2025 중앙대학교 프로그래밍 경진대회 풀이
- Open Contest -

by

중앙대학교 알고리즘 학회 ChAOS

지원



A	A1	Easy	하늘에서 떨어지는 N개의 별	… 4
	A2	Easy	L-트로미노 타일링	… 6
B	B1	Normal	우주 여행	… 9
	B2	Normal	푸앙이와 콩나무	… 11
	B3	Normal	로마의 휴일	… 13
	B4	Normal	세기의 대결	… 15
C	C1	Hard	p^n!과 쿼리	… 18
	C2	Hard	택배 상하차는 힘들어	… 20
	C3	Hard	하늘에서 떨어지는 ∞개의 별	… 23
D	D1	Challenge	CPC 문제 정렬 순서	… 25

A1. 하늘에서 떨어지는 N개의 별

#simulation

출제진 의도 - Easy

✓ 출제자: wapas (세터 : saywoo)

✓ 제출: 440번, 정답: 161명 (정답률 38.409%), 처음 푼 사람: woohyun_jng, 1분

A1. 하늘에서 떨어지는 N개의 별

- ✓ 하루에 별이 가장 많이 쌓이는 점을 찾습니다.
- ✓ 그 점에 쌓인 별의 개수가 K개를 초과하기 전에 청소하게 되면, 다른 점들의 쌓인 별의 개수가 항상 K개를 초과하지 않습니다.
  - ✓ 그리고 별이 떨어질 때 K개를 초과하게 되는 날 낮에 청소하게 되면, 최소 횟수로 청소를 할 수 있습니다.
- ✓ N, D의 범위가 작으므로 시뮬레이션을 돌려 최소 청소 횟수를 찾을 수 있습니다.

A2. L-트로미노 타일링

#combinatorics

출제진 의도 - Easy

✓ 출제자: wapas

✓ 제출: 289번, 정답: 141명 (정답률 51.211%), 처음 푼 사람: dong5995, 3분

A2. L-트로미노 타일링

✓ 3×1 공간에서 L-트로미노를 채울 수 있는 방법은 없습니다.

✓ 3×2 공간에서 L-트로미노를 채울 수 있는 방법은 다음과 같이 2가지 있습니다.

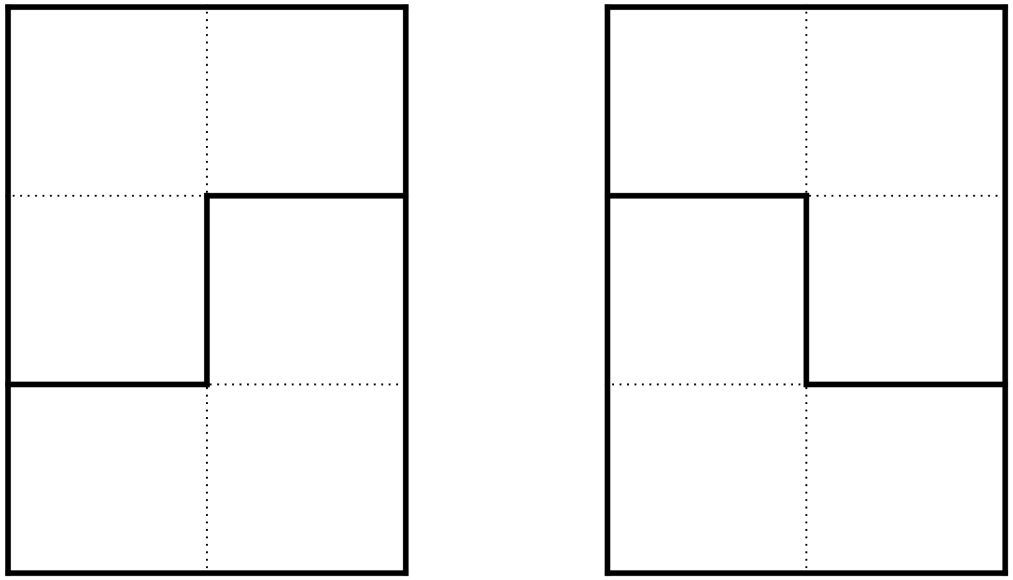


사진 1: 3×2 공간에서 L-트로미노를 채울 수 있는 방법

A2. L-트로미노 타일링

✓ $3 \times N$을 가로 2칸씩 쪼개어 생각해봅니다.

  ✓ N이 홀수일 때는 채울 수 있는 방법이 없습니다. 따라서 0가지 존재합니다.

  ✓ N이 짝수일 때는 가로 2칸당 2가지 채울 수 있으므로 $2^{\frac{N}{2}}$가지 존재합니다.

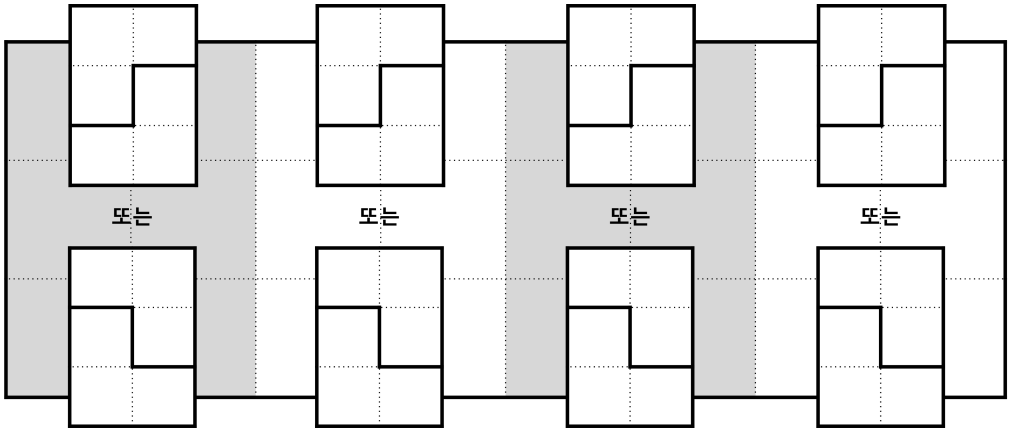


사진 2: N이 짝수일 때, $3 \times N$ 공간에서 L-트로미노를 채울 수 있는 방법

B1. 우주 여행

#ad_hoc, #constructive

출제진 의도 - Normal

✓ 출제자: wapas (세터 : saywoo)

✓ 제출: 107번, 정답: 36명 (정답률 34.579%), 처음 푼 사람: chmpro, 8분

B1. 우주 여행

- i번 이동했을 때의 칸을 (a_i, b_i)라고 할 때 시작 지점에서 도착 지점까지의 이동 소요 시간의 총합 T는 다음과 같습니다.
  - $T = \left(t_{a_0 b_0} - t_{a_1 b_1}\right) + \left(t_{a_1 b_1} - t_{a_2 b_2}\right) + \ldots + \left(t_{a_{L-1} b_{L-1}} - t_{a_L b_L}\right)$
    $= t_{a_0 b_0} - t_{a_L b_L}$
- 즉, 어떠한 경로로 이동해도 이동 소요 시간의 총합이 항상 같고, 최소가 됩니다.
- 정확히 L번 이동하여 도착 지점까지 가는 경로가 존재하는지 판단해야 합니다.
  - $L < N + M - 2$인 경우, L번의 이동으로 도착 지점까지 이동할 수 없습니다.
  - $L \geq N + M - 2$인 경우, $L - (N + M - 2)$가 짝수면 경로가 항상 존재하며, 홀수면 경로가 존재하지 않습니다.

B2. 푸앙이와 콩나무

#bfs | #dp

출제진 의도 - Normal

✓ 출제자: skjd1234

✓ 제출: 169번, 정답: 64명 (정답률 38.462%), 처음 푼 사람: dong5995, 7분

B2. 푸앙이와 콩나무

- ✓ 초기 콩나무 길이인 0을 시작 노드로 설정합니다.
- ✓ N의 최댓값이 1000000 이므로, 콩나무 길이를 인덱스로 가지면서 최소 날짜와 물의 양을 갱신해줄 수 있는 공간을 선언해줍니다.
- ✓ 초기 노드 0에서 시작해 규칙에 따라 다음날 가능한 콩나무 길이의 경우의 수를 갱신해줍니다.
- ✓ 경우의 수를 계속 찾다가 콩나무 길이가 N이 되면 날짜와 물의 양을 출력해줍니다.
- ✓ 새로운 콩나무 길이가 N보다 크다면 탐색하지 않고 건너뜁니다.
- ✓ 이미 방문한 콩나무 길이의 경우, 더 적은 날짜에 방문했다면 다시 방문하지 않고 같은 날짜에 방문했다면 물의 양이 더 적을 경우 추가로 방문합니다.

B3. 로마의 휴일

#two_pointer | #prefix_sum, #binary_search

출제진 의도 - Normal

✓ 출제자: sksms1375

✓ 제출: 77번, 정답: 31명 (정답률 40.260%), 처음 푼 사람: fermion5, 6분

B3. 로마의 휴일

- ✓ i를 휴가를 다녀오기 전 마지막 출근한 날 인덱스, j를 휴가를 다녀오고 복귀한 날 인덱스라고 합시다.
- ✓ $S(l,\ r) = a_l + a_{l+1} + ... + a_{r-1} + a_r$이라고 하면, $X \cdot S(1,\ i) + S(j,\ N) \geq K$를 만족하는 가장 큰 $j - i - 1$을 찾으면 됩니다.
- ✓ i를 1부터 1씩 증가시키다가 $X \cdot S(1,\ i) \geq K$를 만족한다면 멈춥니다. 휴가를 다녀온 후 복귀를 하지 않아도 되니 현재 다녀올 수 있는 휴가 일은 $N - i$일입니다.
- ✓ 하지만 휴가를 다녀오고 일을 하게 되는 경우가 오히려 휴가를 더 많이 다녀올 수도 있습니다. j를 N부터 1씩 감소시키면서 $X \cdot S(1,\ i) + S(j,\ N) \geq K$가 될 때까지 i를 1씩 감소시킵니다. 식을 만족하는 경우 휴가를 다녀온 일 수를 매번 갱신합니다.
- ✓ 따라서 $\mathcal{O}(n)$으로 풀 수 있습니다. 누적 합과 이분 탐색을 이용한 매개 변수 탐색을 사용해도 제한 시간 내에 풀 수 있습니다. 이때는 $\mathcal{O}(n \log n)$입니다.
- ✓ 만약 $X \cdot S(1,\ i) \geq K$를 만족하는 최소 i가 N이라면 어떤 방법으로도 휴가를 가지 못한다는 뜻입니다. 처음에 잘 처리하도록 합시다.

B4. 세기의 대결

#dp

출제진 의도 - Normal

✓ 출제자: xingxing2001

✓ 제출: 51번, 정답: 26명 (정답률 50.980%), 처음 푼 사람: dbgusdn012, 11분

B4. 세기의 대결

- ✓ 게임에서 죽게 되면 점수가 0점이 되므로, 죽지 않고 점수를 획득해야 합니다.
- ✓ 만약, i번째 총알을 사격한 후, 죽지 않고 점수를 획득하려면 $i+1$번째부터 N번째 총알 중에서 i번째 총알보다 위력이 큰 총알을 사격해야 합니다.
- ✓ 죽지 않고 게임을 종료하면, 사격한 총알들의 위력이 오름차순을 이룹니다. 즉, 플레이어가 획득한 점수는 총알의 위력이 이루는 가장 긴 부분 증가 수열의 길이가 됩니다.
- ✓ 이를 통해 dp[i] = i번째 총알을 사격했을 때 얻을 수 있는 최고 점수라고 할 때,
- ✓ dp[i] = max(1, max{dp[j] | $j < i$ and arr[j] < arr[i]} + 1)의 점화식을 얻을 수 있습니다.

B4. 세기의 대결

- √ 플레이어는 시작점을 자유롭게 선택할 수 있으므로, 1번째 총알부터 N번째 총알로 시작하는 모든 경우에 대해 원형 배열에서 연속된 N개의 총알을 고려하여 가장 긴 부분 증가 수열의 길이를 찾으면 됩니다.
- √ N의 최댓값이 500이므로 $\mathcal{O}(N^3)$의 시간복잡도로 계산할 수 있습니다.
- √ 위와 같은 방법을 통해 영재와 해강이의 최대 점수를 구하고, 승부를 결정할 수 있습니다.

C1. $p^n!$과 쿼리

`#number_theory, #modular_multiplicative_inverse`

출제진 의도 - Hard

✓ 출제자: sksms1375

✓ 제출: 45번, 정답: 39명 (정답률 88.889%), 처음 푼 사람: dkim110807, 10분

C1. p^n!과 쿼리

✓ 문제에서 요구하는 것은 $p^n!$에서 소수 p가 몇 번 곱해졌는지를 뜻합니다. 이 값은 $\sum_{k=1}^{\infty} \left\lfloor \frac{p^n}{p^k} \right\rfloor$입니다.

✓ 식을 전개하면 $\left\lfloor \frac{p^n}{p} \right\rfloor + \left\lfloor \frac{p^n}{p^2} \right\rfloor + \ldots = p^{n-1} + p^{n-2} + \ldots + p + 1$입니다. 이는 초항이 1이고 등비가 p인 등비수열의 합입니다.

✓ 등비수열의 합 공식으로 나타내면 $\frac{p^n-1}{p-1}$ 이고 쿼리 당 $\mathcal{O}(\log n)$으로 구할 수 있습니다.

C2. 택배 상하차는 힘들어

`#tree, #graph_traversal | #dp_tree`

출제진 의도 - Hard

✓ 출제자: asdarwin03

✓ 제출: 38번, 정답: 21명 (정답률 55.263%), 처음 푼 사람: dkim110807, 35분

C2. 택배 상하차는 힘들어

- 택배의 상차 및 하차 횟수의 총합을 최소화하는 문제입니다.
- 트럭이 한 번 지나간 도로는 다시 사용이 불가능하므로, 1번 도시에서 각 도시를 목적지로 하는 $N-1$개 트럭에 도시별 택배를 싣고 배송하는 전략은 사용할 수 없습니다.
- 마찬가지로 해당 조건에 의해 트럭이 지나간 도로는 다시 되돌아갈 수 없으므로, 모든 도시에 배송을 완료하기 위해서는, 어떤 도시에서 다음 도시들로 배송할 때 반드시 도로로 연결된 도시 개수 만큼의 트럭을 활용해야 합니다.
- 이때 활용할 수 있는 트럭은 해당 도시에 있는 비어있는 트럭과, 앞서 도시에 도달하기 위해 활용했던 트럭 1대가 있습니다.
- 상하차 횟수를 최소화하기 위해서는, 앞서 사용했던 트럭을 최대한 활용하여 다음 도시로 택배를 배송하는 것이 반드시 최적일 것입니다.
- 그 도시에 도달했던 트럭에서 택배를 하차하여 다른 트럭에 상차를 해야하는데, 만약 그 트럭을 다시 활용한다면 일부 도시를 위한 택배는 그 트럭에서 하차할 필요 없이 바로 출발하는 발상을 떠올릴 수 있습니다.

C2. 택배 상하차는 힘들어

- √ 이제 각 트럭에서 어떤 전략으로 택배를 실어야할 지 생각해봅시다.
- √ 어떤 도시에서 앞서 사용했던 트럭을 활용하여 다음 도시로 배송할 때, 그 트럭에 있는 택배의 개수만큼 하차 및 상차 횟수에 이득을 봅니다.
- √ 따라서 각 도시에서 앞서 사용했던 트럭을 활용하되, 그 트럭에 '남을' 택배 개수를 최대화하는 전략이 반드시 최적입니다.
- √ 이는 어떤 도시에서 갈 수 있는 다음 도시들 중, 각 도시를 루트 노드로 한 서브트리의 택배 수의 총합이 최대가 되는 택배들이 바로 트럭에 '남을' 택배에 해당합니다.
- √ 즉 각 도시에 대해 해당 전략을 취하면 반드시 최적이며, 이 전략을 토대로 트럭에 택배를 넣는 순서가 반드시 존재함을 보장할 수 있습니다.
- √ 상하차 횟수를 구하는 것이 문제의 요구사항이므로, 해당 전략을 토대로 횟수만 적절히 계산해주면 쉽게 문제를 해결할 수 있습니다.
- √ 각 도시를 루트 노드로 한 서브트리의 택배 수의 총합을 계산하기 위해 DFS를 활용할 수 있습니다.
- √ 총 시간복잡도는 $\mathcal{O}(2N)$에 준합니다.

C3. 하늘에서 떨어지는 ∞개의 별

#greedy, #number_theory

출제진 의도 - Hard

✓ 출제자: wapas (세터 : sksms1375)

✓ 제출: 60번, 정답: 21명 (정답률 35.000%), 처음 푼 사람: fermion5, 29분

C3. 하늘에서 떨어지는 ∞개의 별

- ✓ A1번 문제와 마찬가지로 하루에 별이 가장 많이 쌓이는 점을 찾으면 됩니다.
- ✓ $s_{i+1} = (s_i + X) \bmod M$를 일반화 하여 나타내면 $s_i = (s_0 + X \cdot i) \bmod M$입니다. $\gcd(X, M) = d$라고 합시다.
- ✓ $s_i \bmod d = (s_0 \bmod d + (X \bmod d) \cdot (i \bmod d)) = s_0 \bmod d + 0 \cdot (i \bmod d) = s_0 \bmod d$ 이므로 s_i는 $0 \le s_i < M$를 만족하면서 $s_i \bmod d = s_0 \bmod d$인 값 중에 가장 큰 값을 찾으면 됩니다. $r = s_0 \bmod d$라고 합시다.
- ✓ s_i 중에 최댓값을 M_s이라고 하고, $M_s = M - 1 - \Delta$로 나타내 봅시다. M_s를 구하는 방법은 위의 조건을 만족하면서 Δ가 가장 작은 값을 구하면 됩니다.
- ✓ 조건을 모두 적용하면 $(M - 1 - \Delta) \equiv r \pmod d$가 되고 M, r, d를 모두 알고 있으므로 이 Δ 값을 구할 수 있게 되고 이 값을 이용해 M_s를 구합니다.
- ✓ 최댓값을 $\mathcal{O}(\log(\min(X, M)))$으로 구했으므로 K와 D를 이용해서 $\mathcal{O}(1)$로 최소 청소 횟수를 구할 수 있습니다.

D1. CPC 문제 정렬 순서

`#greedy, #priority_queue | #greedy, #disjoint_set, #map`

출제진 의도 - Challenge

✓ 출제자: wapas

✓ 제출: 53번, 정답: 8명 (정답률 16.981%), 처음 푼 사람: woohyun_jng, 61분

D1. CPC 문제 정렬 순서

- ✓ 다음과 같이 배열 A를 정의합니다.
  - ✓ N개의 문제의 난이도를 결정하여 배열 A에 넣는다.
- ✓ M개의 섹션을 만들 수 있다는 뜻은, 배열 A에 서로 다른 수가 M개 이상이 존재하는 것과 동치입니다.
- ✓ 따라서 배열 A의 원소를 최대한 서로 다르게 넣는 것이 문제의 목표가 됩니다.
- ✓ 만들 수 있는 배열 A 중 원소가 최대한 서로 다르게 구성된 배열 A를 배열 B라고 정의합니다.
- ✓ 배열 B를 찾았다면 다음과 같이 M개의 섹션으로 나눌 수 있습니다.
  - ✓ 원소의 종류가 M개 미만이면 -1을 출력하며, 그렇지 않으면 다음 과정을 진행합니다.
  - ✓ 배열 B를 원소 오름차순으로 순회하면서 각 섹션에 한 종류의 수를 넣습니다.
  - ✓ 마지막 섹션에는 순회하지 않은 배열 B의 모든 수를 넣습니다.

D1. CPC 문제 정렬 순서

- ✓ 배열 B를 찾는 방법 (1) - 우선 순위 큐 / $\mathcal{O}(N \log N)$
  - ✓ 난이도 d를 1부터 증가시켜보면서 현재 난이도 d를 선택할 수 있는 구간들을 찾을 것입니다.
  - ✓ 문제들을 (l_i, r_i) 기준으로 오름차순으로 정렬합니다.
  - ✓ 우선순위 큐에는 현재 난이도 d를 할당할 수 있는 구간들이 담기도록 유지합니다.
    - ✓ 가능한 구간 중 탐욕적으로 r_i이 작을 때를 가장 먼저 할당하면, 최대한 서로 다른 수를 만들 수 있습니다.
  - ✓ 정렬한 문제를 다음과 같이 순회합니다. (현재 보고 있는 문제를 i번이라 정의)
    - ✓ 우선순위 큐가 비었다면 d를 할당할 문제가 없다는 뜻이므로 $d = l_i$로 점프합니다.
    - ✓ d보다 l_i가 클 때까지 i를 증가시키며 r_i를 우선순위 큐에 넣습니다.
    - ✓ d보다 r_i가 작을 때까지 우선순위 큐의 원소를 뽑습니다.
    - ✓ 우선순위 큐가 비어있지 않다면 우선순위 큐 top의 구간에 난이도 d를 할당합니다.
  - ✓ 할당되지 않은 구간들에는 $[l_i, r_i]$ 구간 중 아무 정수로 할당합니다.

D1. CPC 문제 정렬 순서

- ✓ 배열 B를 찾는 방법 (2) - 분리 집합 & 맵 / $\mathcal{O}(N \log^2 N)$
  - ✓ 문제를 r_i가 큰 순서대로 보면서, l_i보다 크면서 할당 가능한 최소 난이도를 비교하여 각 구간에 난이도를 할당할 것 입니다.
    - ✓ (1)번 방법과 마찬가지로 탐욕적으로 최대한 서로 다른 수를 만들게 됩니다.
  - ✓ 문제들을 (r_i, l_i) 기준으로 오름차순으로 정렬합니다.
  - ✓ 분리 집합에는 이미 할당된 난이도들을 넣으며 find를 호출 할 때는 할당되지 않은 가장 최소의 난이도를 반환합니다.
  - ✓ 정렬한 문제를 다음과 같이 순회합니다. (현재 보고 있는 문제를 i번이라 정의)
    - ✓ find(l_i)를 호출하면 l_i보다 크면서 할당 가능한 최소 난이도 x를 얻을 수 있습니다.
    - ✓ x가 r_i보다 작거나 같다면 할당한 뒤 parent[x] = find($x+1$)로 저장합니다.
    - ✓ x가 r_i보다 크다면 $[l_i, r_i]$ 구간 중 아무 정수로 할당합니다.
- ✓ parent[x]에서 x의 범위가 10^9까지되니 parent를 map으로 관리합니다.

모두 수고하셨습니다!

중앙대학교 알고리즘 학회 ChAOS