1. 육각형 밥그릇

시간 제한 : 2초 | 메모리 제한 : 256MB

해설

"김병식"이 만든 육각형 밥그릇을 도형으로 생각해보자. 육각형 밥그릇은 한 변의 길이가 1cm인 정삼각형으로 이루어져 있다.

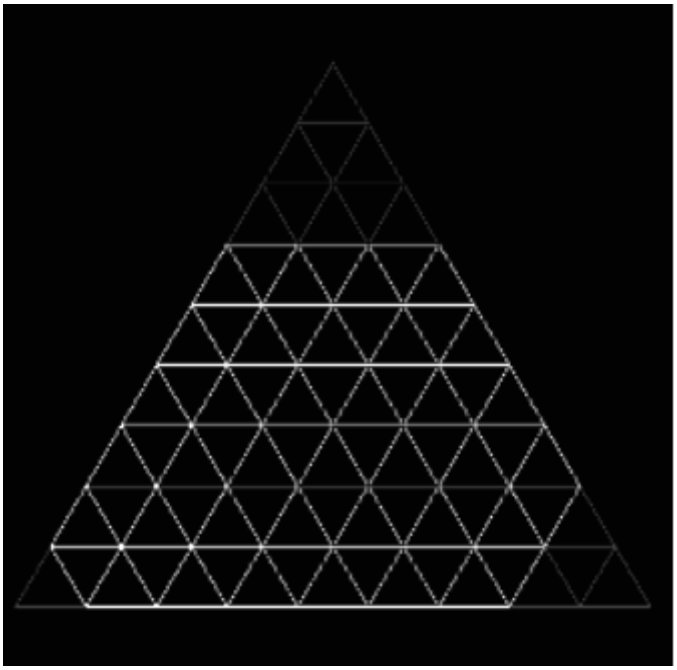
한 변의 길이가 k인 정삼각형을 k-정삼각형이라고 하자.

k-정삼각형을 각 변에 평행하게 선을 그어서 여러 개의 1-정삼각형으로 쪼개면, 가장 큰 정삼각형이 가장 작은 정삼각형보다 k^2배만큼 더 크다는 것을 알 수 있다.

따라서 k-정삼각형은 k^2개의 작은 정삼각형으로 쪼개진다.

만약에 육각형의 a_1, a_3, a_5 길이의 각 변에 정삼각형을 맞춰서 붙인다면, $a_1 + a_2 + a_3$길이의 변으로 이루어진 삼각형을 얻게 된다.

고로 육각형의 넓이는 $(a_1 + a_2 + a_3)^2 - a_1^2 - a_3^2 - a_5^2$와 같고, 이는 "김병식"이 먹을 수 있는 모든 아몬드의 개수이다.



2. 음수의 개수

시간 제한 : 1초 | 메모리 제한 : 256MB

해설

주어지는 질문에 따라서 "구간의 음수의 개수"를 구하거나, "해당하는 수의 부호를 변환"하는 문제.

수열의 길이 N이 최대 $100{,}000$이고 질문의 개수 Q 또한 최대 $100{,}000$개이므로, 매 질문마다 전체 수열을 확인하면 시간 복잡도가 $O(NQ) = 10^{10}$가 되고, 제한 시간을 초과하게 된다.

그러므로 이 문제는 **'segment tree(구간 트리)'**라는 자료구조를 사용해야한다. 이 자료구조를 사용하면 시간 복잡도 **O(QlogN)**만에 해결 할 수 있으므로 제한 시간을 초과하지 않는다.

3. 우서 더 라이트브링어

시간 제한 : 2초 | 메모리 제한 : 256MB

해설

문제를 좀더 일반화해서 표현해보자.

처음에 수열 n이 주어지고, 이를 숫자들의 합이 적어도 k 이상인 수열을 만드는데 가능한 적게 고쳐야 한다.

아무 숫자나 9로 바꾸는 것이 가장 이상적이다. 만약 우리가 숫자 d를 9로 바꿀 때, 숫자들의 합이 $9-d$만큼 증가한다. 즉, 작은 숫자들을 바꾸는 것이 이상적이다.

cnt_i는 수열 n에서 숫자 i가 나타나는 횟수라고 하자.

숫자들의 합이 k보다 작을 동안 아래와 같은 알고리즘을 반복한다.

- $cnt_i > 0$인 가장 작은 i를 찾는다.
- cnt_i를 1만큼 감소시킨다.
- 답(바꾼 숫자의 개수)을 1만큼 증가시킨다.
- 숫자들의 합을 $9-i$만큼 증가시킨다.

해당 알고리즘은 가능한 적은 횟수로 숫자들을 바꾸도록 짜여 있다.

시간 복잡도 $O(\log_{10} n)$, 공간 복잡도 $O(1)$로 해결된다.

4. 우서 더 우드커터

시간 제한 : 2초 | 메모리 제한 : 256MB

해설

이 문제의 핵심은 '나무의 길이'에 있다. 주어진 범위는 int 자료형으로 표현할 수 있는 최대치이고, 범위 내에서는 수많은 종류의 나무가 주어질 수 있다. 그렇기 때문에 0부터 1씩 더해가면서 확인해보는 brute force로 푸는 것은 불가능에 가깝다. $O(TNx) = 5 \times 10{,}000 \times (2^{31} - 1)$로 시간 제한을 초과하기 때문이다.

주어지는 나무들이 특별한 규칙을 가지고 있는 것도 아니기 때문에 출력해야 하는 나무의 길이를 Binary search의 응용인 Parametric search로 찾는 것이 시간 복잡도를 최소화 하는 방법이다.

Upper bound는 가장 큰 나무의 길이로, Lower bound는 0으로 잡고 Parametric search의 iteration을 실행한다. 이때 모든 나무의 길이를 mid의 값으로 나눈 몫의 합이 필요한 통나무의 개수보다 큰지 확인을 한다. 필요한 통나무의 개수보다 많이 나오면 통나무의 길이를 좀 더 늘려도 된다는 뜻이고, 적게 나온다면 현재의 통나무의 길이보다 작게 잘라야 한다는 것이다.

시간 복잡도 $O(TNlogx) = 5 \times 10{,}000 \times \log(2\text{^}31 - 1)$로 해결된다.

5. 소수정예 덕배

시간 제한 : 2초 | 메모리 제한 : 256MB

해설

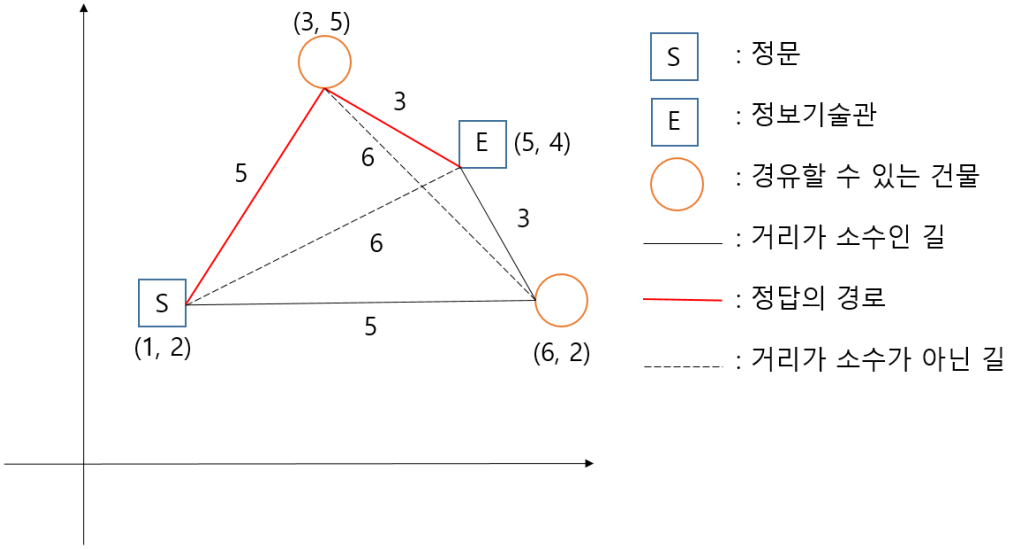
해당 문제는 소수를 판별하는 배열을 생성하고 Dijkstra 알고리즘을 이용하여 최단 이동 거리를 구할 수 있다.

- 소수 판별

덕배가 현재 위치에서 다음 위치로 이동할 수 있는 최대 거리를 M이라 하자. M의 값은 10,000을 넘지 않는다 $(0 \le |x_i|, |y_i| < 5{,}000)$.

소수는 가장 간단한 방법인 이중 반복문으로 시간 복잡도 $O(M^2)$만에 구할 수 있다. 하지만 에라토스테네스의 체를 사용하면 시간 복잡도 $O(Mlog(log(M)))$만에 구할 수 있다.

- 그래프 생성



[그림 1]

주어진 좌표 간의 거리를 모두 구해서 그래프를 만들어야 한다. (완전 그래프임을 인지해야 한다)

- Dijkstra 알고리즘

기본적인 Dijkstra 알고리즘을 통해서 최단 거리를 구해주면 된다. **단, 구현과정에서 이동할 거리가 소수가 아니라면 더 이상 우선순위 큐에 넣지 않도록 주의해야한다.**

Dijkstra 알고리즘의 시간 복잡도는 $O(ElogV)$ (단, $E: ((N+2)(N+1))/2, V: (N+2)$)

- 전체 시간 복잡도

$N \le 4{,}000, M \le 10{,}000$이므로, $O((((N+2)(N+1))/2)log(N+2)) > O(Mlog(log(M)))$가 성립한다. 그러므로 이 문제 전체의 시간 복잡도는 $O(N^2log(N))$이다(상수 제외).

6. 정보기술관의 이상한 엘리베이터

시간 제한 : 2초 | 메모리 제한 : 256MB

해설

수열의 길이 N은 최대 500, 건물의 층 수 H는 최대 1000이다.

주어진 수열의 각 자리에서 아래로 가는 방법과 위로 가는 방법 2가지 모두 가능하다고 가정하면, 일반적인 재귀함수로 작성할 경우 수열의 길이가 최대 500 이므로

2 * 2 * 2 * … = 최대 2500번의 계산이 필요 (Brute-Force - Time-limit Exceeded)

동적계획법(Dynamic Programming)으로 문제 해결해야 한다.

dp[idx][floor] : idx 번째 수열일 때, floor층을 갈 수 있는지 여부를 확인하는 dp table 생성.

ex)

만약 수열이 10 5 7이라면

dp[0][10] = true → dp[1][15] = true → dp[2][22] = true

↘ dp[2][8] = true

↘ dp[1][5] = true → dp[2][12] = true

↘ dp[2][-2] (불가능)

주의할 점

현재 층 + 수열의 값 <= H, 현재 층 - 수열의 값 >= 0 인지 확인해야함.

시간 복잡도는 dp[500][1000]을 채우는 것과 같으므로 $O(NM)$이다.

N 최대 500, H 최대 1000 이므로 $500 \times 1000 = 500{,}000$ 이므로 제한 시간안에 가능!

dp를 채우고 마지막 수열에서 갈수 있는 최대 층 수를 찾아 정답 조건에 맞게 출력.

** Bottom-Up 방식과 Top-Down 방식 둘다 가능
동일한 문제이므로 블로그 소스 코드 참고하시면 됩니다.
Top-Down 방식 : http://vjerksen.tistory.com/41 vjerksen 님 블로그
Bottom-Up 방식 : http://codersbrunch.blogspot.kr/2016/12/1495.html choyi0521 님 블로그

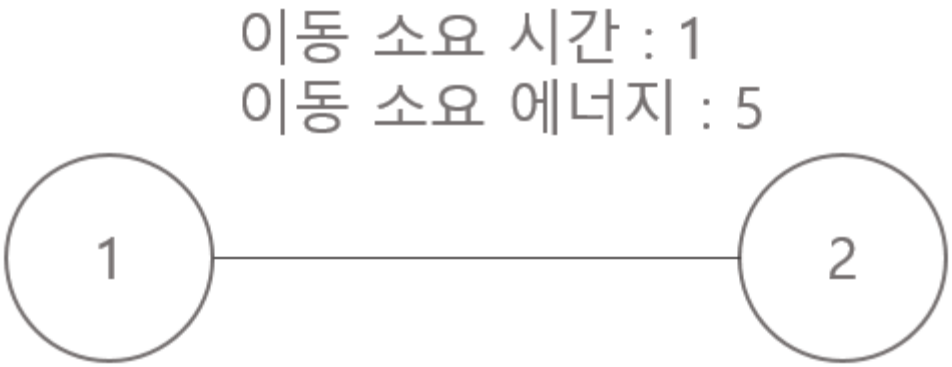
7. 광물

시간 제한 : 2초 | 메모리 제한 : 256MB

해설

해당 문제는 시작 마을에서 광물이 있는 곳까지 최소 소요 시간으로 이동하되, 보유하고 있는 에너지가 이동 시 소모값보다는 커야 이동할 수 있다는 제약을 고려해야하는 문제다. **즉 최단 거리(최소 소요 시간)를 구하는 알고리즘을 사용하되, 각 마을에서의 보유하고 있는 에너지의 상태를 저장해야 한다.**

1번 마을부터 N번 마을까지의 최소 소요 시간을 구해야 한다는 것을 인지하고 있어야한다. 그러나 **단순히 최소 소요 시간만을 구하려 하면 안된다.**



위의 그림을 참고하여 마을 1번에서 2번으로 이동한다고 생각하자. 이 때 김도매의 남은 에너지에 따라 이동 가능 여부가 달라진다. 만약 보유한 에너지가 5보다 적다면 이동하지 못한다. **이렇게 현재 상태에 따라 이동 가능 여부가 달라지므로, 현재 상태를 저장해야 한다.** 이 때 다이나믹 프로그래밍(DP, Dynamic Programming)를 사용할 수 있다. DP table은 아래와 같이 생각할 수 있다.

dp[i][j] = 마을 번호 i이고 j만큼의 에너지를 가지고있을 때의 가장 최단 경로의 길이

이 테이블을 기존의 Dijkstra 알고리즘에서 사용하던 dist배열이라 생각하면 쉽다.