목차

A. 펭귄추락대책위원회

- 펭귄이 있는 위치 x

A. 펭귄추락대책위원회

- 펭귄이 있는 위치 x
- x+1 ~ N에 위치한 얼음 중 단 하나만 부셔도 펭귄이 N과 단절된다. → 두개 이상 부술 필요가 없다.

A. 펭귄추락대책위원회

- 펭귄이 있는 위치 x
- x+1 ~ N에 위치한 얼음 중 단 하나만 부셔도 펭귄이 N과 단절된다. → 두개 이상 부술 필요가 없다.
- 마찬가지로, 1 ~ x-1에 위치한 얼음 중 단 하나만 부셔도 1과 단절된다.

A. 펭귄추락대책위원회

- 펭귄이 있는 위치 x
- x+1 ~ N에 위치한 얼음 중 단 하나만 부셔도 펭귄이 N과 단절된다. → 두개 이상 부술 필요가 없다.
- 마찬가지로, 1 ~ x-1에 위치한 얼음 중 단 하나만 부셔도 1과 단절된다.
- 펭귄 왼쪽부분에서 하나, 오른쪽부분에서 하나씩 얼음을 부수는 문제.

A. 펭귄추락대책위원회

- 펭귄이 있는 위치 x
- x+1 ~ N에 위치한 얼음 중 단 하나만 부셔도 펭귄이 N과 단절된다. → 두개 이상 부술 필요가 없다.
- 마찬가지로, 1 ~ x-1에 위치한 얼음 중 단 하나만 부셔도 1과 단절된다.
- 펭귄 왼쪽부분에서 하나, 오른쪽부분에서 하나씩 얼음을 부수는 문제.
- 각각 비용이 적은 걸 부수는 게 좋다. → 양쪽에서 최소값 하나씩 찾기.

A. 펭귄추락대책위원회

- 펭귄이 있는 위치 x
- x+1 ~ N에 위치한 얼음 중 단 하나만 부셔도 펭귄이 N과 단절된다. → 두개 이상 부술 필요가 없다.
- 마찬가지로, 1 ~ x-1에 위치한 얼음 중 단 하나만 부셔도 1과 단절된다.
- 펭귄 왼쪽부분에서 하나, 오른쪽부분에서 하나씩 얼음을 부수는 문제.
- 각각 비용이 적은 걸 부수는 게 좋다. → 양쪽에서 최소값 하나씩 찾기.
- 정답은 MIN(A[1] ~ A[x-1]) + MIN(A[x+1] ~ A[N])

시간복잡도 : O(N) 분류 : 탐색

B. 내가 살게, 아냐 내가 살게

- 처음으로 손을 K이상 뻗은 사람과 시점을 구하는 문제.

B. 내가 살게, 아냐 내가 살게

- 처음으로 손을 K이상 뻗은 사람과 시점을 구하는 문제.
- 사람들이 순서대로 손을 뻗는 과정을 구현

B. 내가 살게, 아냐 내가 살게

- 처음으로 손을 K이상 뻗은 사람과 시점을 구하는 문제.
- 사람들이 순서대로 손을 뻗는 과정을 구현
- 손을 뻗은 정도를 각 사람마다 저장.

B. 내가 살게, 아냐 내가 살게

- 처음으로 손을 K이상 뻗은 사람과 시점을 구하는 문제.
- 사람들이 순서대로 손을 뻗는 과정을 구현
- 손을 뻗은 정도를 각 사람마다 저장.
- Just Do It !

시간복잡도 : O(NM)　　　　분류 : 시뮬레이션

C. 2xN 예쁜 타일링

- 2x1을 두개 사용 = 1x2로 돌려서 두개 사용 (항상 2x1로만 사용해도 됨.)

C. 2xN 예쁜 타일링

- 2x1을 두개 사용 = 1x2로 돌려서 두개 사용 (항상 2x1로만 사용해도 됨.)
- 사용한 2x1 타일 : 2 3 5 8 10 20

C. 2xN 예쁜 타일링

- 2x1을 두개 사용 = 1x2로 돌려서 두개 사용 (항상 2x1로만 사용해도 됨.)
- 사용한 2x1 타일 : 2 3 5 8 10 20
- 같은 개수를 사용하는 더 좋은 방법 : 2 3 5 8 10 20

C. 2xN 예쁜 타일링

- 2x1을 두개 사용 = 1x2로 돌려서 두개 사용 (항상 2x1로만 사용해도 됨.)
- 사용한 2x1 타일 : 2 3 5 8 10 20
- 같은 개수를 사용하는 더 좋은 방법 : 2 3 5 8 10 20
- 어떤 종류의 타일을 사용할거면 그 종류에서 가장 예쁜 타일부터 사용하면 됨.
- 2x2 타일도 마찬가지

C. 2xN 예쁜 타일링

- 2x1을 두개 사용 = 1x2로 돌려서 두개 사용 (항상 2x1로만 사용해도 됨.)
- 사용한 2x1 타일 : 2 3 5 8 10 20
- 같은 개수를 사용하는 더 좋은 방법 : 2 3 5 8 10 20
- 어떤 종류의 타일을 사용할거면 그 종류에서 가장 예쁜 타일부터 사용하면 됨.
- 2x2 타일도 마찬가지
- N이 홀수일 때 : 2x1 타일 하나는 반드시 사용해야 함. 설치하면 N은 짝수.

C. 2xN 예쁜 타일링

- 2x1을 두개 사용 = 1x2로 돌려서 두개 사용 (항상 2x1로만 사용해도 됨.)
- 사용한 2x1 타일 : 2 3 5 8 10 20
- 같은 개수를 사용하는 더 좋은 방법 : 2 3 5 8 10 20
- 어떤 종류의 타일을 사용할거면 그 종류에서 가장 예쁜 타일부터 사용하면 됨.
- 2x2 타일도 마찬가지
- N이 홀수일 때 : 2x1 타일 하나는 반드시 사용해야 함. 설치하면 N은 짝수.
- N이 짝수일 때 : 2x1 타일을 사용하면 다시 N이 홀수가 돼서 2x1 타일 하나를 추가로 설치해야 함. → 2x1 타일을 두개 동시에 놓는 것과 동일

C. 2xN 예쁜 타일링

- 2x1을 두개 사용 = 1x2로 돌려서 두개 사용 (항상 2x1로만 사용해도 됨.)
- 사용한 2x1 타일 : 2 3 5 8 10 20
- 같은 개수를 사용하는 더 좋은 방법 : 2 3 5 8 10 20
- 어떤 종류의 타일을 사용할거면 그 종류에서 가장 예쁜 타일부터 사용하면 됨.
- 2x2 타일도 마찬가지
- N이 홀수일 때 : 2x1 타일 하나는 반드시 사용해야 함. 설치하면 N은 짝수.
- N이 짝수일 때 : 2x1 타일을 사용하면 다시 N이 홀수가 돼서 2x1 타일 하나를 추가로 설치해야 함. → 2x1 타일을 두개 동시에 놓는 것과 동일
- N이 짝수일 때 : MAX(2x1 타일 두개의 합, 2x2 타일)을 놓음.

C. 2xN 예쁜 타일링

- 2x1을 두개 사용 = 1x2로 돌려서 두개 사용 (항상 2x1로만 사용해도 됨.)
- 사용한 2x1 타일 : 2 3 5 8 10 20
- 같은 개수를 사용하는 더 좋은 방법 : 2 3 5 8 10 20
- 어떤 종류의 타일을 사용할거면 그 종류에서 가장 예쁜 타일부터 사용하면 됨.
- 2x2 타일도 마찬가지
- N이 홀수일 때 : 2x1 타일 하나는 반드시 사용해야 함. 설치하면 N은 짝수.
- N이 짝수일 때 : 2x1 타일을 사용하면 다시 N이 홀수가 돼서 2x1 타일 하나를 추가로 설치해야 함. → 2x1 타일을 두개 동시에 놓는 것과 동일
- N이 짝수일 때 : MAX(2x1 타일 두개의 합, 2x2 타일)을 놓음.
- 귀납적으로 해결

시간복잡도 : O(NlogN)   분류 : 정렬, 탐욕법

D. 파괴된 도시

- 어떤 도시와 그 도시를 포함하여 인접한 도시가 전부 파괴되었다면 항상 폭탄을 떨어뜨릴 수 있음.

D. 파괴된 도시

- 어떤 도시와 그 도시를 포함하여 인접한 도시가 전부 파괴되었다면 항상 폭탄을 떨어뜨릴 수 있음.
- 그렇지 않다면 항상 폭탄을 떨어뜨릴 수 없음.

D. 파괴된 도시

- 어떤 도시와 그 도시를 포함하여 인접한 도시가 전부 파괴되었다면 항상 폭탄을 떨어뜨릴 수 있음.
- 그렇지 않다면 항상 폭탄을 떨어뜨릴 수 없음.
- 가능한 후보 도시에 폭탄을 전부 떨어뜨림 → 파괴되지 않아야 할 도시를 파괴하지 않으면서, 파괴되어야 할 도시를 가장 많이 파괴하는 방법.

D. 파괴된 도시

- 어떤 도시와 그 도시를 포함하여 인접한 도시가 전부 파괴되었다면 항상 폭탄을 떨어뜨릴 수 있음.
- 그렇지 않다면 항상 폭탄을 떨어뜨릴 수 없음.
- 가능한 후보 도시에 폭탄을 전부 떨어뜨림 → 파괴되지 않아야 할 도시를 파괴하지 않으면서, 파괴되어야 할 도시를 가장 많이 파괴하는 방법.
- 이렇게 해도 파괴되지 않는 도시는 절대 파괴될 수 없음.

D. 파괴된 도시

- 어떤 도시와 그 도시를 포함하여 인접한 도시가 전부 파괴되었다면 항상 폭탄을 떨어뜨릴 수 있음.
- 그렇지 않다면 항상 폭탄을 떨어뜨릴 수 없음.
- 가능한 후보 도시에 폭탄을 전부 떨어뜨림 → 파괴되지 않아야 할 도시를 파괴하지 않으면서, 파괴되어야 할 도시를 가장 많이 파괴하는 방법.
- 이렇게 해도 파괴되지 않는 도시는 절대 파괴될 수 없음.
- 그래프 구축

시간복잡도 : O(N+M)　　　　분류 : 그래프

E. 텔레포트 정거장

- 점 S에서 점 E로 가는 최단시간을 구하는 문제.

E. 텔레포트 정거장

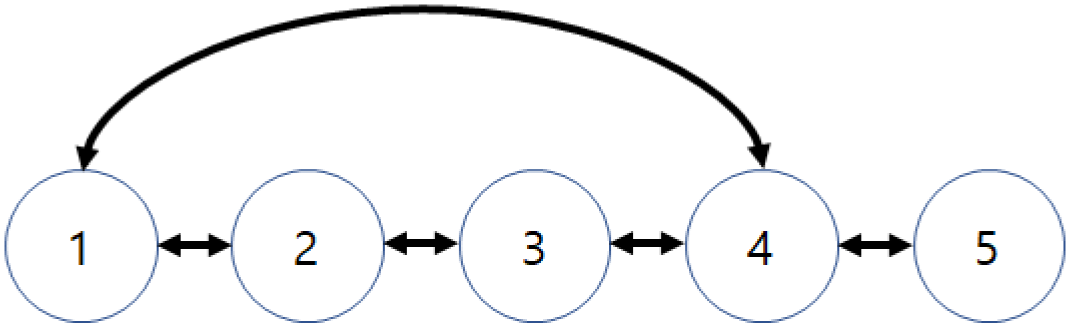
- 점 S에서 점 E로 가는 최단시간을 구하는 문제.
- 그래프 모델링?

E. 텔레포트 정거장

- 점 S에서 점 E로 가는 최단시간을 구하는 문제.
- 그래프 모델링?
- 현재위치가 X라면 X의 텔레포트 연결 정보와 X+1, X-1로의 간선이 있는 그래프. 각, 간선의 가중치 1

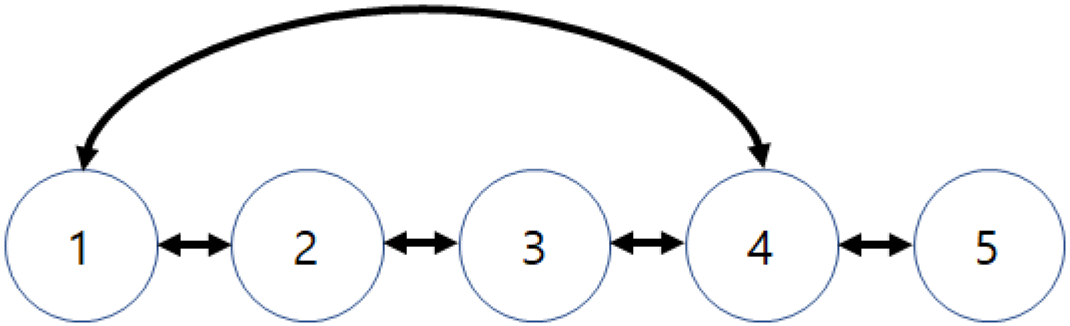
E. 텔레포트 정거장

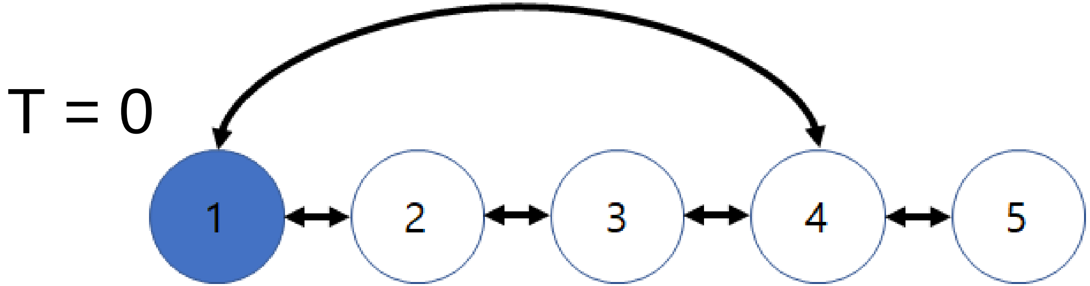
- 점 S에서 점 E로 가는 최단시간을 구하는 문제.
- 그래프 모델링?
- 현재위치가 X라면 X의 텔레포트 연결 정보와 X+1, X-1로의 간선이 있는 그래프. 각, 간선의 가중치 1



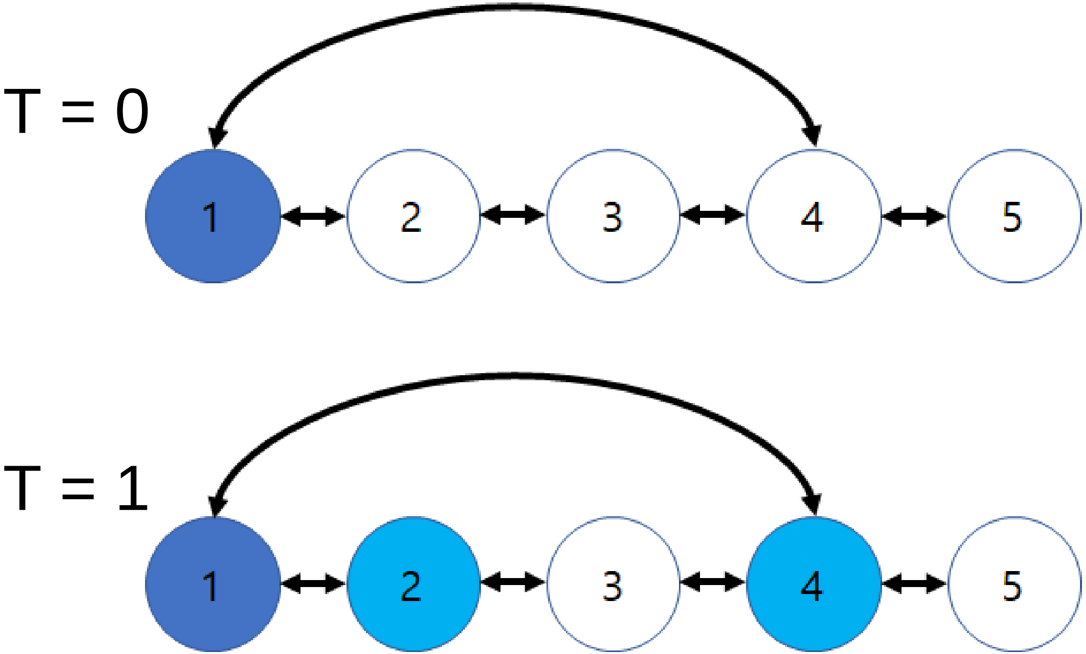
E. 텔레포트 정거장

- 점 S에서 점 E로 가는 최단시간을 구하는 문제.
- 그래프 모델링?
- 현재위치가 X라면 X의 텔레포트 연결 정보와 X+1, X-1로의 간선이 있는 그래프. 각, 간선의 가중치 1이라서 BFS로 해결할 수 있음.

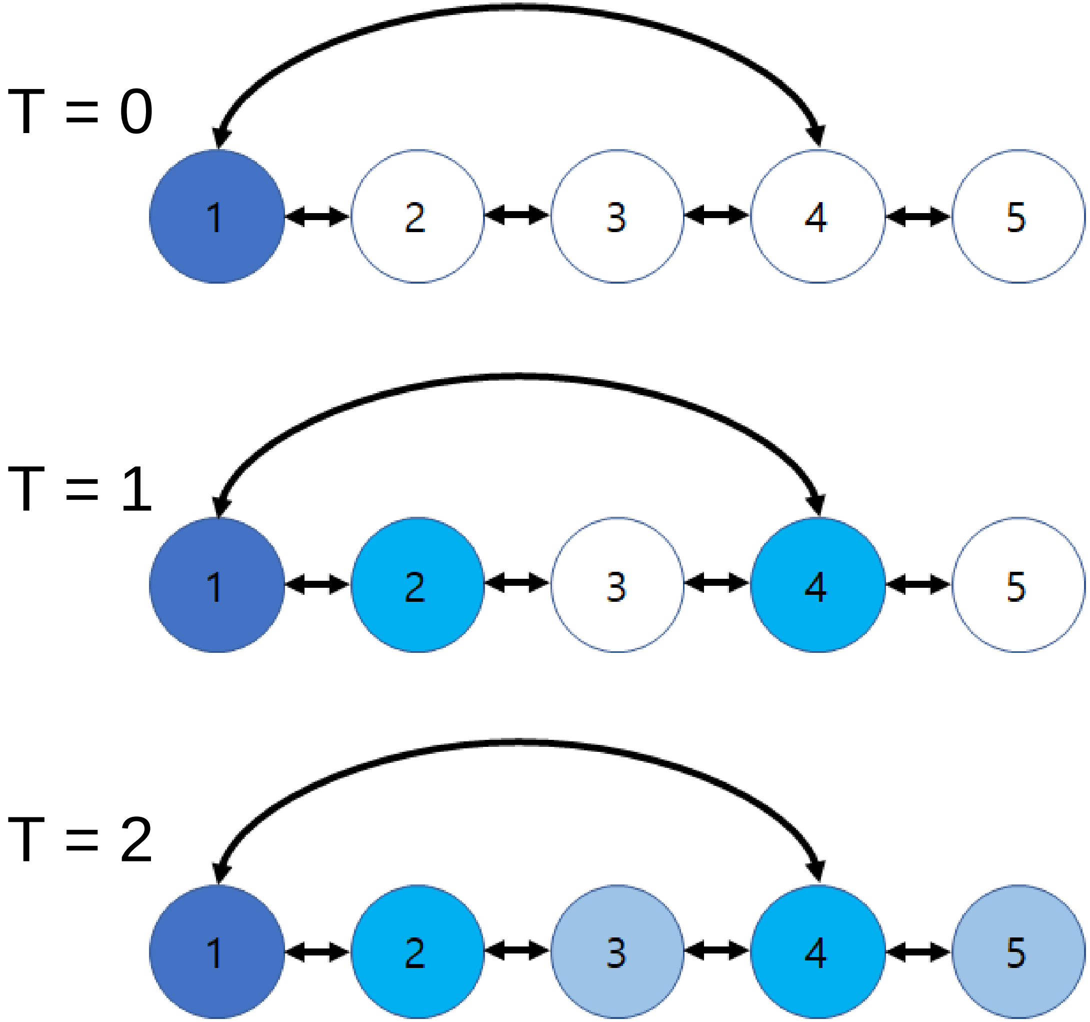




T = 0
1
2
3
4
5



T = 0
1
2
3
4
5
T = 1
1
2
3
4
5


T = 0
1
2
3
4
5
T = 1
1
2
3
4
5
T = 2
1
2
3
4
5

E. 텔레포트 정거장

- 점 S에서 점 E로 가는 최단시간을 구하는 문제.
- 그래프 모델링?
- 현재위치가 X라면 X의 텔레포트 연결 정보와 X+1, X-1로의 간선이 있는 그래프.
  각, 간선의 가중치 1이라서 BFS로 해결할 수 있음.
- 그래프 모델링? : 동적 배열, 링크드 리스트 etc.

시간복잡도 : O(N+M)　　　　분류 : 그래프, 최단경로

F. 러버덕을 사랑하는 모임

- P > N이라면 항상 불가능하다.

F. 러버덕을 사랑하는 모임

- P > N이라면 항상 불가능하다.
- N명중 P명을 고를 경우의 수 ${}_{N}C_{p}$. (${}_{20}C_{10}$ = 184,756가지. 많지 않다!)

F. 러버덕을 사랑하는 모임

- P > N이라면 항상 불가능하다.
- N명중 P명을 고를 경우의 수 $_NC_P$. ($_{20}C_{10}$ = 184,756가지. 많지 않다!)
- 가능한 모든 구성을 살펴보자!

F. 러버덕을 사랑하는 모임

- P > N이라면 항상 불가능하다.
- N명중 P명을 고를 경우의 수 $_{N}C_{P}$. ($_{20}C_{10}$ = 184,756가지. 많지 않다!)
- 가능한 모든 구성을 살펴보자!
- 내가 고른 P명이 가능하려면?

iff

$$L = \sum_{i \in P} x_i$$

$$R = \sum_{i \in P} y_i$$

$$L \leq E \leq R$$

즉, E가 P에 속한 인원의 하한의 합 이상 and 상한의 합 이하면 항상 가능하다.

이게 왜 될까?

iff

$$L = \sum_{i \in P} x_i$$

$$R = \sum_{i \in P} y_i$$

$$L \le E \le R$$

즉, E가 P에 속한 인원의 하한의 합 이상 and 상한의 합 이하면 항상 가능하다.

이게 왜 될까?

1. P명에게 각자의 하한만큼 미리 분배.
2. 개인의 상한을 넘지 않는 선에서 E와 같아질 때까지 인형 하나를 주는 것을 반복하면 조건을 위배하지 않고 항상 E에 도달할 수 있다.

F. 러버덕을 사랑하는 모임

- P > N이라면 항상 불가능하다.
- N명중 P명을 고를 경우의 수 ${}_{N}C_{P}$. (${}_{20}C_{10}$ = 184,756가지. 많지 않다!)
- 가능한 모든 집합을 살펴보자!
- 내가 고른 P명이 가능하려면? L ≤ E ≤ R
- 이 부등식을 만족하는 집합을 아무거나 하나 골라서 하한만큼 전부 나눠주고 추가로 분배!

시간복잡도 : O(2^N)

분류 : 완전탐색

G. 당근 훔쳐 먹기

- 1 ~ x-1일차에 당근 i를 k번 먹고, x일차에 당근 i를 먹는 경우,

G. 당근 훔쳐 먹기

- 1 ~ x-1일차에 당근 i를 k번 먹고, x일차에 당근 i를 먹는 경우,
- $(k+1) * w_i + (x-k-1) * p_i$가 된다. $w_i \leq p_i$ 이므로, k가 작을수록 좋다. 모든 당근에 대해 k=0인 경우만 고려하자. 즉, 어떤 당근이든 T일동안 최대 한번만 먹자!

G. 당근 훔쳐 먹기

- 1 ~ x-1일차에 당근 i를 k번 먹고, x일차에 당근 i를 먹는 경우,
- $(k+1) * w_i + (x-k-1) * p_i$가 된다. $w_i \leq p_i$ 이므로, k가 작을수록 좋다. 모든 당근에 대해 k=0인 경우만 고려하자. 즉, 어떤 당근이든 T일동안 최대 한번만 먹자!
- 최대한 많은 종류의 당근을 섭취하는 것이 유리.

G. 당근 훔쳐 먹기

- 1 ~ x-1일차에 당근 i를 k번 먹고, x일차에 당근 i를 먹는 경우,
- $(k+1) * w_i + (x-k-1) * p_i$가 된다. $w_i \leq p_i$ 이므로, k가 작을수록 좋다. 모든 당근에 대해 k=0인 경우만 고려하자. 즉, 어떤 당근이든 T일동안 최대 한번만 먹자!
- 최대한 많은 종류의 당근을 섭취하는 것이 유리.
- N ≤ T이므로, 모든 종류의 당근을 전부 섭취할 수 있다.

G. 당근 훔쳐 먹기

- 1 ~ x-1일차에 당근 i를 k번 먹고, x일차에 당근 i를 먹는 경우,
- $(k+1) * w_i + (x-k-1) * p_i$가 된다. $w_i \le p_i$ 이므로, k가 작을수록 좋다. 모든 당근에 대해 k=0인 경우만 고려하자. 즉, 어떤 당근이든 T일동안 최대 한번만 먹자!
- 최대한 많은 종류의 당근을 섭취하는 것이 유리.
- N ≤ T이므로, 모든 종류의 당근을 전부 섭취할 수 있다.
- T-N+1 ~ T일. 즉, 마지막 N일동안 당근을 몰아서 먹는 것이 좋다.

G. 당근 훔쳐 먹기

- 1 ~ x-1일차에 당근 i를 k번 먹고, x일차에 당근 i를 먹는 경우,
- $(k+1) * w_i + (x-k-1) * p_i$가 된다. $w_i \leq p_i$ 이므로, k가 작을수록 좋다. 모든 당근에 대해 k=0인 경우만 고려하자. 즉, 어떤 당근이든 T일동안 최대 한번만 먹자!
- 최대한 많은 종류의 당근을 섭취하는 것이 유리.
- N ≤ T이므로, 모든 종류의 당근을 전부 섭취할 수 있다.
- T-N+1 ~ T일. 즉, 마지막 N일동안 당근을 몰아서 먹는 것이 좋다.
- 이제 당근을 뽑아 먹을 순서를 정해야 한다. 당근 i를 X일차에 뽑아먹는다고 하면 $w_i + (X-1) * p_i$의 맛을 얻을 수 있으므로 w_i값에 상관 없이 p_i에 대해 정렬을 하여 p_i가 큰 당근을 나중에 뽑아먹도록 한다.

시간복잡도 : O(NlogN)　　분류 : 정렬, 탐욕법

H. 지금 만나러 갑니다.

- X일차에 이동해야 되는 거리는 2^{X-1}이다. 즉, X가 logN보다 크면 이동하는 거리가 항상 N보다 크므로 이동 불가능. 최대 logN일까지만 고려하면 된다.

H. 지금 만나러 갑니다.

- X일차에 이동해야 되는 거리는 2^{X-1}이다. 즉, X가 logN보다 크면 이동하는 거리가 항상 N보다 크므로 이동 불가능. 최대 logN일까지만 고려하면 된다.
- 하루에 오리와 육리의 움직임으로 가능한 것은 최대 4가지이다.
- 오리육리 이동방향을 화살표로 표현하면 (1) ←←, (2) →→, (3) ←→, (4) →←

H. 지금 만나러 갑니다.

- X일차에 이동해야 되는 거리는 2^{X-1}이다. 즉, X가 logN보다 크면 이동하는 거리가 항상 N보다 크므로 이동 불가능. 최대 logN일까지만 고려하면 된다.
- 하루에 오리와 육리의 움직임으로 가능한 것은 최대 4가지이다.
- 오리육리 이동방향을 화살표로 표현하면 (1) ←←, (2) →→, (3) ←→, (4) →←
- 모든 경우를 매번 탐색하면 시간복잡도는 O(4^{logN}) → 시간초과

H. 지금 만나러 갑니다.

- X일차에 이동해야 되는 거리는 2^{X-1}이다. 즉, X가 logN보다 크면 이동하는 거리가 항상 N보다 크므로 이동 불가능. 최대 logN일까지만 고려하면 된다.
- 하루에 오리와 육리의 움직임으로 가능한 것은 최대 4가지이다.
- 오리육리 이동방향을 화살표로 표현하면 (1) ←←, (2) →→, (3) ←→, (4) →←
- 모든 경우를 매번 탐색하면 시간복잡도는 O(4^{logN}) → 시간초과
- 오리와 육리의 움직임은 독립적! 오리가 X일에 위치Y에 존재할 수 있다는 정보, 육리가 A일에 위치B에 존재할 수 있다는 정보를 전부 구하고 저장하자.

H. 지금 만나러 갑니다.

- X일차에 이동해야 되는 거리는 2^{X-1}이다. 즉, X가 logN보다 크면 이동하는 거리가 항상 N보다 크므로 이동 불가능. 최대 logN일까지만 고려하면 된다.
- 하루에 오리와 육리의 움직임으로 가능한 것은 최대 4가지이다.
- 오리육리 이동방향을 화살표로 표현하면 (1) ←←, (2) →→, (3) ←→, (4) →←
- 모든 경우를 매번 탐색하면 시간복잡도는 O(4^{logN}) → 시간초과
- 오리와 육리의 움직임은 독립적! 오리가 X일에 위치Y에 존재할 수 있다는 정보, 육리가 A일에 위치B에 존재할 수 있다는 정보를 전부 구하고 저장하자.
- X=A, Y=B인 경우가 존재하면, 오리와 육리는 X일에 위치Y에서 만날 수 있음. 가장 빨리 만날 수 있는 경우를 구해서 출력.

H. 지금 만나러 갑니다.

- X일차에 이동해야 되는 거리는 2^{X-1}이다. 즉, X가 logN보다 크면 이동하는 거리가 항상 N보다 크므로 이동 불가능. 최대 logN일까지만 고려하면 된다.
- 하루에 오리와 육리의 움직임으로 가능한 것은 최대 4가지이다.
- 오리육리 이동방향을 화살표로 표현하면 (1) ←←, (2) →→, (3) ←→, (4) →←
- 모든 경우를 매번 탐색하면 시간복잡도는 O(4^{logN}) → 시간초과
- 오리와 육리의 움직임은 독립적! 오리가 X일에 위치Y에 존재할 수 있다는 정보, 육리가 A일에 위치B에 존재할 수 있다는 정보를 전부 구하고 저장하자.
- X=A, Y=B인 경우가 존재하면, 오리와 육리는 X일에 위치Y에서 만날 수 있음. 가장 빨리 만날 수 있는 경우를 구해서 출력.
- 각각을 독립적으로 생각하므로 O($2^{logN} = N$)만에 구할 수 있음.

시간복잡도 : O(N)　　　　분류 : 완전탐색, 시뮬레이션

H. 지금 만나러 갑니다.

다른 풀이?

- 오리육리 이동방향을 화살표로 표현하면 (1) ←←, (2) →→, (3) ←→, (4) →←
- D = 오리와 육리의 거리, K = 이동할 수 있는 거리 = 2^{x-1}
- 우리의 목표? D=0, D를 임의의 Q로 나눈 나머지가 0이게 하는 것.
- 맨처음에 K=1. (1) ←←, (2) →→ 인 경우 D에 변화가 없음.
- (3) ←→, (4) →←인 경우, D=K*2만큼 변화
- K는 매번 2배씩 증가하므로. K*4로 나눈 나머지를 0으로 맞춰주지 않으면 다음에는 항상 불가능.
- 즉, K*4로 나눈 나머지가 0이면 (1)과 (2)만 유효.
- K*4로 나눈 나머지가 K*2면 (3)과 (4)만 유효.
- 매번 가능한 경우는 2가지로 결정됨. O($2^{logN} = N$)

시간복잡도 : O(N)　　　　분류 : 완전탐색, 시뮬레이션