PPC 2024

Official Solutions

PPC 2024 - 운영진



- ✓ 출제진
  - 권찬우 kwoncycle
  - 이유담 petamingks
  - 임병찬 hyperbolic
  - 오승혁 menborong
  - 장래오 leo020630
  - 정병현 jbh0224
  - 정윤성 slah007
- ✓ 검수진
  - jk410
  - pasa3232
  - pjshwa
  - tony9402
- ✓ 테스터
  - 김동욱 duk1003
  - 김범수 qjatn0120baek
  - 이건규 kkpos0206
  - 이원기 qwerty_po
  - 이태민 idearendil2018
  - 조정훈 justin8021

문제		의도한 난이도	출제자
A	글로벌 포닉스	**Easy**	장래오leo020630, 권찬우kwoncycle, 이유담petamingks
B	예비소집 결과 보고서	**Easy**	장래오leo020630
C	나무평평설	**Hard**	오승혁menborong
D	PPC 만들기	**Medium**	임병찬hyperbolic, 장래오leo020630
E	포닉스와 지구 정복	**Medium**	이유담petamingks
F	불사조	**Easy**	장래오leo020630
G	학교를 무너뜨리는 포닉스	**Hard**	권찬우kwoncycle
H	저체온증	**Medium**	정윤성slah007
I	78계단 내려가기 대회	**Medium**	장래오leo020630
J	포닉스의 문단속	**Easy**	임병찬hyperbolic, 장래오leo020630
K	시소 배열	**Medium**	장래오leo020630
OA	최고의 친구	**Hard**	장래오leo020630
OB	고장난 계산기	**Hard**	권찬우kwoncycle, 오승혁menborong
OC	나무판자	**Challenging**	정윤성slah007, 임병찬hyperbolic

A. 글로벌 포닉스

`string`, `case_work`
출제진 의도 – **Easy**

- 제출 276번, 정답 158명 (정답률 59.420%)
- First Solve: **seawon0808**, 0분
- 출제자: 장래오[leo020630], 권찬우[kwoncycle], 이유담[petamingks]



- 각 문자열의 첫 글자가 순서와 관계 없이 l, k, p인지 판별합시다.
- l, k, p를 나열하는 6가지 경우를 모두 확인해보면 됩니다.
- 이외에도 문자열을 정렬해 첫 글자가 k, l, p인지 확인하는 방법, 세 문자열의 첫 글자의 곱을 구해 비교하는 방법 등 여러 풀이가 존재합니다.
- P.S. 여러분의 응원 덕에 포스텍은 ICPC World Finals에 진출하게 되었습니다.
- 감사합니다!

B. 예비소집 결과 보고서

`implementation`
출제진 의도 – **Easy**

- 제출 206번, 정답 142명 (정답률 55.340%)
- First Solve: **aeren**, 3분
- 출제자: 장래오 `leo020630`



- 우선 한 문제도 풀지 않은 팀을 가려냅시다. 이는 간단한 조건문으로 처리할 수 있습니다.
- 한 문제 이상을 푼 팀에 대해서 문제를 순서대로 해결했는지 판별합시다.
- 여러 방법으로 해결할 수 있지만 가장 간단한 방법은 다음과 같습니다.
- 해결하지 않은 문제에 대한 T 값을 -1이 아닌 120보다 큰 수로 바꾸어 줍시다.
- 이러한 과정을 거친 후 $T_{i1} \leq T_{i2} \leq T_{i3}$ 이라면 i 번 팀이 문제를 순서대로 해결했음을 의미합니다.
- 각 팀에 대해 예비소집을 성실하게 참여했는지 판별한 후, 성실히 참여한 팀의 수를 구해 출력하면 됩니다.

C. 나무평평설

`tree`, `greedy`
출제진 의도 – **Hard**

- ✓ 제출 18번, 정답 8명 (정답률 44.444%)
- ✓ First Solve: **aeren**, 10분
- ✓ 출제자: 오승혁`menborong`



- 트리의 모든 가중치를 0으로 만드는 어떤 단순 경로 집합이 주어졌다고 가정해 봅시다.
  - 하나의 간선들로 이루어진 경로들로 한가지 집합을 만들 수 있으므로 최소 하나의 집합이 존재합니다.
- 어떤 정점 v 및 v에 연결된 서로 다른 간선 e_1, e_2를 생각해 봅시다.
- e_1을 지나고 v를 끝점으로 하는 단순 경로와 e_2를 지나고 v를 끝점으로 하는 단순 경로가 각각 존재한다면, 이 둘을 연결해 새로운 경로를 만들 수 있습니다.
- 이때 새로운 경로는 두 단순 경로가 v 외의 다른 정점에서 만나지 않기 때문에 새로운 단순 경로가 됩니다.
- 이러한 과정을 v의 합치기 연산이라고 정의합시다.
- 합치기 연산을 1회 적용할 때 마다 단순경로의 개수가 1개 줄어들게 됩니다.



- 서로 다른 정점 v_1, v_2 에 대해서 v_1 에서 합치기 연산을 했다고 가정해봅시다.
- v_1 에서의 어떠한 두 간선을 기준으로 합치기 연산을 적용하더라도 v_2 와 연결된 각 간선에서 v_2 를 끝점으로 하는 단순경로의 개수는 변함이 없습니다.
- 또한 트리이므로 여전히 v_2 에서의 합치기 연산에 의해 단순경로가 아닌 경로가 만들어질 수 없습니다.
- 따라서 어떤 정점에서의 합치기 연산을 하더라도 다른 정점에서의 가능한 합치기 연산 횟수에 영향을 주지 않습니다.
- 그러므로 각 정점에서 최대한 많은 합치기 연산을 하면 됩니다.

- 초기 상태로 모든 단순 경로가 하나의 간선으로 이루어졌다고 생각해봅시다.
- 각 정점에서 최대 몇번의 합치기 연산을 가해 얼마나 많은 경로의 수를 줄일 수 있는지 생각해 봅시다.
- 한 정점 v에 연결된 간선의 수를 k개라고 하고, 이때의 가중치를 내림차순으로 나열했을 때, $w_1 \geq w_2 \geq ... \geq w_k$라고 하고, 이때의 가중치에 대응되는 간선을 각각 $e_1, e_2, ..., e_k$이라고 합시다.
- 합치기 연산에 의해 만들어진 간선은 더 이상 v를 기준으로 합치기 연산에 사용될 수 없으므로, 두 간선의 가중치를 각각 1씩 감소시키는 연산이라고 생각할 수 있습니다.

- 먼저 $w_1 \geq \sum_{i=2}^{k} w_i$ 라고 가정해 봅시다.
- 이 경우에는 e_1 이외의 간선을 지나는 모든 경로들을 e_1 를 지나는 경로들을 이용해서 모두 합치기 연산을 할 수 있습니다.
- 동일한 간선을 지나는 단순 경로끼리는 합칠 수 없으므로, 위와 같이 합치는 방법이 최선입니다.
- 따라서, 이때의 합치기 횟수는 $\sum_{i=2}^{k} w_i$ 입니다.

- 이번에는 반대 케이스인 $w_1 < \sum_{i=2}^{k} w_i$ 라고 가정해 봅시다.
- e_1 를 지나고 v 를 끝점으로 하는 단순 경로와 e_2 를 지나고 v 를 끝점으로 하는 단순 경로를 합쳐봅시다.
- 이러면 w_1, w_2 가 각각 1 씩 감소하게 됩니다.
- 이제 다시 내림차순이 되도록 가중치들을 정렬하여 다시 새롭게 $w_1 \geq w_2 \geq ... \geq w_k$ 를 정의합니다.
- 이 과정을 반복하면 $0 \leq w_1 \leq 1, w_2 = w_3 = ... = w_k = 0$ 인 순간이 반드시 나오게 됩니다.

- 이를 증명해 봅시다.
- 일단 $w_2 > 0$ 이었다면, 위의 연산을 더 적용할 수 있으므로 $w_2 = w_3 = ... = w_k = 0$ 인 순간은 반드시 올 수 있음은 자명합니다.
- 이제 $w_1 > 1, w_2 = w_3 = ... = w_k = 0$ 이 되었다고 가정해 봅시다.
- 모든 $i \neq 1$ 에 대해서 $w_1 - w_i > 1$ 이므로, 바로 직전의 연산에 의해 e_1 에 바뀔 수 없습니다.
- 이 연산에 의해 w_1 은 1 감소했고, 다른 $w_{i \neq 1}$ 는 1 감소했거나 변함없습니다.
- 이는 $w_1 - w_{i \neq 1} > 1$ 를 계속 유지 했음을 의미하고, e_1 는 변한 적이 없게 됩니다.

- $w_1 < \sum_{i=2}^{k} w_i$ 이고 연산을 반복해도 e_1 가 바뀌지 않았지만, w_1 번 반복한다면 e_1 의 비용은 0 이 되고, $\sum_{i=2}^{k} w_i > 0$ 이 됩니다.
- 하지만 이렇게 되면 e_1 는 바뀌어야 하므로 모순이 됩니다.
- 따라서 $0 \leq w_1 \leq 1, w_2 = w_3 = ... = w_k = 0$ 인 순간이 반드시 나오게 됩니다.
- 항상 가중치 합은 2 씩 감소하므로 가중치의 합의 홀짝 여부는 유지되므로, 남은 가중치 합이 1 이하인것은 항상 최적의 방법임을 의미합니다.

- 결국 각 정점 v에 대해서
- $w_1 \geq \sum_{i=2}^{k} w_i$ 인 경우 $w_1 - \sum_{i=2}^{k} w_i$ 개의 끝점을 갖고
- $w_1 < \sum_{i=2}^{k} w_i$ 인 경우 $\left(\sum_{i=1}^{k} w_i\right) \%2$ 개의 끝점을 갖는 것이 최선입니다.
- 이렇게 각 정점에서 끝점의 개수의 합을 구하고, 2로 나눈 것이 최소 단순경로의 개수가 됩니다.

D. PPC 만들기

`greedy`, `combinatorics`
출제진 의도 – **Medium**

- 제출 167번, 정답 46명 (정답률 34.731%)
- First Solve: **pretty_pi**, 10분
- 출제자: 임병찬[hyperbolic], 장래오[leo020630]

D. PPC 만들기

- PPC 부분문자열을 가능한 많이 만들기 위해서는 P가 최대한 문자열의 앞쪽에, C가 문자열의 뒤쪽에 위치하는 것이 최선입니다.
- 따라서 가장 앞에 위치한 C와 가장 뒤에 위치한 P를 서로 바꾸는 연산을 반복합시다.
- 이러한 P와 C의 위치를 매번 찾으면 최악의 경우 $\mathcal{O}(NK)$의 연산량이 필요해 시간 내에 문제를 해결할 수 없습니다.
- 이는 초기 문자열에서의 모든 P와 C의 위치를 각각 구해두거나, 두 포인터 테크닉을 사용하는 것으로 해결할 수 있습니다.
- 이때 만약 K 번을 모두 사용하기 전에 문자열이 PP..PC..C 형태로 정렬되었다면 중단해야 합니다.

- K 번의 연산을 모두 사용했거나, 문자열이 PP..PC..C 형태가 되었다면 완성된 문자열에 대해 PPC 부분문자열의 개수를 구해야 합니다.
- $P_i = S[1 \cdots i]$ 에 등장하는 P 의 개수라고 정의합시다. 이는 간단한 누적 합 테크닉으로 구해줄 수 있습니다.
- $S_i =$ C 인 모든 i 에 대해 $\binom{P_{i-1}}{2} = \frac{P_{i-1}(P_{i-1}-1)}{2}$ 를 구해 모두 더한 것이 PPC 부분문자열의 개수가 됩니다.
- 시간 복잡도는 $\mathcal{O}(N+K)$ 입니다.

E. 포닉스와 지구 정복

`ad_hoc`, `math`, `graph_traversal`
출제진 의도 – **Medium**

- 제출 11번, 정답 8명 (정답률 72.727%)
- First Solve: **aeren**, 24분
- 출제자: 이유담`petamingks`

- A의 상태에서 B의 상태로 만드는 것이 목표이므로, 기대하는 높이의 변화량인 $C_{ij} = A_{ij} - B_{ij}$를 고려해봅시다.
- C의 모든 칸을 0으로 변화시키는 것과 A에서 B로 상태를 변화시키는 것은 동치입니다.
- 만약 화산이 폭발한 칸이 없다면 문제를 어떻게 풀 수 있을까요?



- 체스 판처럼 모든 격자에 대해 검은 색과 흰 색을 번갈아가며 칠해봅시다.
- 연산을 실행할 때 검은 색 칸 하나와 흰 색 칸 하나에 1 혹은 -1만큼 동시에 변화시키므로, S = (검은 색 칸의 C_{ij} 의 합) - (흰 색 칸의 C_{ij} 의 합)이 불변함을 관찰할 수 있습니다.
- 따라서, S가 0이라면 C의 모든 열에 대해 첫 번째 칸을 제외하고는 0을 만들고, 그 이후 첫 번째 칸들을 0으로 만들어주는 것을 통하여 C를 모두 0으로 만들 수 있으므로 A를 B로 바꾸는 것이 가능합니다.
- S가 0이 아니라면 아무리 연산을 해도 불변량을 변화시킬 수 없으므로 A를 B로 바꾸는 것이 불가능합니다.



- 화산이 폭발했을 때로 확장해봅시다. 만약 화산이 폭발한 칸 (i, j)에 대하여 $C_{ij} \neq 0$이라면 자명하게 A를 B로 바꾸는 것이 불가능합니다.
- 화산이 폭발하지 않은 모든 칸을 정점으로 잡고, 상하좌우로 인접한 칸끼리 간선을 이어 무향 그래프를 구성해봅시다. 비슷하게, 각각의 연결요소 하나마다 연결요소에 들어있는 모든 (i, j)에 대하여 S = (검은 색 칸의 C_{ij}의 합) - (흰 색 칸의 C_{ij}의 합)를 고려해봅시다.
- 모든 연결요소에 대해 S가 0이라면 A를 B로 바꾸는 것이 가능, 그렇지 않다면, 불가능합니다.
- 증명은 아래와 같습니다.



- ✓ 연산을 적용하는 두 칸은 자명하게 같은 연결 요소에 있으므로 모든 연결 요소에 대해 S는 불변값입니다. 따라서 S가 0이 아닌 연결요소가 존재한다면 불가능합니다.
- ✓ 모든 S가 0인 경우를 생각해봅시다.
- ✓ 임의의 연결요소에 대하여 DFS 트리를 잡고, 정점이 하나가 될 때까지 DFS 트리에서 leaf node에 해당하는 칸의 값을 0으로 만들고 제거해주는 작업을 반복합시다. (트리의 성질에 의해 leaf node는 반드시 존재합니다.)
- ✓ 불변값 S에 의하여 남은 하나의 정점에 해당하는 칸의 값은 0이 되고, 모든 연결요소에 대해 같은 방식으로 모든 칸의 값들을 0으로 만들 수 있으므로, 가능합니다.
- ✓ 위와 같은 방식으로 작업의 방법도 구할 수 있습니다.
- ✓ 시간복잡도는 $O(NM)$입니다.

F. 불사조

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

- 제출 91번, 정답 69명 (정답률 82.418%)
- First Solve: **sjhi00**, 13분
- 출제자: 장래오 `leo020630`



풀이 1.

- ✓ 조상 포닉스가 명령을 내릴 때 새로 만들어지는 포닉스의 수는 2마리, 4마리, 8마리.. 와 같이 증가합니다.
- ✓ 따라서 조상 포닉스가 M 번 명령을 내릴 때 만들어진 포닉스의 총 수는 $2^{M+1}-1$ 마리임을 알 수 있습니다.
- ✓ $M \leq 20$ 이므로 시간 안에 모든 포닉스의 마력을 충분히 구할 수 있습니다.
- ✓ 재귀함수 등의 적절한 방법을 사용해 구현하면 됩니다.

풀이 1.

- 구체적으로는 다음과 같습니다.
- 조상 포닉스의 마력 x와 남은 명령 횟수 m을 인자로 가지는 함수 $f(x, m)$을 정의합시다.
- 이 함수의 반환값은 마력이 x인 조상 포닉스가 m번 명령을 내린 후 존재하는 모든 포닉스의 마력의 합입니다.
- $f(X, M)$이 문제의 답이 되며, 이는 다음과 같은 관계식으로 구할 수 있습니다.
- $$f(x,m) = \begin{cases} x & \text{if } m = 0 \\ f(\lfloor \frac{x}{2} \rfloor, m-1) + f(\lceil \frac{x}{2} \rceil, m-1) & \text{if } m > 0 \end{cases}$$

풀이 2.

- ✓ $\lfloor \frac{x}{2} \rfloor + \lceil \frac{x}{2} \rceil = x$ 입니다.
- ✓ 이를 통해 조상 포닉스가 매번 명령을 내릴 때 새로 만들어지는 포닉스의 마력의 합이 항상 M 임을 알 수 있습니다.
- ✓ 따라서 $X(M+1)$ 이 답이 됩니다.

G. 학교를 무너뜨리는 포닉스

`greedy`, `two-pointer`
출제진 의도 – **Hard**

- 제출 28번, 정답 4명 (정답률 14.286%)
- First Solve: **aeren**, 47분
- 출제자: 권찬우 `kwoncycle`



- 어떤 벽돌 A를 제거했을 때 무너지는 벽돌의 집합을 S_A라고 합시다. 그리고 벽돌 B가 S_A의 벽돌들 중 하나라고 하고, B를 제거했을 때 무너지는 벽돌의 집합을 S_B라고 합시다. 이때, $S_B \subset S_A$임을 관찰할 수 있습니다.
- 따라서 어떤 벽돌 B가 다른 벽돌 A에 의해 무너진 적이 있는 경우, 무조건 A가 B보다 많은 벽돌을 무너뜨리므로 B는 보지 않아도 됩니다.
- 따라서 밑에서부터 벽돌을 하나씩 제거하되, 이미 무너진 적이 있던 벽돌은 건너뛰는 방식으로 탐색을 진행하면 중복되는 벽돌 없이 모든 벽돌을 다 탐색할 수 있습니다.
- 각각의 벽돌을 무너뜨렸을 때 얼마나 많은 벽돌이 무너질지는 이분탐색이나 투포인터 등을 사용하면 쉽게 구할 수 있습니다. 투포인터로 구현할 경우 O(N)의 시간복잡도로 모든 벽돌을 탐색할 수 있습니다.

H. 저체온증

`graph_traversal`, `ad_hoc`
출제진 의도 – **Medium**

- 제출 20번, 정답 11명 (정답률 55.000%)
- First Solve: **pretty_pi**, 31분
- 출제자: 정윤성[slah007]



- ✓ 우선 첫날 낮의 상황을 생각해 봅시다.
- ✓ 저체온증에서 정상 체온으로 회복되기 위해서는 주변에 두 명 이상의 정상 체온인 사람이 있어야 합니다. 정상 체온인 사람과 인접한 사람들부터 순차적으로 조사해 봅시다.
- ✓ 초기에 큐에 정상 체온인 사람을 모두 집어넣어 줍시다. 그리고 큐에서 각각의 정상 체온인 사람과 인접한 네 사람을 조사합니다. 이 사람이 저체온증이면서 인접한 두 명이 정상 체온이라면 회복시킨 뒤 큐에 집어넣고, 큐가 완전히 빌 때까지 이러한 과정을 반복합니다.
- ✓ 이러한 과정을 반복하면 결국 정상 체온인 사람과 인접한 모든 사람을 검사하게 되므로 규칙에 맞게 사람들을 회복시킬 수 있습니다. 한 사람은 많아야 한 번 큐에 들어가고, 각각 인접한 네 명을 조사하므로 $O(NM)$ 번의 연산만 사용합니다.

- 또한, 첫날 낮이 끝난 뒤 연결된 정상 체온인 사람들의 집단은 모두 직사각형 모양입니다. 두 정상 체온인 사람이 팔방향으로 인접한 사람을 건너 연결되어 있다면 이 두 사람을 마주보는 두 꼭짓점으로 하는 직사각형 영역에 있는 사람이 모두 정상 체온이 되기 때문입니다.
- 이때 직사각형의 두 변의 길이 중 하나라도 K 이하라면 그 변의 맨 끝 줄에 해당하는 모두가 저체온증에 걸리는 경우 변의 길이가 1 씩 줄어들어 결국 모두가 저체온증에 걸리게 됩니다. 이외에는 어떠한 K 명이 저체온증에 걸리더라도 모두 정상 체온으로 회복됩니다.
- 즉, 직사각형 중 두 변의 길이가 모두 K 초과인 것들의 길이의 합을 구하면 됩니다. 단순하게 정상 체온인 사람들을 구성하는 직사각형의 좌상단 꼭짓점에 해당하는 사람만 보면서 직사각형의 크기를 구하면 $O(NM)$ 번의 연산으로 구할 수 있습니다.



I. 78 계단 내려가기 대회

DP, binary_search
출제진 의도 – **Medium**

- 제출 56번, 정답 20명 (정답률 35.714%)
- First Solve: **riroan**, 40분
- 출제자: 장래오 leo020630

- DP를 사용합니다. DP_i를 i번 칸에 도달했을 때의 최대 점수로 정의합시다.
- 이에 기반해 다음과 같은 점화식을 세울 수 있습니다.
- $$DP_i = \max_{1 \le j < i} \begin{cases} DP_j + A_i, & \text{if } H_j \ge H_i + B_i \\ DP_j, & \text{otherwise} \end{cases}$$
- 이를 그대로 계산하는 것은 $\mathcal{O}\left(N^2\right)$으로 시간 내에 문제를 해결하기 어렵습니다. 더 개선해 봅시다.

- $j < i, H_j \geq H_i + B_i$인 최대의 j를 $f(i)$라 합시다.
- 문제 조건에 의해, $j_1 < f(i)$인 j_1에 대해 j_1번 칸에서 i번 칸으로 바로 내려오는 것보다 j_1번 칸에서 $f(i)$번 칸으로 이동한 후 i번 칸으로 내려오는 것이 이득입니다.
- 마찬가지로, $f(i) < j_2 < i - 1$인 j_2에 대해 $i - 1$번 칸으로 이동한 후 내려오는 것이 이득입니다.



- 이를 바탕으로 DP 식을 다음과 같이 최적화할 수 있습니다.
- $DP_i = \max(DP_{f(i)} + A_i, DP_{i-1})$
- $f(i)$는 각 i에 대해 이분 탐색으로 찾아줄 수 있습니다. $f(i)$가 존재하지 않거나 $B_i = 0$인 경우에 유의합시다.
- 따라서 $\mathcal{O}(N \log N)$의 시간 복잡도에 문제를 해결할 수 있습니다.

J. 포닉스의 문단속

`greedy`
출제진 의도 – **Easy**

- ✓ 제출 102번, 정답 42명 (정답률 42.157%)
- ✓ First Solve: **riroan**, 30분
- ✓ 출제자: 임병찬`hyperbolic`, 장래오`leo020630`



- 다이얼을 돌리면 A에 도달하기 전까지 문자열이 사전순으로 증가하고, A에 도달한 이후에는 해당 다이얼을 더 돌리지 않아도 됩니다.
- 따라서 앞에 위치한 다이얼부터 확인해 A로 만들 수 있는지 판별한 후, 가능하다면 만드는 것을 반복합시다.
- 마지막 다이얼에서는 남은 횟수를 모두 사용해야 함에 유의합시다.
- 시간 복잡도는 $\mathcal{O}(N)$ 입니다. 구현에 따라 $\mathcal{O}(N+K)$ 에 해결하는 것도 가능합니다.



K. 시소 배열

`data_structure`, `simulation`
출제진 의도 – **Medium**

- 제출 60번, 정답 30명 (정답률 50.000%)
- First Solve: **riroan**, 25분
- 출제자: 장래오 `leo020630`



- 문제에서 주어지는 질의는 배열, std::vector, 스택, 큐 등의 적절한 자료구조를 사용해 그대로 구현할 수 있습니다.
- 그렇다면, 이를 구현한 프로그램으로 주어진 제한 시간 안에 문제를 해결할 수 있을까요?
- 놀랍게도, 주어진 질의를 그대로 처리하도록 구현한 프로그램은 제한 시간 안에 동작합니다.
- 이것이 가능한 이유를 분석해 봅시다.

- 주어지는 2번 질의의 개수를 K라 합시다.
- i번째 2번 질의와 $i+1$번째 2번 질의 사이에 주어지는 1번 질의의 개수를 T_i, 첫 번째 2번 질의 전에 주어지는 1번 질의의 개수를 T_0, K번째 2번 질의 후에 주어지는 1번 질의의 개수를 T_k라 정의합시다.



- 첫 번째 2번 질의에서 배열의 길이는 T_0 입니다.
- 두 번째 2번 질의에서 배열의 길이는 $T_1 + \frac{T_0}{2}$ 입니다.
- 이를 확장하면, i 번째 2번 질의에서의 배열의 길이를 $\sum_{j=0}^{i-1} \frac{T_j}{2^{i-1-j}}$ 로 나타낼 수 있습니다.



- 이를 모두 더해 봅시다.
- $$\sum_{i=1}^{K}\sum_{j=0}^{i-1}\frac{T_j}{2^{i-1-j}}=\sum_{j=0}^{K}\sum_{i=0}^{K-j}\frac{T_j}{2^i}\leq\sum_{j=0}^{K}2T_j\leq 2Q$$
- 따라서 모든 2번 질의에 대해 배열의 길이의 합이 $\mathcal{O}(Q)$ 이므로 주어진 연산을 그대로 구현하더라도 $\mathcal{O}(Q)$ 의 시간 복잡도로 문제를 해결할 수 있습니다.

OA. 최고의 친구

`geometry`, `data_structure`
출제진 의도 – **Hard**

- ✓ 제출 33번, 정답 7명 (정답률 21.212%)
- ✓ First Solve: **dkim110807**, 41분
- ✓ 출제자: 장래오`leo020630`



- ✓ 두 점 $(-t, 0), (t, 0)$과 (x, y)가 이루는 삼각형이 예각삼각형임은 아래의 두 가지 조건을 동시에 만족함과 동치입니다.
  - 1. $-t < x < t$
  - 2. $t^2 < x^2 + y^2$
- ✓ $y \neq 0$이므로 $x^2 + y^2 \leq t^2$인 점들은 모두 1번 조건을 만족합니다.
- ✓ 따라서 $-t < x < t$를 만족하는 기지국의 개수에서 $x^2 + y^2 \leq t^2$을 만족하는 기지국의 개수를 빼면 조건을 만족하는 기지국의 개수를 구할 수 있습니다.



- 이는 적절한 자료구조를 사용해 해결해줄 수 있습니다.
- 출제자의 풀이는 모든 기지국에 대한 $x, x^2 + y^2$ 값을 모두 구한 후 좌표압축해 각각 세그먼트 트리로 관리하는 것입니다.
- 이외에 PBDS를 사용하는 풀이 등 여러 접근이 가능합니다.
- 시간 복잡도는 $\mathcal{O}(Q \log Q)$ 입니다.

OB. 고장난 계산기

`dp`, `greedy`
출제진 의도 – **Hard**

- 제출 7번, 정답 1명 (정답률 14.286%)
- First Solve: **aeren**, 167분
- 출제자: 권찬우`kwoncycle`, 오승혁`menborong`

- 계산기의 모든 연산을 순차적으로 적용하는 문제이므로, 역으로 N 을 0 으로 만드는 문제로 생각해도 무방합니다.
- 이 경우 역순이 되며, $+, -$ 끼리 바뀌고, $\times, \div$ 끼리 바꾸면 됩니다.
- 편의를 위해 해설에서는 각 연산을 등장 순서대로 적용하도록 작성하였습니다.
- $N \leq 10^{18}$ 이므로, bfs로 모든 가능성을 탐색하는것은 시간안에 불가능합니다.
- 따라서 특정 상황에서 특정 연산으로 한정지을 수 있을지 생각해 봅시다.

- ✓ N을 0으로 만드는 데에 필요한 최소 횟수를 $f(N)$이라고 합시다.
- ✓ $N = 0$의 경우 $f(N) = 0$번입니다.
- ✓ $N = 1$의 경우 $\times 4 - 4$이므로 $f(N) = 2$번입니다.
- ✓ $N = 2$의 경우 $\times 4 - 4 - 4$이므로 $f(N) = 3$번입니다.
- ✓ $N = 3$의 경우 $-4 \times 4 + 4$이므로 $f(N) = 3$번입니다.

Lemma1. N**이** 0**이 아닌** 16**의 배수라면,** $f(N) = f(N/4) + 1$이다.

- ✓ 나누기 4가 아닌 다른 연산을 하는 방법이 더 좋은 해를 갖는다고 가정해봅시다.
- ✓ $+4$를 하는게 더 이득이라고 가정할 경우,
  - $+4$를 4번 이상 연속으로 하는 경우에는 $\div 4 + 4 \times 4$로 횟수를 줄일 수 있으므로 3번 이하만 고려하면 됩니다.
  - $+4$를 3번 이하를 한 경우, $16x + 4y$ (x, y는 정수, $x > 0, 1 \leq y \leq 3$) 의 형태가 됩니다.
  - $16x + 4y$에서 $\div 4$를 한 경우에는 $4x + y$가 되므로, $\times 4$를 하기 전에는 y를 제거할 수 없습나다.
  - 그런데 이는 $\times 4$이후에 $+4y$ 해도 동일한 결과를 가져오므로, $\div 4$를 먼저해도 무방합니다.
  - 따라서 $16x + 4y$에서 $\times 4$를 한 경우에만 고려하면 됩니다.

Lemma1. N**이** 0**이 아닌** 16**의 배수라면,** $f(N) = f(N/4) + 1$이다.

- ✓ -4를 하는게 더 이득이라고 가정할 경우,
  - 마찬가지로 $16x - 4y$에서 $\times 4$를 한 경우에만 고려하면 됩니다. (x, y는 정수, $x > 0, 1 \leq y \leq 3$)
  - 만약 $16x - 4y$에서 $\times 4$하는 경우 또한 더 좋은 결과를 줄 수 없다면, 모든 경우에서 -4로는 이득을 볼 수 없으므로 모순이 됩니다.

이제 $+4y \times 4, -4y \times 4$ 두가지 경우와, $\times 4$에 대해서 함께 보일 것입니다.



Lemma1. N이 0이 아닌 16의 배수라면, $f(N) = f(N/4) + 1$이다.

- ✓ $\pm 4y \times 4$를 하는게 더 이득이라고 가정하면, $(x > 0, 0 \leq y \leq 3)$
  - 연산 후에 $64x \pm 16y$의 꼴이 되며, 이는 다시한번 16의 배수가 됩니다.
  - 또한 해당 범위에서 $64x \pm 16y > 16x \pm 4y$이므로 항상 증가합니다.
  - $(\pm 4y \times 4)$를 1회 후에 $\div 4$를 적용할 경우, $\times 4 \div 4$는 상쇄되므로, $(\pm 4y \times 4)$를 2회 이상 적용해야 합니다.
  - $(\pm 4y \times 4)$를 k회 후에 $\div 4$를 적용할 경우, $\times 4 \div 4$는 상쇄되므로, $(\pm 4y \times 4)$를 $k+1$회 이상 적용해야 합니다.
  - 하지만, k가 증가함에 따라 N은 계속 커지므로, 유한한 어떤 k번의 $(\pm 4y \times 4)$ 후에는 반드시 다른 연산을 적용해야만 하지만, 다른 가능한 연산은 존재하지 않습니다.
  - 따라서 이는 모순이 됩니다.
- ✓ 따라서 N이 16의 배수라면 $f(N) = f(N/4) + 1$입니다.

Lemma2. N **이** 4**의 배수가 아닐 경우,** $f(N) = \min(f(N\%4) + \lfloor N/4 \rfloor, f(4N) + 1)$ **이다.**

- ✓ $f(N\%4) + \lfloor N/4 \rfloor$ 의 경우 N 이 4 미만이 될때까지 -4 연산을 적용한 것입니다.
- ✓ $f(4N) + 1$ 의 경우 $\times 4$ 연산을 1 회 적용한 것입니다.
- ✓ 이외의 방법으로는 더 적은 연산 방법을 제공하지 않음을 의미합니다.
- ✓ 이를 증명하기 위해, 이외의 방법으로 더 적은 연산으로 0 을 만들 수 있다고 가정해 봅시다.

Lemma2. N **이** 4**의 배수가 아닐 경우,** $f(N) = \min(f(N\%4) + \lfloor N/4 \rfloor, f(4N) + 1)$ **이다.**

- ✓ N 은 4의 배수가 아니므로 $\div 4$ 를 할 수 없습니다.
- ✓ 따라서 처음에 ± 4 연산만을 사용하는 상황을 고려하면 됩니다.
- ✓ ± 4 만으로는 0 이 될 수 없으므로, 어느 시점에서 반드시 $\times 4$ 연산을 적용해야만 합니다.
- ✓ $\times 4$ 이후 바로 $\times 4$ 를 할 경우 16 의 배수가 되고, 이는 Lemma1 에 의해 다시 $\div 4$ 를 하게 되며, 이는 상쇄되어 횟수 낭비가 발생합니다.
- ✓ $\times 4$ 이후 바로 $\div 4$ 를 할 경우에도 상쇄가 발생하므로 역시 낭비가 발생합니다.
- ✓ 따라서 $\times 4$ 이후에는 ± 4 연산을 해야만 합니다.

Lemma2. N **이** 4**의 배수가 아닐 경우,** $f(N) = \min(f(N\%4) + \lfloor N/4 \rfloor, f(4N) + 1)$ **이다.**

- ✓ 일단 ± 4 이후에도 16의 배수가 아니라고 가정합시다.
- ✓ ± 4 이후에도 여전히 Lemma1에 의해 $\times 4$ 연산을 고려할 필요가 없습니다.
- ✓ ± 4 이후에 적용하는 $\div 4$의 경우에는, $\times 4$ 이전에 존재했던 ± 4 연산들을 $\div 4$ 연산 뒤로 보내도 동일한 결과를 가져옵니다.
- ✓ 이는 결국 $\times 4$ 연산을 처음으로 하는 상황과 동일합니다.
- ✓ 따라서 다시한번 ± 4를 해야합니다.
- ✓ 만약 ± 4 이후에 16의 배수가 되었다면 Lemma1에 의해 $\div 4$를 하게 되며, 이는 마찬가지로 $\times 4$ 연산을 먼저 하는 상황으로 재배치 가능합니다.

Lemma2. N **이** 4**의 배수가 아닐 경우,** $f(N) = \min(f(N\%4) + \lfloor N/4 \rfloor, f(4N) + 1)$ **이다.**

- ✓ 예외적으로 N 이 $\times 4$ 이후 ± 4 를 반복하다가 16 의 배수가 된 순간이 0 이 된 경우입니다.
- ✓ 해당 상황이 가능하려면 $\times 4$ 직전에 N 의 절대값이 3 이하여야 합니다.
- ✓ 여기서 $f(N)$ 와 $f(-N)$ 는 연산에서 덧셈 뺄셈 연산만 바꾸면 동일하므로, 음수로 넘어가는 상황은 고려하지 않아도 됩니다.
- ✓ 이 예외를 제외한다면 이외의 방법으로 더 적은 연산으로 0 을 만들 수 있다고 가정이 모순이 됩니다.
- ✓ 따라서 $f(N) = \min(f(N\%4) + \lfloor N/4 \rfloor, f(4N) + 1)$ 가 됩니다.



Lemma3. N **이** 4**의 배수이고** 16**의 배수가 아니라면**

$f(N) = \min\left(f(N - N\%16) + \frac{N\%16}{4}, f(N + 16 - N\%16) + \frac{16 - N\%16}{4}\right)$ **이다.**

- ✓ 위의 식은 N이 16의 배수가 될 때까지 $+4$ 또는 -4 연산을 반복한다는 뜻입니다.
- ✓ $\times 4$를 하게 될 경우 16의 배수가 되고 Lemma 1에 의해 다시 $\div 4$를 하게 되는데, 이를 상쇄할 수 있으므로 $\times 4$로는 더 적은 연산으로 0을 만들 수 없습니다.
- ✓ $\div 4$를 하게 될 경우 4의 배수가 아니게 되어 Lemma 2의 상황이 됩니다.
- ✓ 여기서 $\times 4$를 하는 경우는 연산의 상쇄에 의한 손해가 발생합니다.
- ✓ $N \leq 3$가 될때까지 -4를 하는 경우에서도 $\div 4$를 하지 않고 -4를 해서 0을 항상 더 빠르게 만들 수 있습니다.
- ✓ 따라서 $\div 4$로는 더 적은 연산으로 0을 만들 수 없습니다.
- ✓ 따라서 ± 4의 연산만이 남습니다.

Lemma3. N **이** 4 **의 배수이고** 16 **의 배수가 아니라면**
$f(N) = \min\left(f(N - N\%16) + \frac{N\%16}{4}, f(N + 16 - N\%16) + \frac{16 - N\%16}{4}\right)$ **이다.**

- ✓ ± 4 을 적용해서 여전히 4 의 배수이고 16 의 배수가 아니라면, 다시 위의 상황과 동일하므로, 다시 ± 4 연산을 적용하게 됩니다.
- ✓ ± 4 을 적용해서 16 의 배수가 되었다면, Lemma1 에 의해 다른 연산을 적용하게 됩니다.
- ✓ 여기서 $+4, -4$ 두가지 연산을 모두 확인해 볼 필요가 있으므로 두가지 모두 시행합니다.
- ✓ 따라서 $f(N) = \min\left(f(N - N\%16) + \frac{N\%16}{4}, f(N + 16 - N\%16) + \frac{16 - N\%16}{4}\right)$ 가 됩니다.

- ✓ Lemma 2의 적용 횟수는 최초 많아야 1회 뿐입니다.
- ✓ Lemma $1, 3$을 통해 완전 탐색을 할 경우, 2번의 연산마다 많아야 2가지 분기가 생기고, 두 분기 모두 약 $1/4$배가 됩니다.
- ✓ 따라서 Lemma를 통한 완전 탐색을 할 경우의 시간복잡도는 $O(\sqrt{N})$ 입니다.
- ✓ 하지만 이 경우 전체 시간복잡도가 $O(T\sqrt{N})$ 이므로 여전히 시간안에 문제를 해결하기 어렵습니다.



- 놀랍게도 이 과정에서 참고하는 서로 다른 $f(\cdot)$의 수는 $O(\log N)$입니다!
- N을 4진수로 생각해 봅시다.
- N이 16배수인 경우에는 4진수 상에서 right shift 한 것과 동일합니다.
- N이 16배수가 아닌 4의 배수인 경우에는 왼쪽에서 첫번째 수는 0으로, 왼쪽에서 두번째 수는 0 또는 1 증가시키는것과 동일합니다.
- 첫번째 수의 승계를 고려한다면 왼쪽에서 두번째 수가 0, 1, 또는 2 증가시키는것을 케이스로 볼 수 있으므로, 많아야 3가지 입니다.
- 4진수로 나타냈을 때 $O(\log N)$자리 이므로, 관리해야하는 수의 개수는 $O(\log N)$개입니다.



- ✓ 결국 DP 로 관리하면 각 테스트 케이스에 대하여 $O(\log N)$ 시간에 해결 가능하고, 총 시간 복잡도는 $O(T \log N)$ 입니다.
- ✓ 필요한 수만 관리하는게 아닌, DP 테이블을 해시테이블이나 $BBST$ 의 자료구조로 관리하더라도 충분히 해결 가능합니다.



OC. 나무판자

`probability`, `combinatorics`, `dp`
출제진 의도 – **Challenging**

- 제출 0번, 정답 0명 (정답률 00.000%)
- First Solve: -, -분
- 출제자: 정윤성[slah007], 임병찬[hyperbolic]

- 우선 M 일 차가 끝난 뒤 각각의 i 번째 나무의 높이가 h 일 확률은 $\binom{M}{h} \times {p_i}^h \times (1-p_i)^{M-h}$ 입니다.
  각각의 i 번째 나무의 높이가 h 이하일 확률을 전처리해 둡시다.
- M 일 차가 끝난 뒤 나무판자의 넓이의 기댓값은 $\sum_{S=0}^{NM} \left\{ S \times P(area = S) \right\}$ 이고, 다른 식으로 쓰면
  $\sum_{S=0}^{NM} P(area > S)$ 와 같으므로 모든 가능한 S 에 대해 $P(area \leq S)$ 를 구하면 됩니다.



- 넓이를 결정하는 요인인 밑변과 높이를 이용하여 dp table을 정의해 봅시다.
- $dp[l][r][S][h] := l$ 번째부터 r 번째까지 나무만 있을 때 M 일 차가 끝난 뒤 모든 나무의 높이가 h 이상이면서 나무판자의 넓이가 S 이하일 확률 $(1 \leq l \leq r \leq N; 0 \leq h \leq M)$
- $(r - l + 1) \times h > S$ 인 경우 확률은 0입니다.
- $l = r$ 인 경우 l 번째 나무의 높이가 h 이상 S 이하인 확률과 같고, 전처리해 둔 값을 이용하면 $O(1)$ 에 구할 수 있습니다.

- l 번째부터 r 번째까지 나무만 있을 때 모든 나무의 높이가 h 이상이면서 i 번째에 높이가 정확히 h 인 나무가 최초로 등장하는 경우를 생각해 봅시다.
- $[l, i-1]$ 의 모든 나무의 높이는 $h+1$ 이상이고 $[i+1, r]$ 의 모든 나무의 높이는 h 이상입니다.
- $[l, i-1]$ 과 $[i+1, r]$ 을 양쪽을 연결하는 직사각형을 사용하면 $(r-l+1) \times h \leq S$ 에 해당되기 때문에, 양쪽에서 각각의 최대 넓이가 S 이하이면 전체 $[l, r]$ 에서도 최대 넓이가 S 이하입니다.
- 따라서 높이가 정확히 h 인 나무가 있으면서 나무판자의 넓이가 S 이하일 확률은 $\sum_{i=l}^{r} \left\{ dp[l][i-1][S][h+1] \times P(h_i = h) \times dp[i+1][r][S][h] \right\}$ 입니다.
- 이는 $dp[l][r][S][h] - dp[l][r][S][h+1]$ 과 같으므로, 각각의 원소를 $O(N)$ 에 구할 수 있습니다.
- 전체 table의 크기는 $O(N^3 M^2)$ 이므로 모든 값을 구하는 데 총 $O(N^4 M^2)$ 의 연산이 필요합니다.