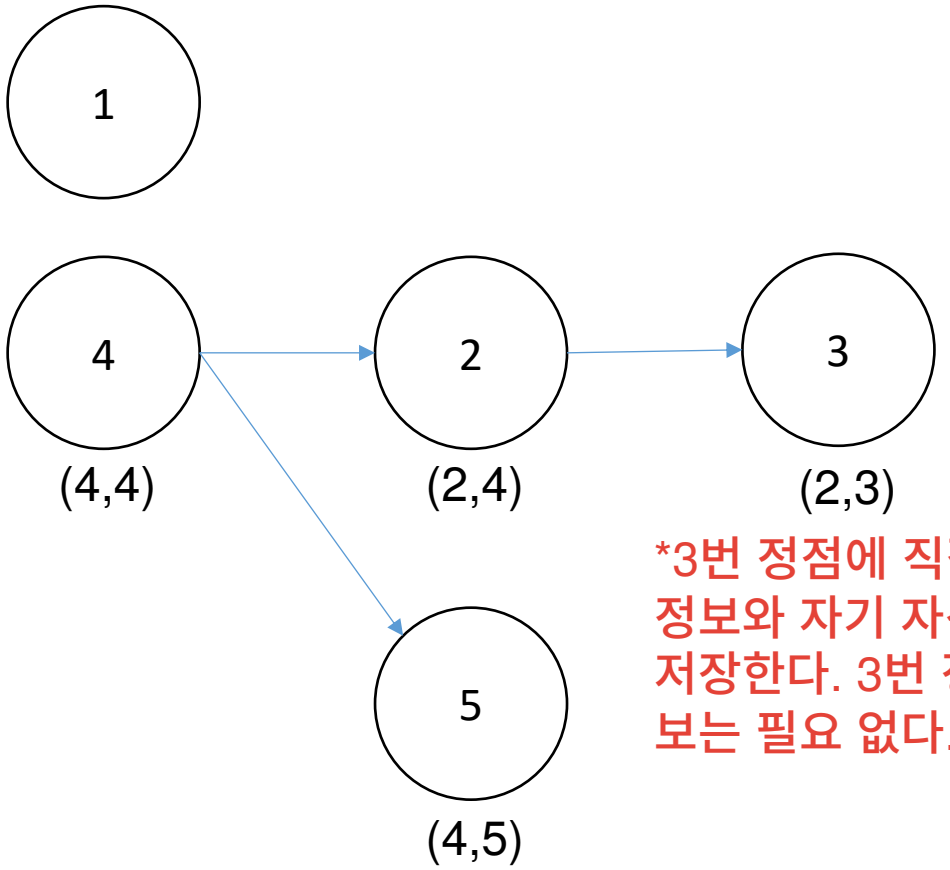
1G. 문제집 만들기

출제자: 김준서



1G. 문제집 만들기

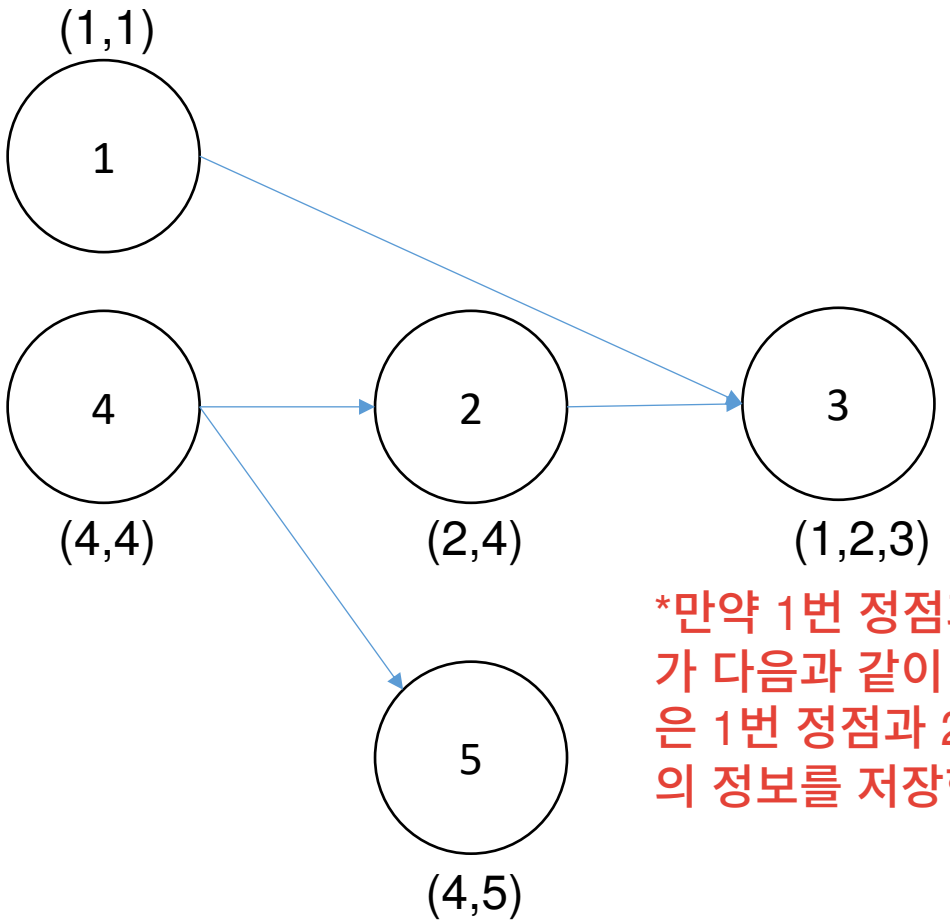
출제자: 김준서



*3번 정점에 직접 연결된 2번 정점의 정보와 자기 자신의 정보를 3번 정점에 저장한다. 3번 정점에서 4번 정점의 정보는 필요 없다.

1G. 문제집 만들기

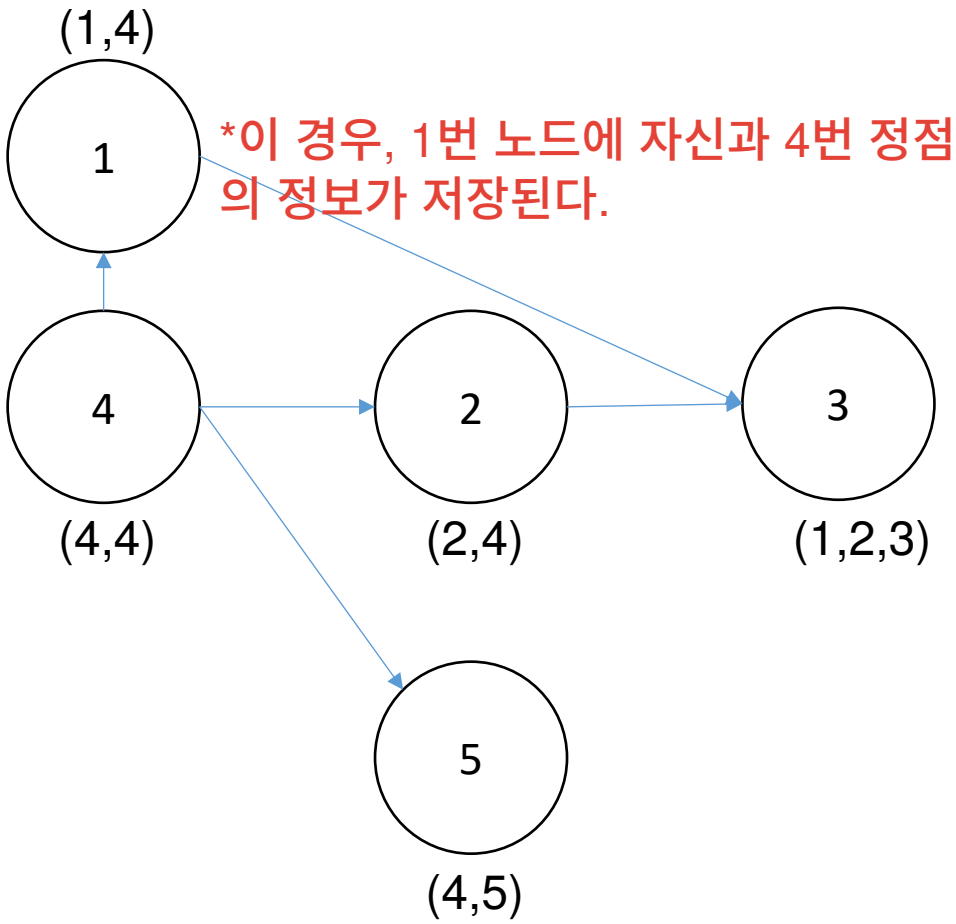
출제자: 김준서



*만약 1번 정점과 3번 정점 간의 관계가 다음과 같이 생성된다면, 3번 정점은 1번 정점과 2번 정점, 그리고 자신의 정보를 저장한다.

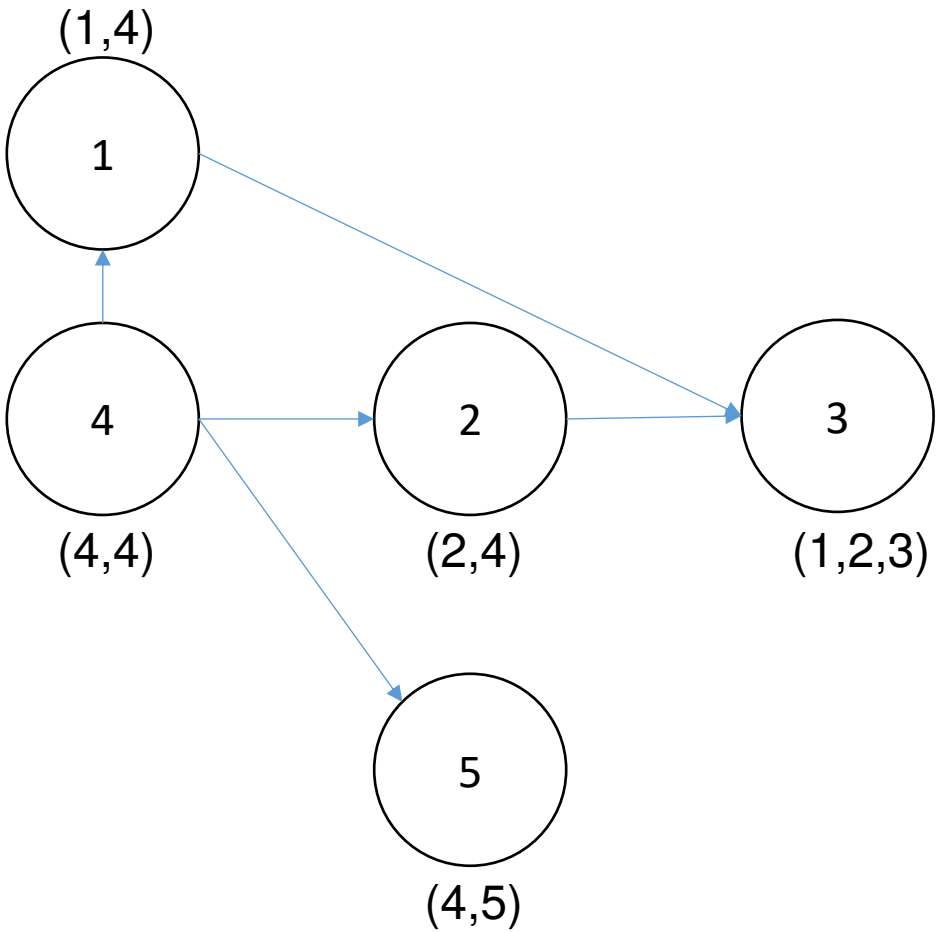
1G. 문제집 만들기

출제자: 김준서



1G. 문제집 만들기

출제자: 김준서



바로 이전에 연결된 정점들만 저장해도 되는 이유는, 다음과 같다.

1) 만약 x번 부터 3번 까지의 문제들을 풀 수 있는지 확인 하고자 할 때, 3번 정점에서 1의 최소값과 3의 최대값을 얻을 수 있다.

2) 1의 최소값을 가졌을 때,
만약 1 < x이면 답은 NO이다.
만약 x <= 1 이면 정점 1에 저장된 최소값, 최대값을 탐색하여야 한다.
즉, 3번에 저장된 1번 정점의 정보를 탐색하게 된다.

3) 1), 2)에서 볼 수 있듯이, 바로 이전에 연결된 정점만 저장하여도 x~y 구간 내에 있는 정점의 정보는 전부 탐색하므로 정답을 판단할 수 있음을 알 수 있다.

1G. 문제집 만들기

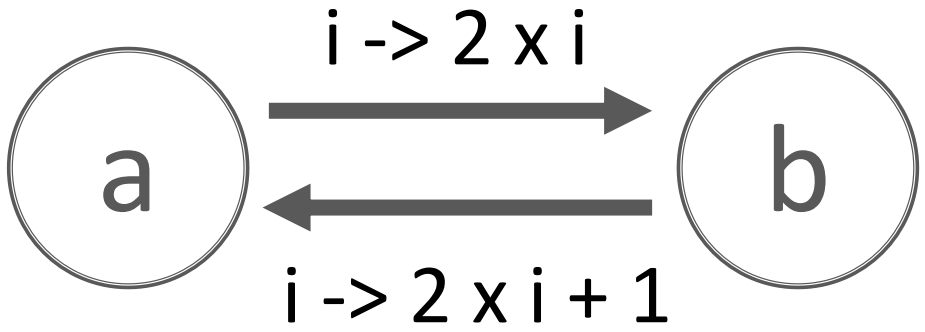
출제자: 김준서

- sub3 solution_O((N+Q)logN)
- 2) 1 x y : 세그먼트 트리를 이용하여 x부터 y까지의 최소값(min)과 최대값(max)을 구한다. 만약, min값과 x값이 같고, 동시에 max값과 y값이 같다면 YES를 출력한다. 아니라면, NO를 출력한다.
- 3) 2 x y : 노드 x에 저장된 y값을 삭제 한다.
- 4) 3 x y : 노드 x에 y를 추가한다.

1H. 이건 버그야!

출제자: 홍준표

- 서브태스크1:
  - 정점이 아닌 간선을 기준으로 dp를 정의합니다.



i번째로 입력된 간선 번호를 재정의

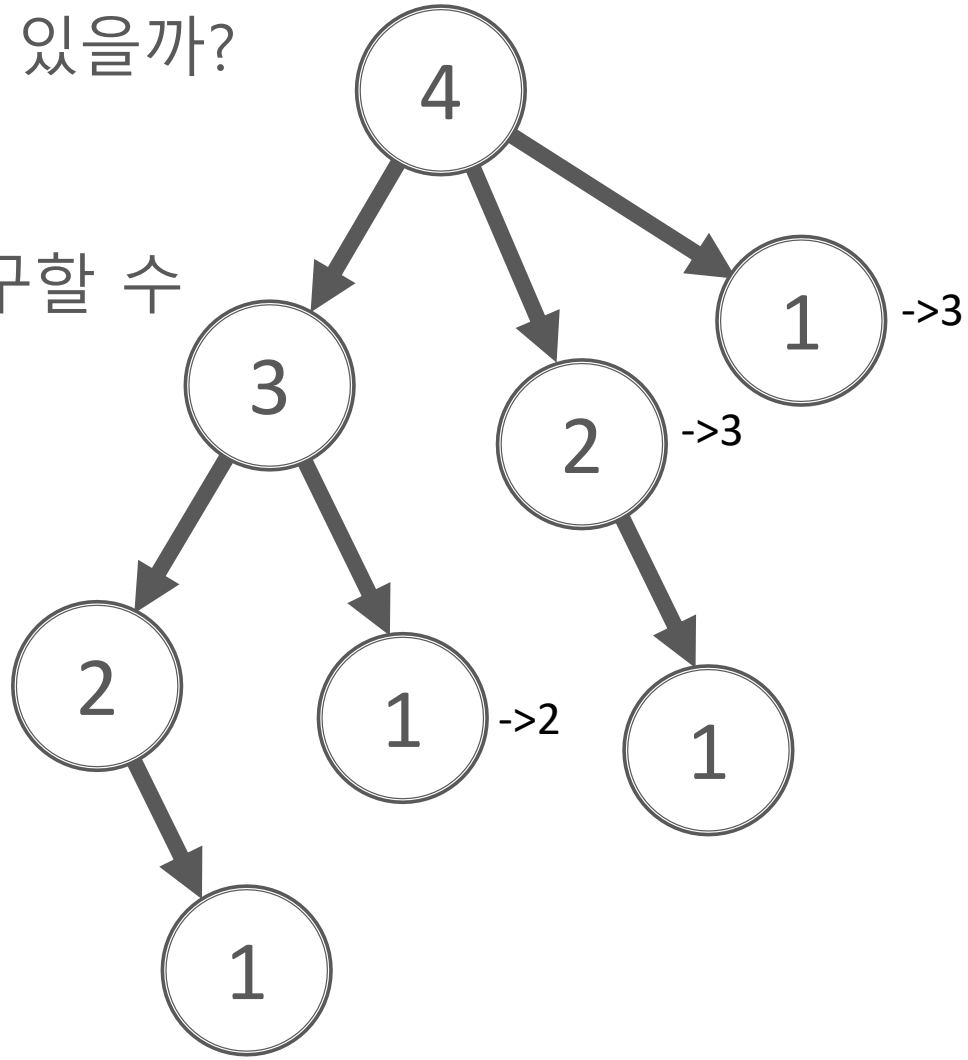
  - dp[j] = j 번째 간선의 꼬리를 선봉으로 지정하고 간선 방향으로 공격을 갈 때의 최적
  - ep[j] = 꼬리를 루트로 하는 서브트리 내에 선봉을 설치했을 때의 최적
- 간선을 나눠 재정의 하면 간선의 개수 x 2 개의 상태공간으로 나타낼 수 있음.

1H. 이건 버그야!

출제자: 홍준표

- 자식의 최적을 이용해 부모의 최적을 구할 수 있을까?

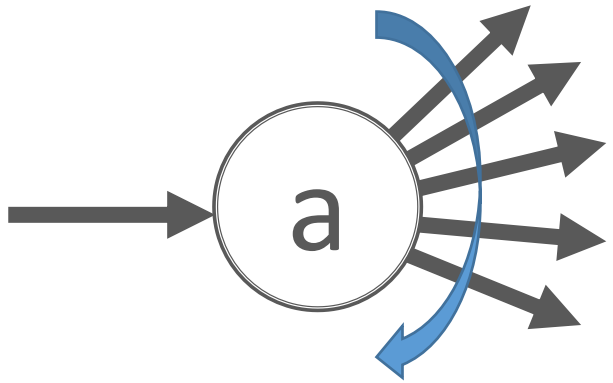
- 자식노드의 전체합과 높이를 구하면 최적을 구할 수 있다.



1H. 이건 버그야!

출제자: 홍준표

- 한 노드의 상태를 채우기 위해
- 그 방향을 제외한 모든 간선의 최적값을 확인해야함.



최대 N개를 확인

- 시간복잡도 : O(N^2)

1H. 이건 버그야!

출제자: 홍준표

- 서브태스크2:
  - 루트를 하나 잡고 상태공간을 채우면
  - 채워지지 않는 상태공간의 개수는 N-1개
- 기존에 업데이트 된 정보 여러개를 빠르게 가져올 수 있을까?
- 노드를 트리로 관리해 나가는 간선 상태의 값을 구간으로 가져올 수 있도록 설계
- 시간복잡도 : O(NlogN)
- O(N) 솔루션도 존재한다.