제3회 보라매컵 예선 풀이

2024년 1월 21일

문제		의도한 난이도	출제자
A	대한민국을 지키는 가장 긴 힘	**Easy**	99asdfg
B	사격	**Easy**	kybs0627
C	훈련	**Medium**	kybs0627
D	축구 대회	**Hard**	99asdfg
E	시간 외 근무 멈춰!!!	**Hard**	99asdfg
F	물자 조달	**Challenging**	kybs0627

A. 대한민국을 지키는 가장 긴 힘

greedy
출제진 의도 – **Easy**

- 제출 256번, 정답 76명 (정답률 29.688%)
- 처음 푼 사람: **p_ce1052**, 3분
- 출제자: 99asdfg

A. 대한민국을 지키는 가장 긴 힘

Subtask 1

- $N = 3$ 이므로, 입력된 S 를 정수로 입력받아 S 가 641 이하라면 1, 아니라면 2 를 출력하면 됩니다.

Subtask 2

- S 의 각 자릿수가 1 이상 5 이하이므로, 앞에서부터 순서대로 3 자리씩 잘라서 읽어도 됩니다.
- 따라서, $\lfloor \frac{N}{3} \rfloor$ 를 출력하면 됩니다.

A. 대한민국을 지키는 가장 긴 힘

Subtask 3

- ✓ S에 0이 등장하지 않으므로, S의 모든 지점이 새로운 수의 첫 번째 자리수가 될 수 있습니다.
- ✓ 또한, 마지막 부분을 제외하면 S를 세 자리 수 혹은 두 자리 수로만 구성할 수 있습니다.
- ✓ 이때, 세 자리수로 읽을 수 있는 지점에서 두 자리수로 읽었을 때 얻을 수 있는 이득은 항상 손해보다 작습니다.
- ✓ 따라서, 앞에서부터 차례대로 세 자리씩 끊어보며, 세 자리수가 641 이하라면 세 자리로 끊고, 아니라면 앞의 두 자리수로만 끊는 것을 구현하면 됩니다.
- ✓ $N \leq 5\,000$이므로, S를 입력받을 때 정수형 자료형이 아닌 문자열 자료형으로 입력받아야 합니다.
- ✓ $\mathcal{O}\left(N^2\right)$으로도 널널히 풀리는 제한이므로, 편한 대로 구현해주면 됩니다.

Subtask 4

- Subtask 3와 비슷하게 앞에서부터 최대한 세 자리씩 자르면서 읽다가, 현재 수의 시작점이 0이라면 시작점을 앞으로 당겨줍시다.
- 주어진 수 S가 전역일 페이퍼에 적힌 수가 될 수 있음이 보장되므로, 해당 방식에 무한루프가 걸릴 일은 없습니다.
- 또한, 시작점이 0이 아니라면 Subtask 3와 동일하게 앞에서부터 최대한 3자리씩 끊는 것이 이득이므로, 그리디하게 문제를 해결할 수 있습니다.
- Greedy 풀이 외에도, $dp[i]$를 i번째 숫자까지 읽었을 때 필요한 최소 병사 수로 정의한 뒤 DP로 문제를 해결할 수도 있고, 가지치기를 적절히 하며 backtracking을 해 줘도 문제를 시간 안에 해결할 수 있습니다.

B. 사격

`sorting`, `binary_search`, `parametric_search`
출제진 의도 – **Easy**

- 제출 241번, 정답 33명 (정답률 13.693%)
- 처음 푼 사람: **kes0716**, 11분
- 출제자: kybs0627

Subtask 1

- ✓ $1 \leq N, M \leq 100, 1 \leq s_i \leq 1000$ 이므로 모든 시작 점수 s_i 에 대해서 시뮬레이션을 돌리면 해결할 수 있습니다.
- ✓ 총 시간복잡도는 $\mathcal{O}\left(NM \max(s_i)\right)$ 입니다.

Subtask 2

- $1 \leq N \leq 10^5$ 이므로 subtask1과 동일하게 하되 성현이가 현재 맞출 수 있는 표적 중 가장 점수가 높은 표적을 찾을 때 binary_search를 사용해 찾아주면 됩니다.
- 총 시간복잡도는 $\mathcal{O}\left(N\log(N) + \max(s_i)\log(N)M\right)$ 입니다.

Subtask 3

- 초기 사격 실력이 낮을 때에는 진급할 수 없다가 초기 사격 실력이 일정 이상 되는 순간 진급하게 됩니다.
- 따라서 모든 초기 사격 실력에 대해서 확인하는 대신 진급 가능한 초기 사격 실력을 이분 탐색으로 찾아가면 됩니다.
- 총 시간복잡도는 $\mathcal{O}\left((N\log(N)+M\log(N)\log(\max(s_i))\right)$ 입니다.

C. 훈련

dp
출제진 의도 – **Medium**

- 제출 86, 정답 12명 (정답률 16.279%)
- 처음 푼 사람: **menborong**, 21분
- 출제자: kybs0627

C. 훈련

Subtask 1

- 모든 훈련시간의 총합이 최대 훈련 시간보다 작으므로 단순히 모든 훈련의 훈련시간 총합을 출력해주면 됩니다.

C. 훈련

Subtask 2

- ✓ $N = 1$ 이므로 단순 배낭 문제입니다. 굉장히 잘 알려진 문제이므로 풀이는 생략하겠습니다.

C. 훈련

Subtask 3

- 각 훈련상황마다 적어도 하나의 훈련을 훈련 계획에 포함시켜야 합니다.
- 우선 1차원 dp 테이블을 정의합니다. $dp[i] = k$ 일때의 의미는 $1 \sim k$ 번째 훈련 상황에서 적어도 하나의 훈련을 훈련 계획에 넣어 i만큼의 시간을 소요할 수 있다는 뜻입니다.
- 이후 각 $1 \sim N$ 번 훈련 상황에 속한 훈련들을 차례로 순회하며 i 번째 훈련 상황의 j 번째 훈련을 순회할 때 $dp[l] = i - 1 \; or \; dp[l] = i$ 인 l 들에 대해 $dp[l + t_{ij}] = i$ 로 채워주면 됩니다.
- 마지막으로 $dp[t] = k$ 인 최대 t 를 구하면 답입니다.
- 총 시간복잡도는 $\mathcal{O}\left(M \sum_{i=1}^{N} d_i\right)$ 입니다.

D. 축구 대회

`ad_hoc`, `greedy`, `sorting`
출제진 의도 – **Hard**

- 제출 55번, 정답 8명 (정답률 14.545%)
- 처음 푼 사람: **cinador**, 44분
- 출제자: 99asdfg

D. 축구 대회

Subtask 1

- $a_i = b_i = c_i = d_i$ 이므로, 어떤 식으로 포지션을 배정해도 팀의 만족도는 동일합니다.
- 따라서, 단순히 11 개의 a_i 값을 더한 뒤 출력하면 됩니다.

Subtask 2

- Subtask 1 에서 N 이 11 보다 커질 수 있습니다.
- 따라서, a_i 를 내림차순으로 정렬한 뒤 가장 큰 11 개의 원소를 더하여 출력하면 됩니다.

Subtask 3

- ✓ $N = 11$ 이므로, 각각의 병사들을 각 포지션에 선발하는 모든 경우의 수는 $4^{11} = 4\,194\,304$ 가지입니다.
- ✓ 따라서, 완전 탐색으로 시간 내에 문제를 해결할 수 있습니다.

Subtask 4

- $a_i = b_i = c_i$ 이므로, 골키퍼를 제외한 세 가지 포지션에는 병사들을 어떻게 선발해도 상관이 없습니다.
- 따라서, 골키퍼를 맡을 선수를 한 명 고른 뒤, 해당 선수를 제외한 상위 선호도 10 명의 선호도를 합한 뒤 출력하면 됩니다.
- 골키퍼를 맡을 선수를 고르는 경우의 수는 N 가지이므로, $\mathcal{O}(N \log N)$ 으로 문제를 해결할 수 있습니다.

D. 축구 대회
Subtask 5

- 임의로 N 명 중 11 명을 선발한 경우, 각 포지션에 선발된 병사중 적어도 한 명 이상은 각 포지션별 선호도의 11 등 이내에 드는 병사가 있어야 합니다.
- 만약 그러한 병사가 한 명도 포함되지 않았다면, 해당 포지션에 선발된 병사를 11 등 이내에 드는 병사로 교체해 줌으로써 더 나은 최적해를 만들 수 있기 때문입니다.
- 또한, 각 포지션에 적어도 한 명씩을 이미 선발했다면, 선발한 병사들을 제외하고 7 명의 병사를 $\max(a_i, b_i, c_i)$ 가 큰 순서대로 선발하는 것이 최적임은 자명합니다.
- 따라서, 각 포지션별로 선호도의 상위 11 명을 뽑아 11^4 의 모든 경우의 수를 순회하며, 선발되지 않은 나머지 $N-4$ 명의 병사들 중 $\max(a_i, b_i, c_i)$ 의 최댓값 7 개를 더해 주면 됩니다.
- 총 시간복잡도는 $\mathcal{O}\left(N \log N + 11^5\right)$ 입니다.
- 별해로, bitfield를 활용한 DP로도 문제를 해결할 수 있습니다.

E. 시간 외 근무 멈춰!!!

`greedy`, `binary_search`, `parametric_search`, `implementation`, `simulation`
출제진 의도 – **Hard**

- 제출 16번, 정답 2명 (정답률 12.500%)
- 처음 푼 사람: **kes0716**, 77분
- 출제자: 99asdfg

E. 시간 외 근무 멈춰!!!

Subtask 1

- ✓ $N = 1, K = 0$ 이므로, 단순 시뮬레이션 혹은 수학 수식으로 문제를 해결할 수 있습니다.

Subtask 2

- ✓ $K = 0$ 이므로 데드라인을 전혀 조작하지 않습니다.
- ✓ 따라서, 문제에서 주어진 대로 시뮬레이션만 해주면 됩니다.

Subtask 3

- 초기 데드라인의 오름차순으로 작업을 정렬한 뒤, 순서대로 작업 $1, \cdots, N$ 이라고 합시다.
- 만약 작업 i 의 데드라인을 미룬 뒤 근무를 배치하면 다른 작업들은 다음과 같이 변합니다.
  - $j > i$ 인 작업 j 에 대해서, 바뀐 d_i 가 d_j 보다 크다면 최소한 $d_i <= d_j$ 가 되도록 데드라인을 미뤄야 합니다.
  - $j < i$ 인 작업 j 에 대해서는 아무 것도 변하지 않습니다.
- 따라서, 데드라인을 미룰 때는 반드시 초기 데드라인이 늦은 작업부터 미루는 것이 최선임을 알 수 있습니다.

Subtask 3

- 데드라인의 오름차순 순서대로, 작업에 최대한 근무를 배정하다가 시간 내에 모든 근무를 끝낼 수 없다면 데드라인을 최소한으로 늘려주는 방식으로 작업의 가능성 여부를 판별합시다.
- 만약 K 번 데드라인을 미뤄도 불가능한 작업이 생긴다면, 즉시 -1 을 출력하고 프로그램을 종료하면 됩니다.
- 가능성 여부 판별 완료 후, 데드라인이 늦은 작업이 시간 외 근무를 해야 한다면 데드라인을 미뤄줍시다.
- N, K, d_i, t_i 모두 $5\,000$ 이하이므로, 다양한 방식으로 구현할 수 있습니다.

Subtask 3

- 추가로, 데드라인이 늦은 작업의 데드라인을 먼저 미뤄야 한다는 말은, 결국 데드라인이 이른 작업을 우선적으로 시간 외 근무로 옮겨줘야 한다는 것으로 환원할 수 있습니다.
- 데드라인이 늦은 작업의 데드라인을 미루는 행위 자체가 결국 해당 작업의 시간 외 근무를 평시 근무로 바꾸기 위함이기 때문입니다.
- 따라서, 초기 데드라인이 늦은 작업의 데드라인을 미루는 구현 대신, 시간 외 근무 횟수를 매개변수로 하는 매개 변수 탐색을 활용하면 데드라인이 이른 작업에 우선적으로 시간 외 근무를 할당하는 풀이도 가능합니다.

E. 시간 외 근무 멈춰!!!
Subtask 4 + Subtask 5

- 전체적인 풀이의 결은 동일하지만, 최적화를 더욱 해주어야 합니다.
- 가장 단순한 방법 중 하나는, 시간 외 근무를 매개변수로 한 매개 변수 탐색에 각 작업을 얼마나 미뤄야 하는지도 매개 변수 탐색으로 찾는 것입니다.
- 위의 풀이로 문제를 해결하면, $\mathcal{O}\left(N \log 10^{18} \log N\right)$ 으로 문제를 해결할 수 있습니다.
- 시간복잡도를 더욱 최적화할 수도 있습니다.
- 작업 i의 데드라인을 미룰 때 $j > i$인 작업 j의 데드라인을 한 번 미룬다면 그 이후에는 $k < i$인 작업 k를 미룰 때도 작업 j를 미뤄야 합니다.
- 따라서, 이전 작업들의 데드라인이 몇 번 미뤄져야 각 작업의 데드라인을 미뤄야 하는지를 따로 저장해둔 뒤, 미뤄야만 하는 작업들은 각 요일에 몇 개나 있는지만 저장해 두며 시뮬레이션하면 $\mathcal{O}\left(N \log N\right)$으로도 문제를 해결할 수 있습니다.

F. 물자 조달

`offline_query`, `floyd_warshall`, `shortest_path`, `dijkstra`
출제진 의도 – **Challenging**

- 제출 37번, 정답 2명 (정답률 5.405%)
- 처음 푼 사람: **junhyung9985**, 152분
- 출제자: kybs0627

F. 물자 조달

Subtask 1

- ✓ $N = 2$ 이므로 부대의 검문시간이 의미가 없습니다.
- ✓ 따라서 1번 쿼리는 전부 무시하고 2번 쿼리는 단순히 두 부대 사이의 도로의 소요 시간을 출력해주면 됩니다.

Subtask 2

- B국이 최대 10번만 습격하므로 B국이 습격할 때 마다 floyd_warshall을 돌려주면 됩니다.
- 총 시간복잡도 $O(10N^3)$

F. 물자 조달

Subtask 3

- B국이 a번 부대만 공격한다고 가정하면, 이때 그래프에서 유일하게 바뀌는 점은 부대 a의 검문시간입니다.
- floyd_warshall은 k번째 루프에 모든 노드 쌍 (i, j)에 대해 $1 \sim k-1$번 노드를 지나는 최소 경로를 가지고 $1 \sim k$번 노드를 지나는 최소 경로를 구합니다.
- 따라서 floyd_warshall을 돌릴 때 a번 부대를 가장 마지막에 경유하게 알고리즘을 돌리면 $N-1$번째 루프에서 모든 노드 쌍 (i, j)에 대해 a번 부대를 지나지 않는 최소 경로를 구할 수 있습니다.
- 이후 a번 부대의 검문 시간이 바뀌더라도 그 값은 변하지 않으므로 a번 부대의 검문 시간이 바뀔 때 마다 $N-1$번째 루프의 값으로 a번 부대를 경유할 수 있는 경로를 계산해주면 $O(1)$만에 계산이 가능합니다.

F. 물자 조달

Subtask 3

- ✓ 총 시간복잡도는 $O(N^3 + Q)$ 입니다.

Subtask 4

- 최단 거리를 구하는 문제이므로 relaxation을 사용하려면 그래프의 weight가 감소하는 쪽으로 봐야 합니다.
- 기본적으로 offline_query이므로 쿼리를 뒤에서부터 처리하면 부대의 검문시간이 단조감소함을 알 수 있습니다.
- 초기에 floyd_warshall 알고리즘을 이용해 최소거리를 구해줍니다.
- B국이 k번째 부대를 공격하고 있는 상황에서는 아래와 같이 $O(1)$만에 2번 쿼리를 처리할 수 있습니다.
- $dist[i][j] = \min(dist[i][j], dist[i][k] + dist[k][j] + inspection[k])$

Subtask 4

- B국이 k 번째 부대에서 l 번째 부대로 공격 목표를 바꾼다면 이전에 공격받은 부대의 줄어든 검문시간을 다시 최소경로에 반영해줘야 하며. 해당 과정을 아래와 같이 $O(N^2)$ 만에 처리할 수 있습니다.
- 모든 (i, j) 쌍에 대해서 $dist[i][j] = \min(dist[i][j], dist[i][k] + dist[k][j] + inspection[k])$
- B국은 공격 횟수를 최대 N 번 변경하므로 총 시간복잡도는 $O(N^3 + Q)$ 입니다.