25-1 수뭉컵
Official Solution

25-1 수뭉컵 풀이

문제		난이도	출제자
A	낙엽	Easy	김남주
B	문자열 줄이기	Easy	김수성
C	6의 배수	Normal	김수성
D	양치기	Normal	김남주
E	수뭉이와 쿼리	Hard	김수성
F	7016 사수 대작전	Hard	김수성
G	잠꾸러기 수뭉이	Hard	김수성

A. 낙엽

Simulation

난이도 - Easy　　출제자 - 김남주

제출 100회, 정답 12명 (정답률 12%)

mandooking, 15분 28초

A. 낙엽

풀이

청소를 하면 모든 낙엽이 0장이 되기 때문에 하루마다 낙엽이 가장 많이 쌓이는 장소의 낙엽 개수만 확인해도 무방합니다.

하루에 낙엽이 가장 많이 쌓이는 장소의 낙엽 개수가 오늘 밤에 B를 초과하게 된다면 0으로 만들어주고 정답에 1을 추가하면 정답을 구할 수 있습니다.

A. 낙엽

풀이

알고리즘은 다음과 같습니다.
하루에 가장 많이 쌓이는 낙엽의 개수를 mx 라고 합시다.

A번 동안 반복문을 돕니다.
오늘 밤에 낙엽이 쌓였을 때 B를 초과하게 된다면,
즉 오늘 낮에 낙엽의 개수가 B - mx를 초과하면
낙엽의 개수를 0으로 초기화하고 정답에 1을 추가합니다

그리고 낙엽의 개수에 B를 추가합니다.

B. 문자열 줄이기

Strings

난이도 – Easy　　출제자 - 김수성

제출 46회, 정답 9명 (정답률 19.57%)

mandooking, 26분 17초

B. 문자열 줄이기

풀이

다음과 같은 두 가지 케이스를 고려해봅시다.

모든 인덱스 i에 대해서 S[i] != S[i + 1]인 경우

어떤 인덱스 i에 대해서 S[i] == S[i + 1]인 경우

B. 문자열 줄이기

풀이

모든 인덱스 i에 대해서 S[i] != S[i + 1]인 경우

이 경우에는 한번의 연산도 수행할 수 없습니다.
따라서 정답은 주어진 문자열의 길이가 됩니다.

B. 문자열 줄이기

풀이

어떤 인덱스 i에 대해서 S[i] == S[i + 1]인 경우

이 경우에는 무조건 문자열을 1의 길이로 줄일 수 있습니다.

연산을 수행한 뒤에 S[i]를 S[i]와 인접한 문자로 바꿔주면 연산을 또 수행할 수 있습니다.

이는 문자열의 길이가 1이 될 때 까지 수행 할 수 있습니다.

C. 6의 배수

Math

난이도 – Normal 출제자 - 김수성

제출 82회, 정답 7명 (정답률 8.54%)

tempnickname, 11분 21초

C. 6의 배수

서브 태스크 1

N의 범위가 64비트 정수형 이내이므로,
N % 6 == 0 인지 확인 해주면 됩니다.

C. 6의 배수

서브 태스크 2

N의 범위가 64비트 정수형을 초과하므로 모듈러 연산을 사용할 수 없습니다.

우선 주어진 N을 문자열로 받습니다.

C. 6의 배수

서브 태스크 2

6의 배수인 조건은 3의 배수와 2의 배수의 조건을
동시에 만족하는 것 입니다.

2의 배수는 짝수, 즉 문자열의 끝자리가 2의 배수인지
판단하면 됩니다.

3의 배수는 모든 자리 수의 합이 3의 배수인지
판단하는 것으로 알아 낼 수 있습니다.

D. 양치기

Sorting

난이도 – Normal　　출제자 – 김남주

제출 42회, 정답 3명 (정답률 7.14%)

김주헌, 57분 45초

D. 양치기

풀이

더 빠른 양이 느린 양 뒤에 있으면 더 빠른 양은 결국
느린 양을 따라 잡아서 같은 그룹을 형성하게 됩니다.

따라서 자신보다 느린 양이 앞에 없는 양의 개수를 세면
전체 그룹의 개수를 구할 수 있습니다.

이 문제는 위치 순으로 입력이 들어온다는 보장이 없기 때문에
우선 위치 순서대로 정렬을 해줘야 합니다.

D. 양치기

풀이1

자신보다 느린 양이 앞에 있는지 판단하는 첫 번째 방법은
자신보다 앞에 있는 가장 느린 양의 속도를 저장해 두는 것 입니다.

앞의 양부터 순회를 하면서 앞의 가장 느린 양의 속도보다
자신이 느리면 정답에 1을 추가합니다.

이렇게 하면 각 양을 한번씩만 순회하므로
O(N)의 시간 복잡도가 들게 됩니다.

D. 양치기

풀이2

자신보다 느린 양이 앞에 있는지 판단하는 두 번째 방법은
뒤의 양부터 스택에 삽입하고, 순회를 하면서 스택의 값이
자신보다 빠를 때 스택의 값을 제거하면 됩니다.

위의 연산 후에 스택의 길이가 정답이 됩니다.

각 양들은 스택에 최대 한번 씩 삽입/삭제가 되므로
시간 복잡도는 O(N)이 됩니다.

E. 수뭉이와 쿼리

Prefix Sum

난이도 – Hard

출제자 - 김수성

제출 11회, 정답 1명 (정답률 9.09%)

mandooking, 2시간 1분 25초

E. 수뭉이와 쿼리

서브 태스크 1

쿼리마다 모든 L <= i < j <= R에 대해서 S, M의 쌍을 브루트포스로 구해주면 됩니다.

시간 복잡도는 O(N^2 * M)이 됩니다.
N <= 300, M <= 10이므로 제한 시간 안에 돌아갑니다.

E. 수뭉이와 쿼리

서브 태스크 2

L <= i <= R, S[i] == 'M' 인 어떤 인덱스 i를 사용해서
"SM" 쌍을 만드는 것은 L <= j < i, S[j] == 'S' 인
j의 개수만큼 만들 수 있습니다.

각 쿼리마다 L부터 R까지 순회하면서
현재 값이 'M'일 때 이전의 'S'의 개수를 더해주면 됩니다.

'S'에 대해서 누적 합을 사용하면 O(NM)에 처리가 가능합니다.

E. 수뭉이와 쿼리

서브 태스크 3

서브 태스크 2의 풀이에서는 각 ‘M’에 대해서 ‘S’에 대한 누적합으로 ‘SM’ 쌍의 개수를 더해서 풀었습니다.

‘S’에 대한 누적 합 이외에도 ‘M’에 대한 누적 합과 ‘SM’ 쌍에 대한 누적 합을 생각 해봅시다.

편의상 각각 S_Pre, M_Pre, SM_Pre 라고 합시다.

E. 수뭉이와 쿼리

서브 태스크 3

(L, R) 범위의 “SM”쌍의 값을 구하는 것은
SM_Pre[R] – SM_Pre[L]로 구할 수 있습니다.

하지만 L 이전의 ‘S’도 “SM”쌍을 구하는데 포함이 되기 때문에
이 값을 빼줘야 합니다.

이는 L 이전의 ‘S’의 개수와 (L, R) 범위의 ‘M’을 곱한 값과 같습니다.

E. 수뭉이와 쿼리

서브 태스크 3

L 이전의 'S'의 개수와 (L, R) 범위의 'M'을 곱한 값은 누적 합으로
다음과 같이 구할 수 있습니다.
S_Pre[L - 1] * (M_Pre[R] - M_Pre[L - 1])

(L, R) 범위의 "SM"쌍의 값에 위의 값을 빼면
정답을 구할 수 있습니다.

식은 다음과 같습니다.
SM_Pre[R] - SM_Pre[L - 1]
- S_Pre[L - 1] * (M_Pre[R] - M_Pre[L - 1])

E. 수뭉이와 쿼리

서브 태스크 3

각 누적 합을 구하는데 O(N), 각 쿼리 당 O(1)의
시간 복잡도가 소요됩니다.

총 O(N + M)의 시간 복잡도로 문제를 풀 수 있습니다.

F. 7016 사수 대작전

DP

난이도 – Hard

출제자 - 김수성

제출 26회, 정답 0명 (정답률 0%)

25-1 수뭉컵 풀이

F. 7016 사수 대작전

서브 태스크 1

수뭉이는 각 정류장마다 앉거나 서 있는 2가지 선택 중
하나를 할 수 있습니다.

N번의 정류장에 대해서 선택의 모든 경우의 수를 해보고
그 중 최솟값을 구해서 출력해주면 됩니다.

경우의 수는 2^N개 있고, 각 경우의 수에서 최솟값을 구하는데
O(N)의 시간 복잡도가 들기 때문에,
총 시간 복잡도는 O(N * 2^N)이 됩니다.

F. 7016 사수 대작전

서브 태스크 2

DP[i][j]를 i번째 정류장에서 j의 힘이 드는 좌석에 앉은 상태,
DP[i][0]을 i번째 정류장에서 서 있는 상태에서
가장 적게 드는 힘이라고 정의합시다. (j != 0)

DP[i][0]은 서 있는 상태이므로 전의 정류장에서
서 있는 상태와 앉아 있는 상태 모두에서 전이 할 수 있습니다.

전의 상태에서 서 있을 때 드는 비용을 더 해주면 됩니다.
DP[i][0] = MIN(DP[i - 1][j] + B[i])
(0 <= j <= 100)

25-1 수뭉컵 풀이

F. 7016 사수 대작전

서브 태스크 2

DP[i][A[i]]는 이번 정류장부터 앉은 것으로 볼 수 있습니다.
서 있는 상태와 앉아 있는 상태 모두에서 전이 할 수 있습니다.

전의 상태에서 앉아 있을 때 드는 비용을 더 해주면 됩니다.
DP[i][A[i]] = MIN(DP[i – 1][j] + A[i]) (1 <= j <= 100)

F. 7016 사수 대작전

서브 태스크 2

DP[i][j] (j != A[i])는 이미 앉아 있는 상태입니다.

전의 상태에서 현재 드는 비용을 더 해주면 됩니다.
DP[i][j] = MIN(DP[i - 1][j] + j) (1 <= j <= 100 && j != A[i])

F. 7016 사수 대작전

서브 태스크 2

점화식은 다음과 같습니다.

DP[i][0] = MIN(DP[i – 1][j] + B[i])
(0 <= j <= 100)

DP[i][A[i]] = MIN(DP[i – 1][j] + A[i])
(0 <= j <= 100)

DP[i][j] = MIN(DP[i – 1][j] + j)(j != A[i] && j != 0)

25-1 수뭉컵 풀이

F. 7016 사수 대작전

서브 태스크 2

각 DP 상태가 N, MAX(A[i])이므로 모든 DP 상태를 방문하는데
O(N * MAX(A[i]), 상태 전이에 O(MAX(A[i])가 들기 때문에
총 시간 복잡도는 O(N * MAX(A[i]) ^ 2)가 됩니다.

F. 7016 사수 대작전

서브 태스크 3

서브 태스크 2의 점화식을 보면 DP[i][j]의 값을 구할 때
DP[i – 1] 중 최솟값을 사용합니다.

DP[i][A[i]] = MIN(DP[i][A[i]], DP[i – 1][j] + A[i])
(0 < j <= 100)

DP[i][0] = MIN(DP[i][0], DP[i – 1][j] + B[i])
(0 < j <= 100)

F. 7016 사수 대작전

서브 태스크 3

서브 태스크 2의 풀이에서는 DP[i - 1]의 최솟값을 찾기 위해서
반복문을 돌기 때문에 O(MAX(A[i]))의 시간 복잡도가 걸립니다.

DP[i - 1]의 최솟값을 DP[i]의 상태에 전이하기 전에
전처리를 해놓으면 O(1)에 최솟값을 구할 수 있습니다.

DP 상태 전이에 시간 복잡도가 O(1)로 줄었으므로,
총 시간 복잡도는 O(N * MAX(A[i])가 됩니다.

G. 잠꾸러기 수뭉이

Parametric Search, Dijkstra

난이도 – Hard 출제자 - 김수성

제출 3회, 정답 0명 (정답률 0%)

G. 잠꾸러기 수뭉이

서브 태스크 1, 2

정점 N까지 최단 거리로 가야하고,
각 간선마다 일정 시간 이후에는 간선을 사용 할 수 없습니다.

서브태스크 1, 2에서는 L의 제한이 20이므로
모든 출발 시간 L에 대해서 다익스트라를 실행합니다.

각 L에 대해서 처음으로 최단 거리가 늘어나는 L을 구하면 됩니다.
L = 0일 때도 도착할 수 없으면 -1을 출력합니다.

G. 잠꾸러기 수뭉이

서브 태스크 1, 2

다익스트라는 시간 복잡도가 O(M log N)입니다.
다익스트라를 K번 돌리기 때문에
총 시간 복잡도는 O(KM log N)입니다.

G. 잠꾸러기 수뭉이

서브 태스크 3

어떤 출발 시간 L에 대해서 최단거리가 늘어나는 시간이면 f(L) = 0, 최단거리가 출발 시간 0과 동일하면 f(L) = 1인 함수 f를 정의 해봅시다.

그러면 어느 순간부터는 최단거리가 늘어나고, 최단거리가 늘어나기 전 시간은 모두 최단거리가 출발 시간 0과 동일합니다.

즉 함수 f는 1 1 1 1 0 0 0 0 꼴의 내림차순 형태가 됩니다.

G. 잠꾸러기 수뭉이

서브 태스크 3

문제에서는 출발 시간 0과 동일한 최단 시간이 걸리는
최대 출발 시간 L을 찾아야 합니다.

즉 f(L) = 1인 L의 최댓값을 구해주면 됩니다.
함수 f는 항상 단조 감소하기 때문에 매개 변수 탐색을
사용 할 수 있습니다.

L에 대해서 매개 변수 탐색을 진행하고, 각 f(L)값은
다익스트라를 사용해서 구할 수 있습니다.

G. 잠꾸러기 수뭉이

서브 태스크 3

다익스트라에 O(M log N), 매개 변수 탐색에 O(log K)의 시간 복잡도가 들기 때문에 총 시간 복잡도는 O(M log N * log K)가 됩니다.