A. 일우는 야바위꾼

- 바꾸는 두 컵 중에 공이 있을 경우, 공의 위치를 이동되는 컵에 맞춰서 옮겨준다.

시간 복잡도 : O(K)

혹은,

- 길이 N의 배열 A를 선언해서 공이 있는 위치가 X라고 할 때, A_X = 1로 표시.
- 두 컵 i, j의 위치를 교환할 때, swap(A_i, A_j) 하는 방식으로도 구현 가능.

시간 복잡도 : O(N+K)

분류 : 시뮬레이션

B. 유니대전 퀴즈쇼

- 채팅기록을 앞에서부터 탐색하다가, S가 등장한 경우. 정답을 저장.
- 다시 채팅 기록을 탐색하면서, S의 채팅 이전의 기록 중, 정답이 몇 번 등장했는 지 세준다.

시간 복잡도 : O(|length| * N)

분류 : 문자열, 탐색

C. 당근 키우기

- 핵심. 만약, 온기 혹은 수분이 이미 0이라면 감소되지 않는다.
- 예제 2 설명 : 햇빛을 123456 + 1234번 받고, 물을 12345번 받으면. 총 137075번
- 온기 123456, 수분 12345인 상태가 된다.
- 정답 : A + B + $\left\lfloor \frac{\min(A,B)}{10} \right\rfloor$

시간 복잡도 : O(1)

분류 : 수학

D. 부동산 다툼

N개의 노드를 가지는 완전 이진 트리 (complete binary tree)에서, 1번 노드에서 X번 노드까지 순차적으로 이동할 때 가장 처음 만나는 점유된 노드의 번호를 출력하는 문제. 혹은 만나지 않는 경우를 판별하여 노드 X를 점유하는 문제.

- 크기 N+1의 배열 A를 선언.
- 노드 i가 점유됐다면, A_i = 1이라고 하자.
- 1번 노드에서 X번 노드까지 제대로 계산하며 내려가는 건 왠지 어려우므로, X번 노드에서 1번 노드로 올라가보자. j번 노드의 부모노드는 $\left\lfloor \frac{j}{2} \right\rfloor$ 번 노드이므로, 쉽게 1번 노드까지 계산하며 올라갈 수 있다.
- 올라가면서 마주친 점유된 노드 중 가장 마지막에 만나는 노드를 출력하자. 만약, 그런 노드가 없었다면 A_X = 1로 표시하여 점유
- 트리의 깊이는 최대 O(LogN)이므로 전체 문제를 O(QlogN)에 해결 가능하다.

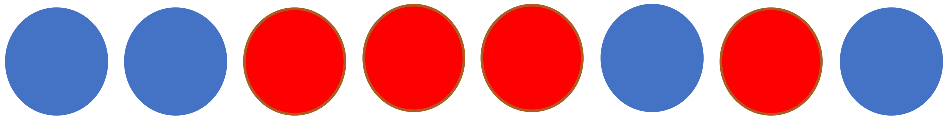
시간 복잡도 : O(QlogN)

분류 : 시뮬레이션, 탐색, 트리

E. 블로그2

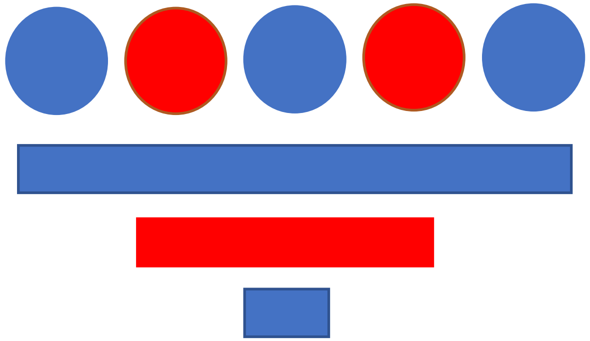
무색인 상태를 시작으로, 하나의 구간을 골라서 임의의 색으로 칠하는 연산을 할 수 있을 때, 주어진 색 배열을 만들 수 있는 최소 연산 횟수를 구하는 문제.

- $A_L = A_{L-1}$ 이라면, L~R을 칠하는 것보다 L-1 ~ R을 칠하는 것이 항상 유리하다.
- $A_R = A_{R+1}$ 인 경우도 마찬가지.



- 예를 들어, 4~7번 점을 빨간 색으로 칠하는 것보다 3~7번 점을 빨간 색으로 칠하는 것이 항상 유리하다.
- 이 관찰을 이용하면, 연속한 같은 색의 점들을 하나의 점으로 볼 수 있고.
- 위의 색 배열을 아래처럼 표현할 수 있다.

E. 블로그2



- i≤l & r≤j 즉, 구간 l~r이 구간 i~j 에 완전히 포함된다면, 바깥에 있는 i~j 를 l~r 보다 먼저 칠하는 것이 항상 유리하다.

- $A_L = A_R$ 인 경우만 칠하는 것이 항상 유리하다. 등등… 여러가지 관찰을 토대로 얻을 수 있는 가장 간단한 최선의 전략은 가장 바깥에 있는 같은 색의 두 점을 양끝으로 하는 구간을 칠하는 것이다. 양끝점이 다른 경우에는, 한쪽 끝점은 따로 칠해준다.

- 위와 같이 색이 계속 반전되게끔 상태를 축약했을 때, 남은 점의 개수를 N이라고 하면.
- $\left\lfloor \frac{N}{2} \right\rfloor + 1$이 정답이 된다.

시간 복잡도 : O(N)

분류 : 탐욕법

F. 같이 눈사람 만들래?

A_i + A_j 와 A_k + A_l 두 값의 차이가 최소이게끔 서로 다른 i, j, k, l를 고르는 문제.

- a ≤ b ≤ c ≤ d 를 만족하는 a, b, c, d에 대해 a+d, b+c로 각각 눈사람을 만드는 것이 항상 최적이다.
  증명은 생략.

- 배열 A를 오름차순으로 정렬하자.

- i < j < k을 만족하는 i, j, k에 대해,

세 인덱스 i, j, k를 고정시켰을 때, A_i+X, A_j+A_k 값의 차이가 최적이 되려면, X는 -A_i+A_j+A_k과의 차이가 최소여야 한다. 이런 X는 -A_i+A_j+A_k 이하인 값 중 가장 큰 값 혹은 -A_i+A_j+A_k이상인 값 중 가장 작은 값이라고 할 수 있다.

- k가 증가할수록 A_k 그리고 -A_i+A_j+A_k가 단조증가하므로 최적이 되는 X 또한 단조증가한다.

즉, X = A_l을 만족하는 l 에 대해서, k가 증가할수록 l 또한 단조증가한다.

※ 단조증가 : 감소하지 않음

F. 같이 눈사람 만들래?

- k가 증가할수록 A_k 그리고 -A_i+A_j+A_k가 단조증가하므로 최적이 되는 X 또한 단조증가한다.

즉, X = A_l을 만족하는 l 에 대해서, k가 증가할수록 l 또한 단조증가한다.

i	j	k		l					
1	3	3	4	5	8	10	11	20	21

-A_i+A_j+A_k = 5

i	j		k	l					
1	3	3	4	5	8	10	11	20	21

-A_i+A_j+A_k = 6

i	j			k	l				
1	3	3	4	5	8	10	11	20	21

-A_i+A_j+A_k = 7

i	j				k	l			
1	3	3	4	5	8	10	11	20	21

-A_i+A_j+A_k = 10

i	j					k	l		
1	3	3	4	5	8	10	11	20	21

-A_i+A_j+A_k = 12

F. 같이 눈사람 만들래?

- i, j를 고정시킨 채로 k를 증가시키면서 최적인 l 위치를 계속 관리해준다면, l은 절대 감소하지 않으므로, 모든 i, j, k에 대해 탐색하는 O(N^3) 복잡도에 문제를 해결할 수 있다.

- 또한, l 위치를 이분탐색하는 O($N^3\mathrm{LogN}$) 코드로도 충분히 문제를 해결할 수 있다.

- 여담으로 이 문제는 O($N^2\mathrm{LogN}$) 복잡도로도 문제를 해결할 수 있다.

- 시간 복잡도 : O(N^3)
- 분류 : 투 포인터, 정렬, 이분탐색

G. 3 Slot Matching

- 3개의 슬롯과 스택 존재, (1) 스택에서 정수 원소를 꺼내서 빈 슬롯에 저장하는 연산과 (2) 2개의 슬롯에 있는 원소를 꺼내고, 꺼낸 두 수를 곱해서 가중치를 얻는 연산 존재. 얻을 수 있는 최대 가중치 합은?
- 원소가 있는 슬롯의 개수가 0, 1개일 때. 스택에서 꺼내는 선택 (1)만 가능.
- 2개일 때, 그 두 수를 곱하는 행위는, 스택에서 하나 더 꺼낸 뒤에 3칸이 모두 차 있는 상태에서 곱하는 행위에 포함됨. (1)을 하는 것이 유리
- 3개 상태에서 2칸을 골라 소비하는 경우의 수는 3개, 그 결과로 1칸만 남게 됨. (2)연산을 해야 함.
- 즉, 최초에 1번 원소를 슬롯에 넣고 2,3번 원소를 넣고 연산(2). 4,5번 원소를 넣고 연산(2) 6,7번 원소 넣고 연산(2), 8,9번 원소 넣고 연산(2) 이런 식으로 반복하는 로직에 대해 생각.
- "i-1, i, j 세 원소가 슬롯에 있는 경우"로 현재의 게임진행 상태를 간단하게 표현할 수 있다.
- 이때 선택지는 i-1을 남기거나, i를 남기거나, j를 남기는 것 3가지 행동 존재

G. 3 Slot Matching

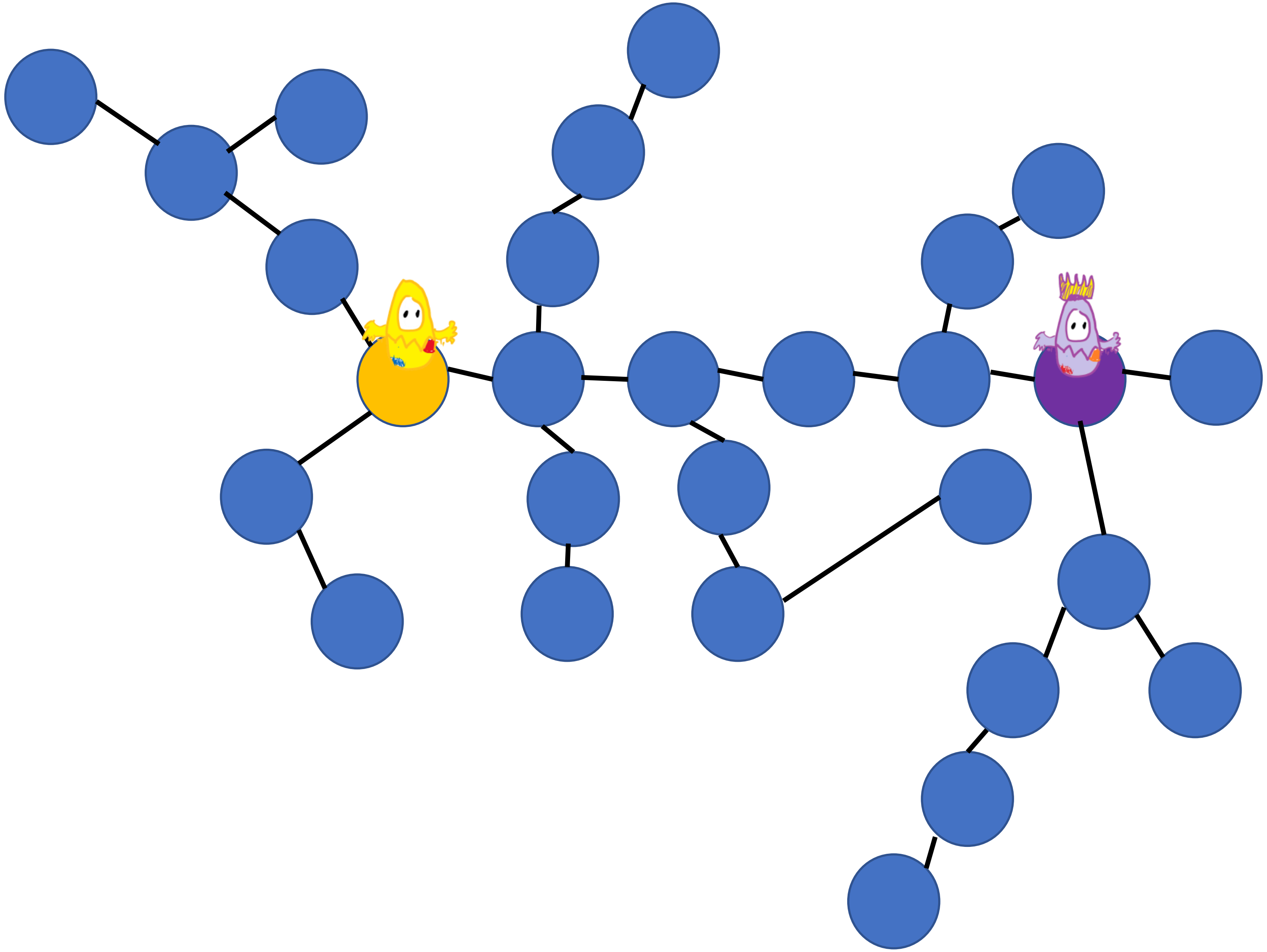
- 점화식을 세워보자.
- 함수 $D_{(i,j)}$ 정의 : i-1, i, j번째 원소가 슬롯에 들어있을 때, 앞으로 얻을 수 있는 가중치 합의 최댓값
- $D_{(i,j)}$ = max($D_{(i+2,j)} + A_i * A_{i-1}$, $D_{(i+2,i)} + A_j * A_{i-1}$, $D_{(i+2,i-1)} + A_j * A_i$)
- $D_{(2,0)}$가 정답
- 모든 i, j 쌍에 대응되는 상태공간에 대해 3가지 행동을 조사하는 정도의 시간으로 문제를 해결할 수 있다.

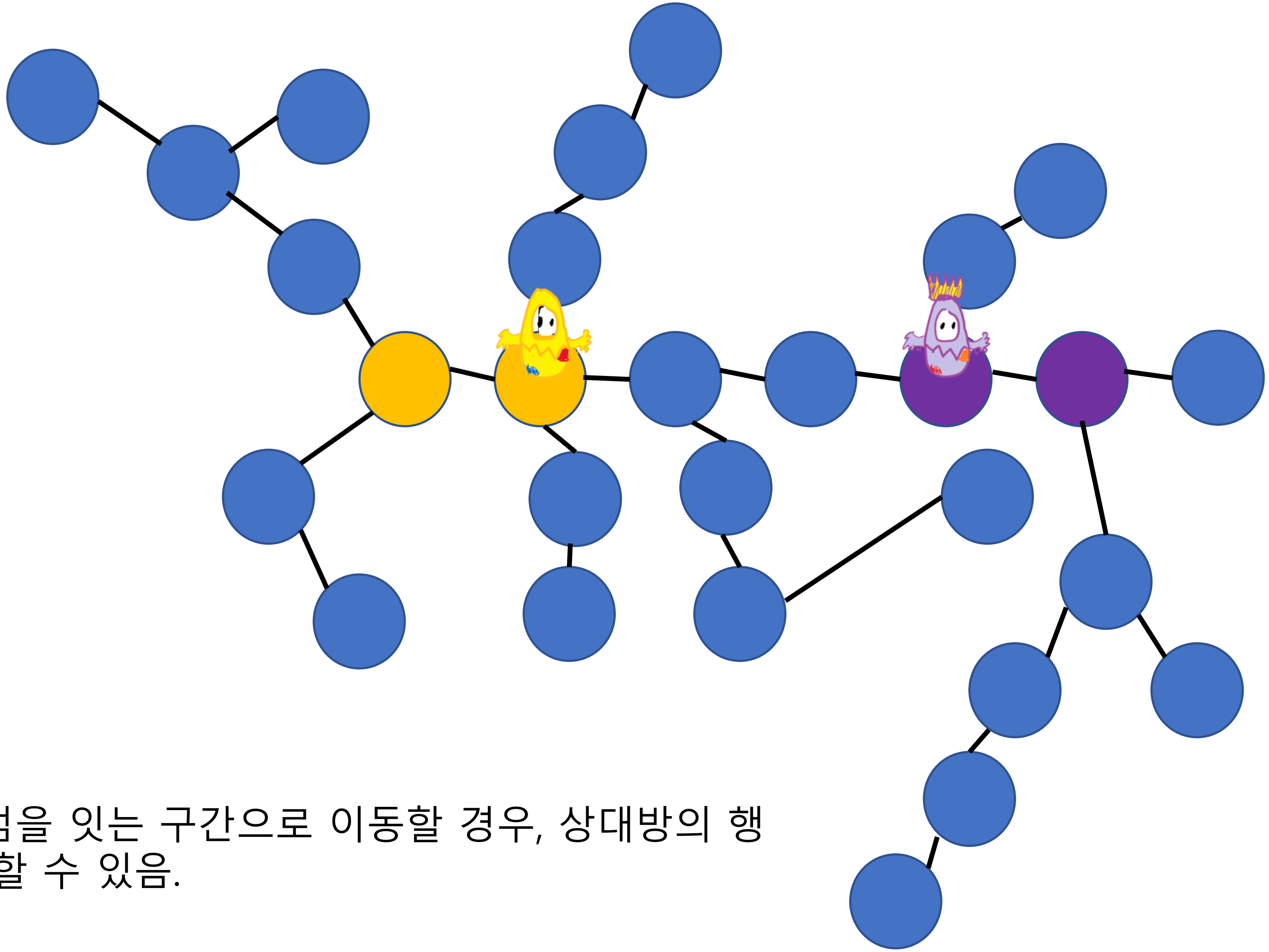
시간 복잡도 : O(N^2)
분류 : 다이나믹 프로그래밍

H. 트리 위의 폴 가이즈

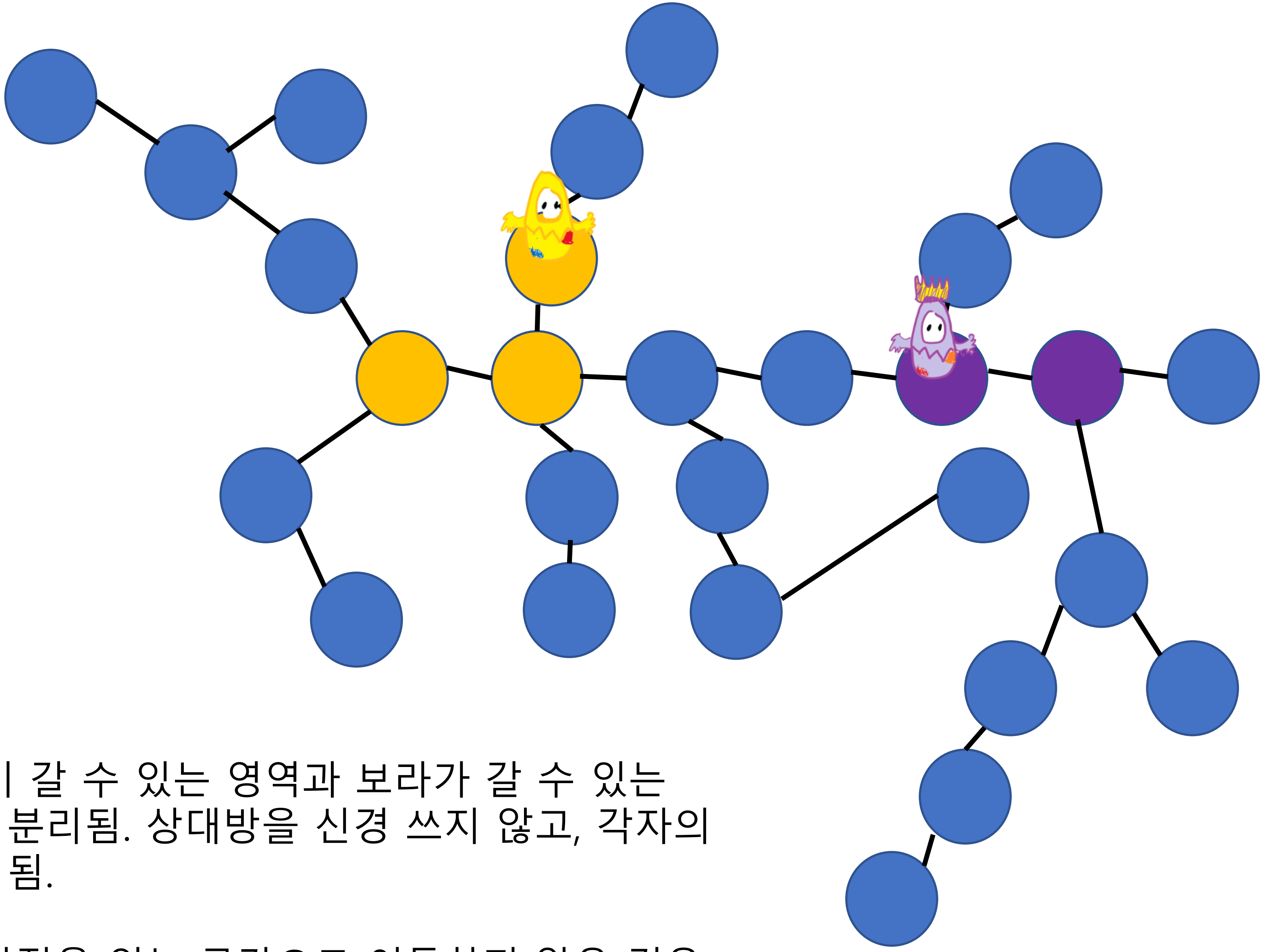
트리 위에서 2명의 플레이어가 서로 턴을 번갈아가며 1칸씩 이동. 두 사람의 경로가 겹쳐선 안됨. 두 사람 모두 (자신의 이동 횟수 – 상대방의 이동 횟수)를 최대화하는 전략으로 게임을 진행할 때, (1P 이동 횟수 – 2P 이동 횟수)를 구하는 문제.
※ 최대화하는 전략이란, 게임 진행의 모든 경우의 수를 계산했을 때의 해당 점수를 최대화하는 최선의 전략

- 어려움



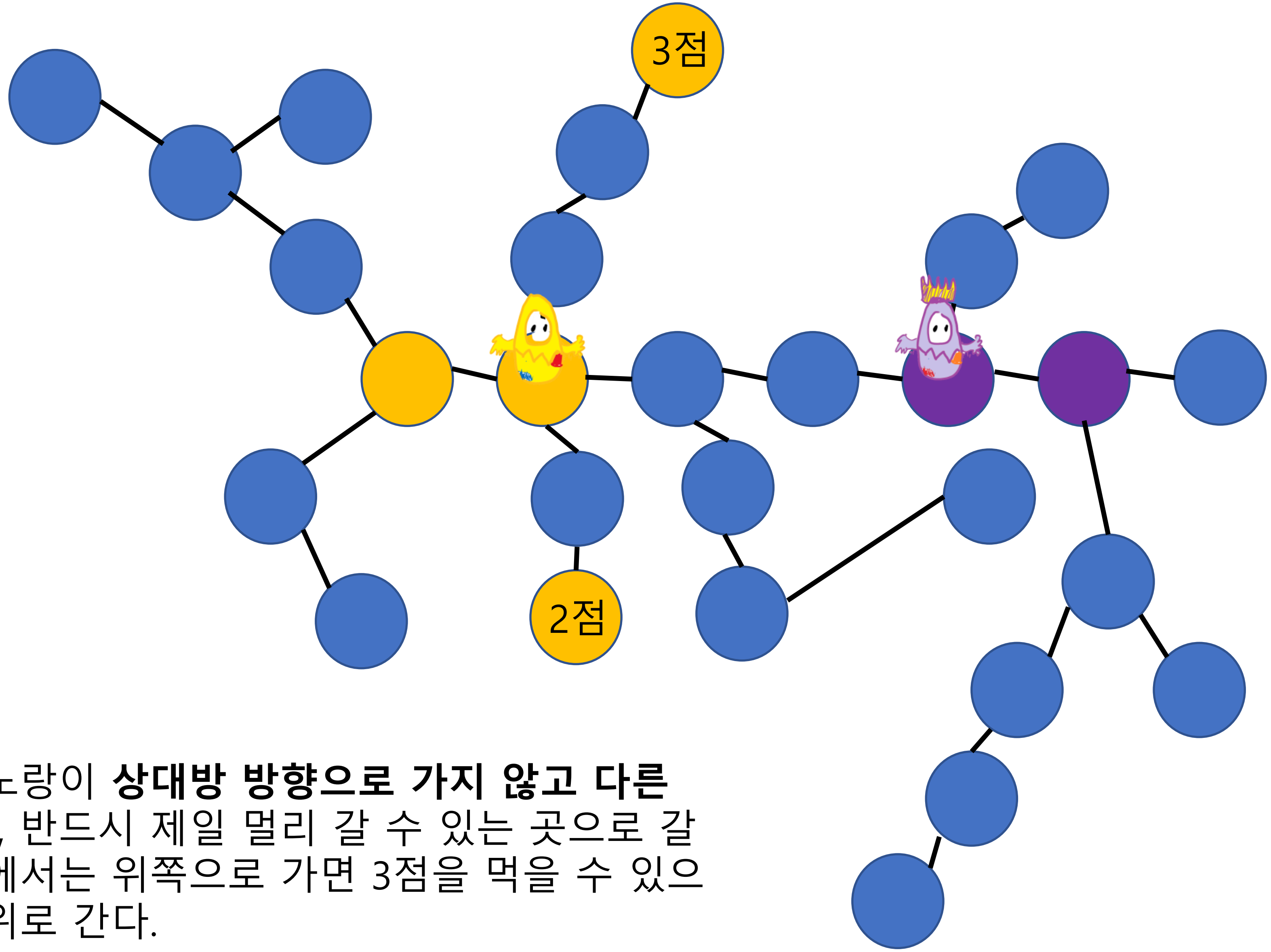


서로의 시작점을 잇는 구간으로 이동할 경우, 상대방의 행동에 간섭을 할 수 있음.

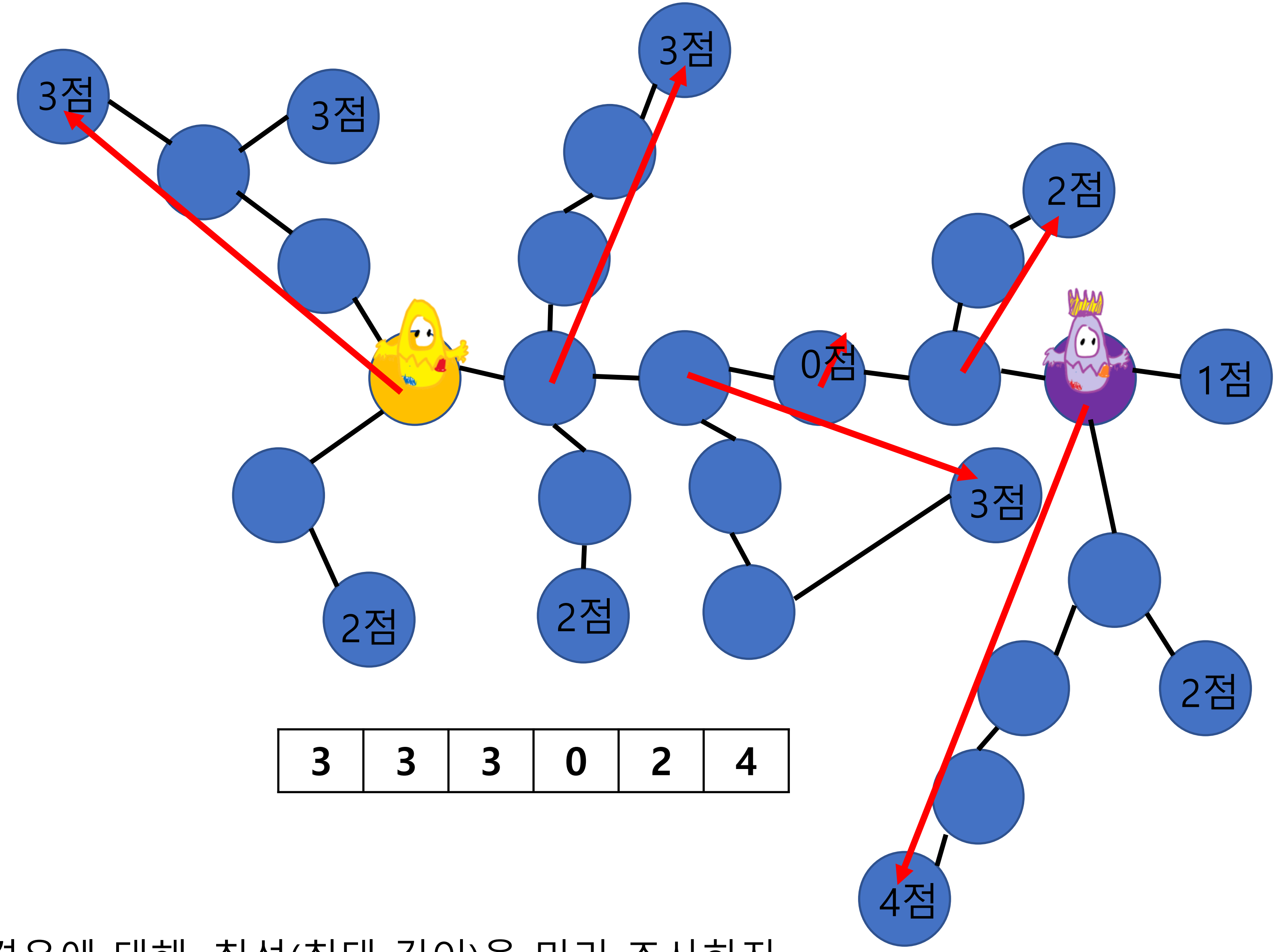


이 경우, 노랑이 갈 수 있는 영역과 보라가 갈 수 있는 영역이 완전히 분리됨. 상대방을 신경 쓰지 않고, 각자의 최선을 고르면 됨.

즉, 서로의 시작점을 잇는 구간으로 이동하지 않은 경우, 서로의 행동은 독립적으로 결정된다.



만약, 여기서 노랑이 **상대방 방향으로 가지 않고 다른 길로 빠진다면**, 반드시 제일 멀리 갈 수 있는 곳으로 갈 것임. 이 상황에서는 위쪽으로 가면 3점을 먹을 수 있으므로, 당연히 위로 간다.



3	3	3	0	2	4

샛길로 빠지는 경우에 대해, 최선(최대 깊이)을 미리 조사하자.
이제, 상대방과의 길을 하나의 배열처럼 표현할 수 있다.

H. 트리 위의 폴 가이즈

index	1	2	3	4	5	6
value	**3**	**3**	**3**	**0**	**2**	**4**

A =	**3**	**4**	**5**	**3**	**6**	**9**	1P

B =	**8**	**7**	**6**	**2**	**3**	**4**	2P

- 주어진 트리와 동일한 배열을 얻었다. 이제, 배열의 양 끝에서 시작하는 것으로 생각할 수 있다. 배열 내에서 이동할 때에도 1점씩 획득하므로 그 부분까지 고려하자. 문제를 조금 더 쉽게 생각하기 위해. 배열을 각 플레이어 별로 분리하여, 해당 칸에서 그 플레이어가 멈췄을 때 최종적으로 얻는 점수라고 하자.

- 각 플레이어는 자신의 턴에 한 칸 전진할 것인지, 배열에 적힌 점수를 얻고 마무리 할 것인지. 두 가지 중 하나의 행동을 선택하여 할 수 있다.

- 게임이 끝날 때의 1P 점수 – 2P 점수를 K라고 하자. 2P의 목표는 K를 최소화하는 것과 동치이다.

- 만약, 1P가 배열의 L, 2P가 R에 위치하고,

1P가 L에서 점수를 얻고 마무리한다면. K = $A_L - \max(B_{L+1} \sim B_R)$
2P가 R에서 점수를 얻고 마무리한다면. K = $\max(A_L \sim A_{R-1}) - B_R$이 된다.

H. 트리 위의 폴 가이즈

- $F_{(L,R)}$: 1P가 L, 2P가 R에 있고 현재 1P의 차례일 때, 결과 K
- $G_{(L,R)}$: 1P가 L, 2P가 R에 있고 현재 2P의 차례일 때, 결과 K

- $F_{(L,R)} = \max(G_{(L+1,R)},\ \ A_L - \max(B_{L+1} \sim B_R))$
- $G_{(L,R)} = min(F_{(L,R-1)},\ \ \max(A_L \sim A_{R-1}) - B_R)$

- $\max(A_i \sim A_j)$ 종류의 연산을 수행하는 데 O(X)의 시간이 걸린다면. 가능한 (L, R)쌍의 상태가 O(N)개이기 때문에, O(XN) 시간에 문제를 해결할 수 있다.

- 구간 max 연산을 빠르게 하는 방법은 "range maximum query"로 잘 알려진 방법들이 있다. O(LogN)

- 혹은, L'~R'이 점점 좁아진다는 특성을 이용해서. 위 함수의 과정을 거꾸로 수행하며, max값을 관리할 수 있다. O(1)

시간 복잡도 : O(N)
분류 : 트리, 게임 이론, 자료구조