SUAPC 2023 Winter Solution

Official Solutions

신촌지역 대학생 프로그래밍 동아리 연합

2023년 2월 25일

목차

240

문제		의도한 난이도	출제자
A	카트라이더: 드리프트	Beginner	김태훈amel
B	신촌방위본부 탈출	Challenging	장래오leo020630
C	점수 내기	Medium	김태훈amel
D	듣기 평가 연습	Challenging	장래오leo020630
E	슬라이딩 퍼즐 마스터	Hard	김도훈dohoon
F	배너 걸기	Easy	박진식pjshwa
G	Azber is playing at Biou's house	Medium	장근영azberjibiou
H	양 가두기	Medium	김태훈amel
I	UFO in the Sinchon	Hard	박신욱bnb2011
J	치즈	Medium	박진식pjshwa
K	시계 맞추기	Easy	김태훈amel
L	따로 걸어가기	Hard	박진식pjshwa
M	좋은 문자열 만들기	Easy	장근영azberjibiou

A. 카트라이더: 드리프트

`sorting`, `string`
출제진 의도 – **Beginner**

- 제출 75번, 정답 42팀 (정답률 56.000%)
- 처음 푼 팀: **내가만든쿠키**연세대학교, 3분
- 출제자: 김태훈`amel`

- 입력이 등수 순서대로 주어지지 않기 때문에, 먼저 레이서들을 기록 순서대로 정렬하여야 합니다.
- 모든 기록의 길이가 같기 때문에 완주 기록을 **문자열**로 간주하여 정렬할 수 있습니다.
- 지문에 적혀있듯이, 리타이어가 없는 8인 경기에서는 순위 점수의 합계가 반드시 다릅니다.
- 따라서 레드팀과 블루팀의 순위 점수를 각각 더해 더 높은 팀의 이름을 출력하면 됩니다.
- 순위 점수는 배열을 하나 더 선언하여 "`index == 순위`"라 생각하면 편하게 구현할 수 있습니다.

B. 신촌방위본부 탈출

`flow`, `MCMF`, `binary_search`
출제진 의도 – **Challenging**

- ✓ 제출 6번, 정답 0팀 (정답률 00.000%)
- ✓ 처음 푼 팀: —[**대학교], 0분
- ✓ 출제자: 장래오[leo020630]

- 만약 복도의 이동 시간이 없다면 이 문제는 전형적인 MCMF 로 해결할 수 있습니다.
- 하지만 시간축의 존재가 이 문제를 어렵게 만듭니다.
- 정점을 각 단위시간에 대해 분할해봅시다.
- 이러면 서로 다른 시간에 있는 정점을 연결하는 것으로 복도의 이동 시간을 고려해줄 수 있습니다.
- 탈출 가능 인원의 최댓값을 먼저 구해봅시다.

- 아래와 같이 플로우 그래프를 구성하면 됩니다. S는 소스, T는 싱크, (t, i)는 t시점의 i 정점입니다. 조건에 의해 적어도 $N + T$ 시점에는 모든 정점이 불에 타 없어지므로 이 시점까지의 정점만 고려해줍시다.
- $S \to (0, i)$로 용량 P_i, 비용 0인 간선 추가
- $(t, i) \to (t + 1, i)$로 용량 ∞, 비용 0인 간선 추가
- $(t, N) \to T$로 용량 ∞, 비용 0인 간선 추가
- 각 i번째 간선에 대해
- $(t, u_i) \to (t + t_i, v_i)$로 용량 c_i, 비용 x_i인 간선 추가
- $(t, v_i) \to (t + t_i, u_i)$로 용량 c_i, 비용 x_i인 간선 추가

240

- 탈출 가능한 최대 인원을 구했으니 이 인원이 모두 탈출 가능한 최소 시간을 구해봅시다.
- 이는 이분 탐색으로 구할 수 있습니다.
- 탈출 시간이 길어질수록, 즉 정점을 많이 사용할수록 탈출 인원이 증가함이 자명하니 이분 탐색으로 최대 인원과 탈출 인원이 같아지는 시점을 구할 수 있습니다.
- 이 시점에서의 MCMF 결과값이 최소 피로도가 됩니다.
- MCMF를 $\log(N+T)$ 번 수행하는 풀이로 아슬아슬하게 통과할 수 있습니다.

- 여기까지로도 **맞았습니다**를 받을 수 있지만, 시간을 더 줄여봅시다.
- 사실 이분 탐색에는 탈출 인원만이 필요하므로, 이분 탐색에는 MCMF가 아닌 일반 플로우를 사용하고 마지막에만 MCMF를 구해주어도 됩니다.
- 더 나아가, 이분탐색을 사용하지 않는 풀이도 있습니다.
- 각 시간에 대해 최대 유량을 구하되, 이전 시점에서의 플로우 그래프에서 정점을 추가하며 최대 유량을 찾으면 시간을 단축할 수 있습니다.
- 이렇게 해도 결과적으로 증가 경로를 총 P번 찾기 때문에 이 풀이는 유효합니다.

C. 점수 내기

`trie`, `string`
출제진 의도 – **Medium**

- 제출 13번, 정답 0팀 (정답률 00.000%)
- 처음 푼 팀: —**대학교, 0분
- 출제자: 김태훈amel

240

- ✓ prefix list와 suffix list를 독립적으로 생각할 수 있습니다.
  - prefix list만 있다고 할 때 최대 점수를 얻을 수 있는 문자열을 A라 합시다.
  - suffix list만 있다고 할 때 최대 점수를 얻을 수 있는 문자열을 B라 합시다.
  - 두 문자열 사이에 아무 숫자나 넣어서 $A0B$와 같은 문자열을 만들면 최대 점수를 얻을 수 있습니다.
- ✓ 따라서 + 및 - 쿼리는 prefix list와 suffix list에서의 결과를 단순히 더해주면 됩니다.

- ✓ 앞서 봤듯이 prefix list와 suffix list를 독립적으로 생각할 수 있습니다.
  - suffix list는 입력 문자열을 뒤집어서 처리해야 함에 유의합시다.
- ✓ 최대 점수를 구하는 것과 최소 점수를 구하는 것 또한 독립적으로 생각할 수 있습니다.
  - 최소 점수는 list의 모든 점수에 -1을 곱한 뒤 최대 점수를 구하는 방식으로 구할 수 있습니다.
- ✓ 따라서 prefix list에서 최대 점수를 구할 수 있으면 전체 문제를 해결할 수 있습니다.
  - 점수를 출력할 때 prefix list와 suffix list에서의 결과를 더해야 함에 유의합시다.

240

- ✓ prefix list에서 최대 점수를 구하는 방법은 여러 가지가 있는데, 이 중 가장 쉬운 풀이를 소개합니다.
- ✓ 입력되는 문자열의 길이 합이 $1\,000\,000$을 넘지 않으므로 Trie를 구성할 수 있습니다.
- ✓ Trie의 각 노드에 점수와 관련된 변수들을 같이 저장하여 관리할 것입니다.
  - `score`: 해당 노드가 나타내는 문자열의 점수 (단, 문자열이 list에 없다면 0)
  - `mx` : 해당 노드의 서브트라이(자신 포함) 중 최대 점수

C. 점수 내기

240

- ✓ 업데이트 쿼리의 문자열이 A, 점수가 X 라 합시다.
- ✓ 문자열 A 를 Trie에 추가하면서 아래의 작업들을 수행해 줍시다.
  - 일단 A 를 Trie에 추가합니다.
  - A 에 해당하는 노드의 `score` 를 X 로 바꿔줍니다.
  - 그 다음 다시 root로 되돌아오면서, 모든 노드에 대하여 `mx` 를 업데이트 해줍니다.
    - 직계 자식의 `mx` 최댓값에 현재 노드의 `score` 를 더해주면 됩니다.
- ✓ 이렇게 하면 root의 `mx` 가 항상 최대 점수가 됩니다.

240

- ✓ 입력되는 문자열 길이 합을 L 이라 하면, Trie 구성은 $\mathcal{O}(26L)$ 의 시간이 걸립니다.
- ✓ 점수를 출력하는 쿼리는 root만 참조하면 되므로 $\mathcal{O}(1)$ 시간이 걸립니다.
- ✓ 따라서 전체 문제를 $\mathcal{O}(26L + M)$ 시간에 해결할 수 있습니다.

D. 듣기 평가 연습

`suffix_array`, `binary_serach`, `segment_tree`
출제진 의도 – **Challenging**

- ✓ 제출 19번, 정답 0팀 (정답률 00.000%)
- ✓ 처음 푼 팀: —[**대학교], 0분
- ✓ 출제자: 장래오[leo020630]

240

- 문자열 S의 i번째 문자부터 j번째 문자까지를 포함하는 부분 문자열을 $S[i, j]$라 합시다.
- 두 문자열이 k-유사하다는 것은 두 문자열의 최장 공통 접두사의 길이가 k라는 것과 동치입니다.
- 두 접미사 $S[i, n]$과 $S[j, n]$의 최장 공통 접두사 길이는 Suffix Array를 통해 구할 수 있는 LCP 배열과 range min segment tree를 이용해 구할 수 있습니다.
- $S[i, n]$과 $S[j, n]$의 최장 공통 접두사 길이가 k라면 Suffix Array 상에서 둘 사이에 있는 모든 접미사도 $S[i, n], S[j, n]$과 k글자 이상이 같아야 합니다.
- 즉, LCP 배열 상에서 $S[i, n]$의 위치와 $S[j, n]$의 위치 사이의 최솟값이 두 접미사의 최장 공통 접두사 길이가 됩니다.

240

- ✓ 쿼리로 i, j, k가 주어졌을 때, $S[x, n]$과 $T[i, m]$의 최장 공통 접두사 길이를 $f(x)$라 합시다.
- ✓ 우선 $k > j - i + 1$일 경우 답은 0이므로 $k \leq j - i + 1$라 가정합시다.
- ✓ $f(x)$와 k의 관계는 다음 셋 중 하나입니다.
  - $f(x) < k$: 이 경우 $S[x, n]$의 접두사는 답의 후보가 될 수 없습니다.
  - $f(x) = k$: 이 경우 $x + k - 1 \leq y \leq n$인 모든 y에 대해 $S[x, y]$가 답이 될 수 있습니다.
  - $f(x) > k$: 이 경우 다시 둘로 나뉩니다.
    - $k = j - i + 1$: 이 경우 $f(x) = k$와 같은 방식으로 처리해줄 수 있습니다.
    - $k < j - i + 1$: 이 경우 $S[x, x + k - 1]$만이 답이 될 수 있습니다.

- ✓ $f(x)$는 S와 T를 이어붙인 문자열에서의 LCP 배열에서 range minimum query를 처리하는 것으로 구해줄 수 있습니다.
- ✓ $f(x)$는 LCP 배열에서의 range minimum 값이기 때문에 i의 위치를 기준으로 양방향으로 단조감소합니다.
- ✓ 따라서 각 경우는 Suffix Array에서 최대 2개의 구간으로 나타납니다.
- ✓ $f(x) = k$인 구간에 대해서는 $n - (x + k - 1) + 1$의 구간 합을 구해주어야 하며, $f(x) > k$인 구간에 대해서는 1의 구간 합을 구해주어야 합니다. 이 때, 초기에 S에 해당하는 접미사에 대해서만 더해주어야 한다는 점에 유의합시다.
- ✓ 추가로 $k = 0$이며 $f(x) = k$인 경우에 예외가 발생하기 때문에 이를 잘 처리합시다.

- ✓ range minimum과 range sum 모두 세그먼트 트리를 이용해 처리해줄 수 있으며, 업데이트가 없다는 점을 이용해 range sum은 누적합 배열으로, range minimum은 sparse table을 이용해 처리해줄 수도 있습니다.
- ✓ 시간 복잡도는 구현 방법에 따라 $O((N+Q)\log^2 N)$ 또는 $O((N+Q)\log N)$ 입니다.

240

E. 슬라이딩 퍼즐 마스터

`constructive`, `recursion`
출제진 의도 – **Hard**

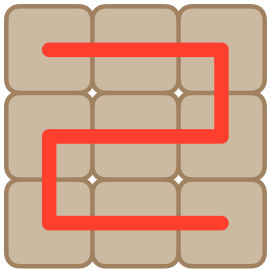
- 제출 14번, 정답 3팀 (정답률 21.429%)
- 처음 푼 팀: **연세대학**연세대학교, 243분
- 출제자: 김도훈dohoon

240

- ✓ 사실 기존 슬라이딩 퍼즐의 시행의 빈칸은 그냥 일반 퍼즐 조각으로 생각해도 똑같습니다.
- ✓ 그럼 문제는 '$N \times M$ 격자에서 인접한 두 조각을 교환하는 시행으로 모든 배치를 만들기'로 정리됩니다.

240

- 꼭 인접한 네 방향과의 교환을 모두 해야 할까요?
  - 그렇지 않습니다.
  - 다음 그림에 그려진 경로 방향에 맞는 인접한 조각들만 교환한다고 강제해봅시다.



240

- 그럼 다음 그림과 같이 **직선형 배열**로 다룰 수 있습니다.
- 따라서 인접한 교환으로 모든 순열을 만드는 문제로 바뀝니다.

- 길이가 3인 직선형 배열에서의 규칙을 관찰해봅시다.
  - $[1, 2, \mathbf{3}]$
  - $[1, \mathbf{3}, 2]$
  - $[\mathbf{3}, 1, 2]$
  - $[\mathbf{3}, 2, 1]$
  - $[2, \mathbf{3}, 1]$
  - $[2, 1, \mathbf{3}]$
- 어떤가요? 3의 위치가 연속적으로 바뀌는 구간에서의 순열들은 3을 무시하면 전부 같습니다.
- 즉, 3을 한쪽 끝에서 다른 쪽까지 움직였다가, 한 번 멈춰서 3을 제외한 순열에 변화를 주고, 다시 3을 반대쪽 끝으로 움직이고 있습니다.

- ✓ 따라서 이 규칙을 통해 길이가 k일 때 모든 순열을 만드는 방법을 갖고 길이가 $k+1$일 때에도 모든 순열을 만들 수 있을 거라고 예상할 수 있습니다.
- ✓ 수학적 귀납법으로 이를 증명해봅시다.
- ✓ 증명에 앞서 인접한 두 원소의 교환 한 번으로 만들 수 있는 순열의 관계를 **인접하다**고 하겠습니다.

240

- ✓ 길이가 1 인 배열일 때는 자명하게 성립합니다.
- ✓ 길이가 k 인 배열에서 조건에 맞게 배치할 수 있다고 가정할 때, 길이가 $k+1$ 인 배열에서도 조건에 맞게 배치할 수 있다는 것을 증명하면 됩니다.
  - ▶ 홀수 번째로 나타나는 순열 $[p_1, p_2, \ldots, p_k]$ 에 대해, $k+1$ 을 오른쪽 끝에서 시작해서 왼쪽 끝까지 순서대로 끼워넣습니다.
  - ▶ 짝수 번째로 나타나는 순열 $[q_1, q_2, \ldots, q_k]$ 에 대해, $k+1$ 을 왼쪽 끝에서 시작해서 오른쪽 끝까지 순서대로 끼워넣습니다.
  - 길이 k 인 순열들은 원래 서로 달랐고, 각 순열에서 $k+1$ 을 끼워넣는 위치가 모두 달라서 결과적으로 생성된 모든 순열은 **서로 다릅니다**.
  - $k+1$ 의 위치가 고정되는 차례에는 나머지 부분이 길이가 k 인 인접한 순열로 바뀌기 때문에, 생성된 배치에서 모든 순열은 다음 순열과 **인접합니다**.

240

- ✓ 이제 문제를 푸는 방법을 알고, 증명도 끝났습니다.
- ✓ 위 규칙대로 구현을 잘 하면, 길이가 n 일 때 $O(n! \times n)$ 만에 문제를 풀 수 있습니다.
- ✓ 따라서 원래 문제 기준으로 $O((NM)! \times NM)$ 에 문제를 풀 수 있습니다.

F. 배너 걸기

`greedy`, `sliding_window`
출제진 의도 – **Easy**

- 제출 215번, 정답 33팀 (정답률 15.349%)
- 처음 푼 팀: **내가만든쿠키**연세대학교, 4분
- 출제자: 박진식`pjshwa`

240

- ✓ 높이 H 를 $\lceil \frac{9M}{10} \rceil$ 개 이상 포함하는 길이 M 의 구간이 있다고 가정하고, 이 구간을 $[i, i+M-1]$ 이라고 해봅시다.
- ✓ 위 구간에서 H 가 마지막으로 등장하는 인덱스를 r 이라 할 경우, $[max(1, r-M+1), r]$ 또한 H 를 적어도 $\lceil \frac{9M}{10} \rceil$ 개 이상 포함함을 알 수 있습니다.
  - 다른 말로, 답이 될 수 있는 모든 길이 M 의 구간에 대해, H 가 오른쪽 끝점에 위치하는 길이 M 이하의 또다른 구간이 존재합니다.

240

- ✓ 따라서 모든 $r\ (1 \leq r \leq N)$에 대하여, 구간 $[max(1, r-M+1), r]$가 $A[r]$를 $\lceil \frac{9M}{10} \rceil$개 이상 포함하는 경우가 있는지만 확인하면 됩니다.
  - 값들의 개수는 sliding window 기법을 이용해 관리할 수 있습니다.

G. Azber is playing at Biou's house

`dp`, `game_theory`
출제진 의도 – **Medium**

- 제출 32번, 정답 11팀 (정답률 34.375%)
- 처음 푼 팀: **연세대학**연세대학교, 53분
- 출제자: 장근영azberjibiou

G. Azber is playing at Biou's house

240

- $dp_0[i][j] = i$번 노드에서 j번 사람이 시작했을 때의 최종 점수라고 합시다. $j = 0$이면 Azber, $j = 1$이면 Biou를 뜻합니다.
- 로봇이 한 번 왼쪽으로 움직이기 시작하면 위로 올라가기 전까지는 계속 왼쪽으로만 움직여야 합니다. 오른쪽 방향에 대해서도 마찬가지입니다.
- i번 노드에서 오른쪽으로 움직일 수 없는 경우의 최종 점수 $dp_1[i][j]$와 왼쪽으로 움직일 수 없는 경우의 최종 점수 $dp_2[i][j]$를 정의해줍시다.
- $dp_0[i][j] = f(dp_1[i][j], dp_2[i][j])$입니다. 여기에서 f는 Azber가 플레이어면 min이고, Biou가 플레이어면 max입니다.

- $dp_1[i][j] = f(dp_1[i-1][1-j], dp_0[\lfloor \frac{i}{2} \rfloor][1-j], 0) + A_i$ 으로 나타납니다. dp_2 에 대해서도 비슷한 식을 세울 수 있습니다.
- 이제 모든 dp 값을 구할 수 있습니다. dp 배열을 채우는 순서에 주의하세요. 0 층에서부터 차례대로 dp 배열을 채우면 됩니다. 각 층에 대해서 dp_1 은 왼쪽에서 오른쪽으로, dp_2 는 오른쪽에서 왼쪽으로 구한 뒤에 dp_0 배열을 채워나가면 됩니다.
- 시간복잡도는 $O(2^n)$ 입니다. n 이 크면 dp 테이블을 만들 때의 생각이 제한될 것 같아서 n 을 작게 주었습니다.

H. 양가두기

`sweeping`

출제진 의도 – **Medium**

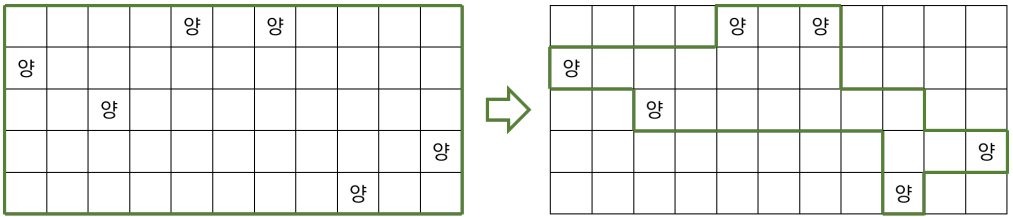
- ✓ 제출 66번, 정답 4팀 (정답률 6.061%)
- ✓ 처음 푼 팀: **내가만든쿠키**연세대학교, 102분
- ✓ 출제자: 김태훈`amel`

240

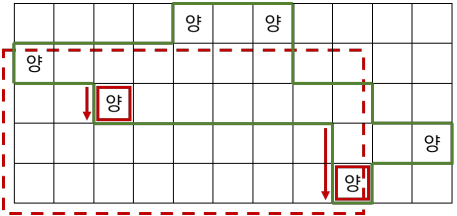
- ✓ 먼저 울타리의 최소 개수를 구해봅시다.
- ✓ 양들을 모두 감싸면서 가장 작은 직사각형 모양의 우리가 최적입니다.
- ✓ 오목한 모양보다 볼록한 모양이 개수가 더 적고, 볼록한 모양의 도형은 항상 둘레가 같은 직사각형과 대응시킬 수 있기 때문입니다.
- ✓ $\mathcal{O}(N)$ 시간에 x, y 좌표가 가장 크거나 가장 작은 양을 찾아 둘레를 계산할 수 있습니다.

240

- 이제 울타리를 최소 개수만큼 사용할 때, 우리의 최소 넓이를 구해봅시다.
- 반드시 볼록한 직각다각형 형태를 지녀야 최적임을 관찰할 수 있습니다.
- 이 때, 위에서 구한 직사각형에서 필요 없는 꼭짓점 부분을 깎아 최적의 우리 모양을 만듭시다.



- 왼쪽 아래 부분을 살펴보면, 깎이는 높이가 왼쪽에서 오른쪽으로 가면서 단조감소합니다.
  - 이는 우리가 볼록하기 때문에 만족하는 특징입니다.
- 최대한 많이 깎아야 하기 때문에 더 낮은 양이 등장하기 전까지는 높이를 유지하다가, 양의 바로 아래까지 깎는 것이 가장 좋습니다.



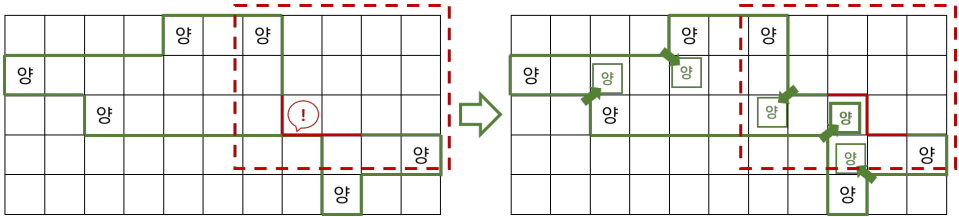
H. 양 가두기

240

- 양의 위치는 10^9 까지 가능하기 때문에 깎을 높이를 한 칸씩 살펴보면 안됩니다.
- 양을 정렬한 뒤 sweeping하여 현재보다 낮은 높이의 양이 등장할 때마다 얼마나 깎을지 계산해 줍시다.
- 이를 4개의 꼭짓점에 대하여 반복해 줍시다.

240

- 그런데 한 가지 문제가 있습니다.
- 두 꼭짓점에서 깎은 부분이 겹쳐 우리가 분리될 수 있습니다.
- 이를 방지하려면 깎아줄 때마다 꼭짓점 칸에 dummy 양을 추가해주면 됩니다.
  - 적어도 양 하나가 지나갈 통로는 확보되어야 한다는 의미를 가집니다.



240

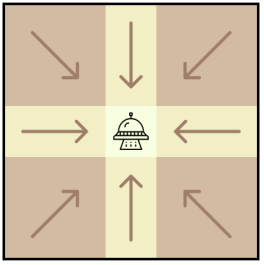
- ✓ 깎는 시행은 최대 N 번 수행되기 때문에 추가되는 dummy 양도 최대 N 마리 입니다.
- ✓ 전체 문제를 해결하는데 필요한 시간복잡도는 $\mathcal{O}(N \log N)$ 입니다.
  - 직사각형의 둘레를 구하는데 $\mathcal{O}(N)$
  - dummy 양의 개수 $\mathcal{O}(N)$
  - 총 $\mathcal{O}(N)$ 개의 양을 정렬하는데 $\mathcal{O}(N \log N)$
  - 양을 sweeping하는데 $\mathcal{O}(N)$

I. UFO in the Sinchon

`segtree`, `lazyprop`
출제진 의도 – **Hard**

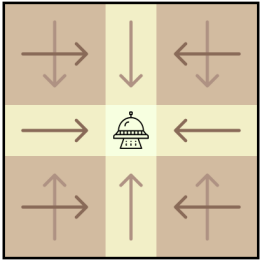
- ✓ 제출 28번, 정답 2팀 (정답률 7.143%)
- ✓ 처음 푼 팀: **내가만든쿠키**연대학교, 194분
- ✓ 출제자: 박신욱`bnb2011`

✓ UFO와의 상대적인 위치에 따라 사람들이 이동하는 방향을 먼저 살펴봅시다.



- ✓ 지문에서 명시된 대로 UFO와 이미 같은 위치에 있는 사람은 움직이지 않습니다.
- ✓ UFO와 동일한 x 좌표 또는 y 좌표를 가지는 경우에는 UFO에 가까워지는 수직 또는 수평 방향으로 이동하는 것이 택시 거리가 가장 가까워지는 방향입니다.
- ✓ 이외의 경우에는 UFO에 가까워지는 방향의 대각선으로 이동하는 것이 택시 거리가 가장 가까워지는 방향입니다.
- ✓ 이렇게 가장 가까워지는 방향은 모든 경우에 대해 유일합니다.

✓ 대각선으로 이동하는 것은 처리하기 번거롭습니다. 아래 그림과 같이 x 축 이동과 y 축 이동을 분리해서 생각해봅시다.



- ✓ 방향을 분리해놓고 보면 UFO와의 상대적인 x 좌표와 y 좌표 위치에 따라 이동해야 하는 x 축 방향과 y 축 방향이 동일하다는 것을 알 수 있습니다.
- ✓ 그리고 x 축 방향으로의 이동과 y 축 방향으로의 이동이 독립적으로 이루어지므로, 이 둘을 분리해서 고려하는 것이 가능해집니다.
- ✓ 이후 슬라이드에서는 x 축 방향의 이동에 대해서만 서술합니다.

- ✓ 사람들의 x 좌표만 추출한 뒤, 정렬한 배열을 X 라고 합시다.
- ✓ UFO가 나타난 위치의 x 좌표가 x_j 이고 t_j 초 동안 나타났을 때, 사람들의 위치는 다음과 같이 변합니다.

$$\begin{cases} X_i + t_j & \text{if } X_i \le x_j - t_j \\ x_j & \text{if } x_j - t_j \le X_i \le x_j + t_j \\ X_i - t_j & \text{if } x_j + t_j \le X_i \end{cases}$$

- ✓ 이는 구간에 특정 값을 더하는 연산과, 구간을 특정 값으로 만드는 연산을 지원하는 Segment Tree with Lazy Propagation을 사용하면 해결할 수 있습니다.

- ✓ 구간에 따라 서로 다른 연산을 진행해야 하니, 구간을 나누는 경계를 이분 탐색으로 잘 찾아주어야 합니다.
- ✓ Segment Tree의 Query를 이용해서 이분 탐색을 하면 전체 문제를 $O(Qlog^2K)$, Segment Tree 내에서 이분 탐색을 진행 하면 $O(QlogK)$에 문제를 해결할 수 있습니다.
- ✓ 아쉽지만 $O(QlogK)$ 풀이만 통과 가능하도록 시간 제한을 설정했습니다.

J. 치즈

`graphs`, `dynamic_programming`
출제진 의도 – **Medium**

- ✓ 제출 155번, 정답 11팀 (정답률 7.097%)
- ✓ 처음 푼 팀: **혼혈 서강..준**서강대학교, 131분
- ✓ 출제자: 박진식`pjshwa`

240

- 치즈 i에 대해, i를 제공하는 상점이 최대 2곳이며, 따라서 적어도 두 곳 중 하나를 방문해야 한다는 사실을 알 수 있습니다.
- N개 종류의 치즈를 각각 그래프의 정점으로 보고, 치즈 u, v를 파는 상점을 정점 u, v를 잇는 간선으로 봅시다. 이 때, 모든 정점들은 크고 작은 cycle에 속해 있게 됩니다.
- Cycle 안에 속해 있는 정점 (치즈) 각각에 대해서는, 적어도 연결된 간선 둘 중 하나를 골라야 합니다.
- 그래프 형태의 특수성을 이용해, 간선을 고르는 최소비용을 다음과 같이 계산할 수 있습니다.

240

- Cycle 각각에 대해서 특정 간선 하나를 골라,
  - 1. 해당 간선을 포함하는 경우
  - 2. 해당 간선을 포함하지 않는 경우
- 각각에 대하여 최소비용을 구할 것입니다. 그래프에서 해당 간선을 제외하면 일자 형태가 되고, 나머지 간선들의 선택 여부는 일자 형태의 그래프 위에서 DP로 확인할 수 있습니다.
  - BOJ 17404. RGB거리 2 문제를 참고하세요.
- 각각에 대해서 최소비용을 구했다면, 정답은 1, 2번 중 더 작은 값이 됩니다.

K. 시계 맞추기

`bruteforcing`
출제진 의도 – **Easy**

- 제출 164번, 정답 41팀 (정답률 11.585%)
- 처음 푼 팀: **Ooo**서강대학교, 52분
- 출제자: 김태훈`amel`

- ✓ 시간 간격 R을 고정하면 그에 대응하는 N을 쉽게 구할 수 있습니다.
- ✓ 모든 기록을 첫 번째 기록의 시각으로 동기화시켜줍니다.
  - i번째 기록에서 $(i-1) \times R$만큼의 시간을 빼주면 됩니다.
  - C/C++, Java의 경우 음수의 modular 처리를 조심해야 합니다.
  - 혹은 $(M-i) \times R$만큼의 시간을 더해 마지막 기록의 시각과 동기화시켜도 됩니다.
- ✓ 이제 서로 다른 시각이 몇 가지 있는지 세어주면 N을 구할 수 있습니다.
  - 이는 배열에 직접 세거나 `std::set` 등을 이용하여 구할 수 있습니다.

- ✓ 이제 가능한 모든 시간 간격 R을 시도해 봅시다.
- ✓ 시계는 12시간$(= 720$분$)$마다 똑같은 시각을 가리키기 때문에, $R > 720$인 경우는 $R' = R - 720$인 경우와 완전히 동일합니다.
- ✓ 따라서 시도해보아야 할 R은 1 이상 720 이하의 정수로 720가지 뿐입니다.
- ✓ 하나의 R에 대해 $\mathcal{O}(M + 720)$ 또는 $\mathcal{O}(M \log M)$ 시간에 N을 구할 수 있습니다.
- ✓ 따라서 전체 문제를 $\mathcal{O}(720(M + 720))$ 또는 $\mathcal{O}(720M \log M)$ 시간에 해결할 수 있습니다.

L. 따로 걸어가기

`combinatorics`
출제진 의도 – **Hard**

- ✓ 제출 13번, 정답 3팀 (정답률 23.077%)
- ✓ 처음 푼 팀: **Make Yonsei Great Again**연세대학교, 224분
- ✓ 출제자: 박진식`pjshwa`

- ✓ 토끼 부부는 처음에 $(1, 1)$ 에서, 한 마리는 $(1, 2)$ 로, 다른 한 마리는 $(2, 1)$ 로 이동합니다. 이 경우 $(1, 2)$ 로 이동한 토끼가 도착점으로 $(N - 1, M)$ 으로부터 이동하며, $(2, 1)$ 로 이동한 토끼가 도착점으로 $(N, M - 1)$ 으로부터 이동합니다.
- ✓ 그 외의 부부의 이동은 아래 4가지로 분류할 수 있습니다.
  - 1. 토순이가 오른쪽, 토준이도 오른쪽
  - 2. 토순이가 오른쪽, 토준이는 아래쪽
  - 3. 토순이가 아래쪽, 토준이는 오른쪽
  - 4. 토순이가 아래쪽, 토준이도 아래쪽

- ✓ 일반성을 잃지 않고 $N \leq M$ 이고, 토준이가 처음에 오른쪽으로 이동하고 토순이가 아래쪽으로 이동했다고 가정합니다. 첫 이동과 마지막 이동은 고정되어 있으므로 나머지 이동에 대해서만 생각할 것입니다.
- ✓ 두 토끼가 중간에 만나지 않으려면, 어느 순간에든 토순이가 아래쪽으로 이동한 횟수가 토준이가 아래쪽으로 이동한 횟수보다 많거나 같아야 하며, 이동을 마쳤을 때 두 토끼가 아래쪽으로 이동한 횟수의 합이 $N-2$ 이어야 합니다.
  - 이 이후로 계속하여 처음과 마지막 이동은 제외하고 계산하고 있음에 유의하세요.

- ✓ 이 이동의 성질은 앞에서 언급된 4가지 이동중 3번을 여는 괄호, 2번을 닫는 괄호라고 생각하면 올바른 괄호 문자열 개수 구하기 문제와 비슷합니다. 2, 3번 이동의 횟수는 정확히 같아야 하고, 어떤 시점에도 2번 이동 횟수가 3번 이동 횟수보다 많아지면 안된다는 점을 관찰합시다.
- ✓ 하지만 1, 4의 이동이 있기 때문에 괄호 문자열 구하는 경우의 수와 완전히 같지는 않습니다.

- 2, 3번 이동 횟수를 a $(0 \le a \le N-2)$라고 합시다. a가 정해지면, 아래쪽으로 이동한 횟수의 합이 $N-2$어야 한다는 조건에 의해 4번 이동 횟수도 정해지고, 집으로 최단거리로 움직이게 되는 이동의 특성상 총 이동 횟수 또한 고정되어있기 때문에 1번 이동 횟수도 정해집니다.
- 2, 3번 이동 횟수가 정해지면, 이들의 순서를 올바르게 결정하는 경우의 수는 $C(a)$ (C는 카탈란 수)입니다. 4번과 1번은 순서 제한이 없기 때문에, 각각을 미리 정해진 2, 3번 이동들 사이에 어떻게 넣을 지만 중복 조합으로 결정하면 됩니다.
- 가능한 a에 대해 경우의 수를 구하여 모두 더해준 뒤, 이 값에 처음과 마지막 이동으로 가능한 경우의 수 2를 곱해준 값이 정답이 됩니다.

M. 좋은 문자열 만들기

`ad_hoc`, `prefix_sum`
출제진 의도 – **Easy**

- 제출 159번, 정답 41팀 (정답률 15.723%)
- 처음 푼 팀: **혼혈 서강..준**[서강대학교], 12분
- 출제자: 장근영[azberjibiou]

240

- ✓ 좋은 문자열이 될 수 있는 n의 조건에 대해서 생각해봅시다.
- ✓ $n = 1$일 때와 $n = 3$일 때에는 좋은 문자열을 만들 수 없습니다.
- ✓ $n = 2$일 때는 10이라는 좋은 문자열이 있습니다.
- ✓ $n \geq 4$일 때는 $10 \ldots 10$이라는 좋은 문자열이 있습니다.

240

- 좋은 문자열의 조건에 대해서 알아봅시다.
- 양끝 문자가 같으면 좋은 문자열이 될 수 없습니다. 일반성을 잃지 않고 첫번째 문자가 0, 마지막 문자가 1 이라고 합시다.
- 0 인 마지막 글자를 x 번째 글자, 1 인 첫 글자를 y 번째 글자라고 합시다. $x - 1 = n - y$ 가 성립해야 좋은 문자열이라고 할 수 있습니다.
- 이를 다른 말로 표현하자면 앞에 연속한 0 의 개수와 뒤에 연속한 1 의 개수가 같다는 것입니다.

240

- 형태로 표현하자면 $(0 \dots 0)1 \dots 0(1 \dots 1)$ 의 형태입니다.
- n 이 짝수이고 $k = \frac{n}{2}$ 인 경우에 $0 \dots 01 \dots 1$ 을 빼먹지 맙시다.
- 각 형태를 만드는데 필요한 비용을 prefix sum으로 $O(1)$ 에 구할 수 있습니다.
- 가능한 좋은 문자열 형태의 가짓수는 $O(n)$ 개 밖에 없기 때문에 전체 문제를 $O(n)$ 에 풀 수 있습니다.

240

- 다른 방법으로도 문제를 풀 수 있습니다.
- $n \geq 4$인 경우에는 $10\ldots10$과 $01\ldots01$이 답이라는 것을 관찰합시다.
- 첫번째 형태로 만드는데 필요한 비용과 2번째 형태로 만드는데 필요한 비용의 합은 4입니다.
- 따라서 두 형태 중 하나는 만드는데 필요한 비용이 2 이하입니다.

- ✓ 비용을 0 사용할 때, 1 사용할 때 좋은 문자열이 되는지를 직접 시뮬레이션할 수 있습니다.
- ✓ std::set을 사용하면 각 경우에 대해서 $O(logn)$에 시뮬레이션할 수 있습니다.
- ✓ 각 문자에 대해서 앞뒤 2개의 위치만 셋으로 관리한다면 각 경우에 대해서 $O(1)$에 시뮬레이션할 수 있습니다.
- ✓ 따라서 전체 문제를 $O(n)$ 또는 $O(nlogn)$에 풀 수 있습니다.
- ✓ 많은 분들이 틀리신 94퍼 쯤에 있는 데이터는 n을 1부터 8까지 가능한 모든 문자열을 넣은 데이터입니다.