Thinking Techniques - 1

고찰 테크닉 - 1

競技プログラミングの鉄則 - 米田優峻

KPSC Algorithm Study 25/1/9 Thu.

by Haru_101

Thinking Techniques

- 이번 시간부터는 문제를 풀기 위한 몇 가지 사고력을 기를 수 있는 문제들을 풀어보겠습니다.

Thinking Techniques

- 다음 문제를 풀어봅시다. (https://atcoder.jp/contests/tessoku-book/tasks/tessoku_book_aj)

問題文

$N \times N$ のマス目があります。「上下左右に隣り合うマスに移動する」という操作をちょうど K 回行うことで、左上のマスから右下のマスまで移動できるかどうかを判定してください。

制約

- $2 \leq N \leq 10^9$
- $1 \leq K \leq 10^9$
- 入力中の値はすべて整数

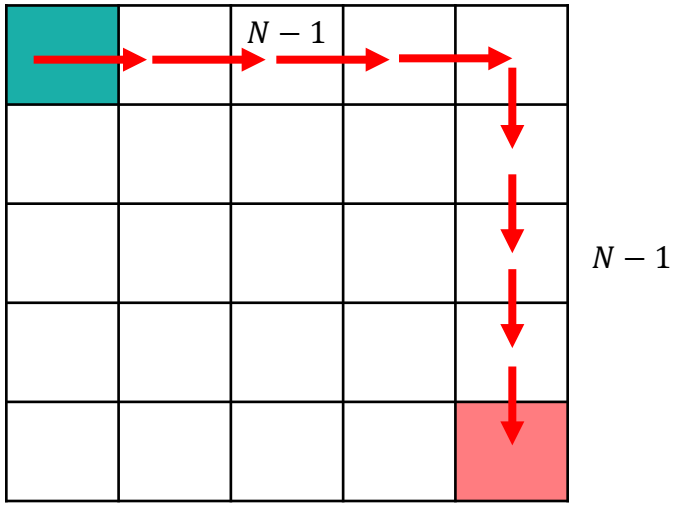
- $N \times N$ 크기의 격자판이 있습니다. 상하좌우로 인접한 칸으로 이동할 수 있는 행동을 정확히 K번 수행할 때, 왼쪽 위의 격자부터 오른쪽 아래의 격자까지 이동 가능한지 출력하는 프로그램을 작성하세요.

- 입력
  - $N\ K$

- 출력
  - 가능하면 Yes, 불가능하면 No

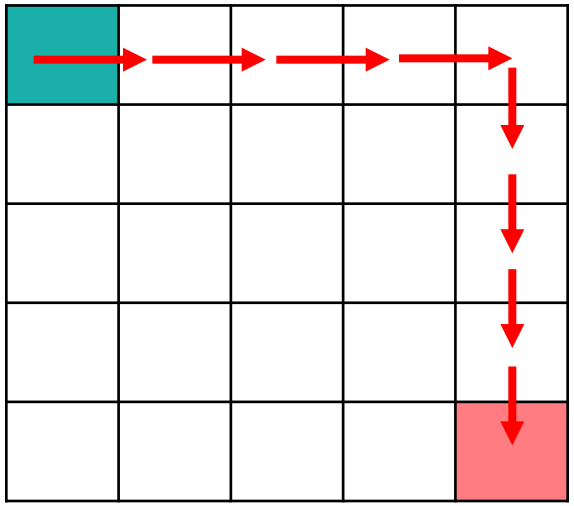
Thinking Techniques

- 먼저, 가장 왼쪽 위 격자에서 가장 오른쪽까지 이동하기 위한 최소 행동 횟수를 구해봅시다.
- 격자의 크기가 $N \times N$일때, 최소 행동 횟수는 $(N-1)+(N-1)=2N-2$가 됩니다.



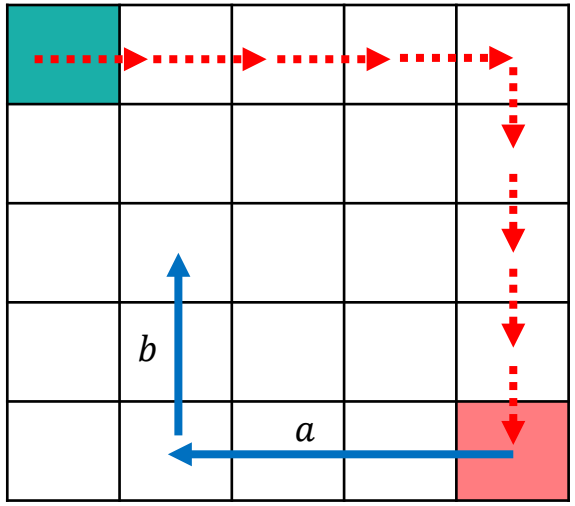
Thinking Techniques

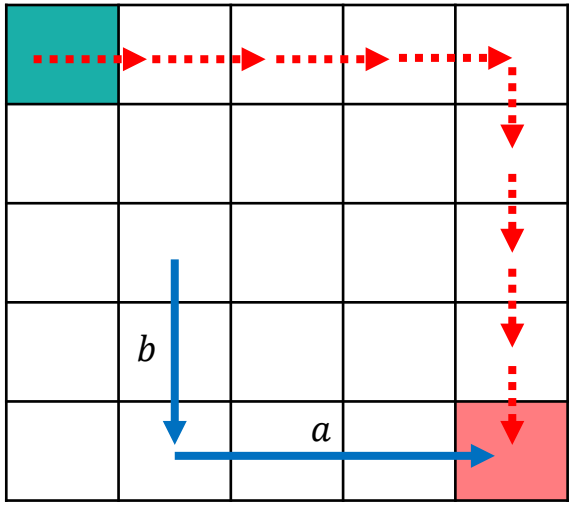
- base case로는 $K < 2N - 2$일때 절대로 이동하지 못하는 것을 확인했고, $K > 2N - 2$일때는 어떻게 되는지 한번 봐봅시다.



Thinking Techniques

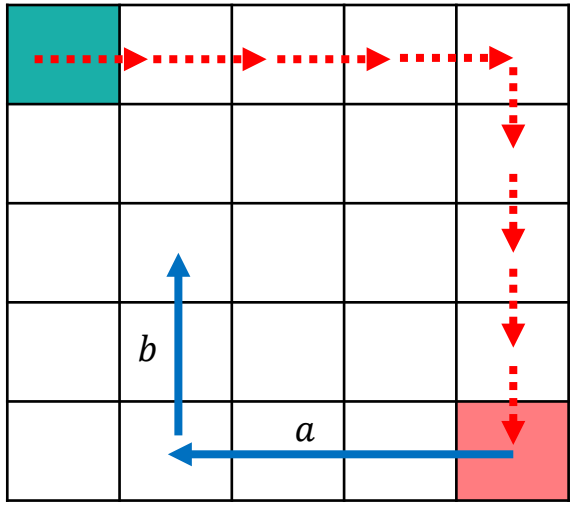
- 오른쪽 아래칸에서 지금까지의 행동을 한 횟수가 K 미만이라고 가정해봅시다.
- 그렇다면 다른 칸으로 이동해서 행동한 횟수를 늘려야 하는데, 왼쪽으로 a번, 위쪽으로 b번 이동했다면, 다시 반대로 오른쪽으로 a번 아래쪽으로 b번 이동해야 다시 원래 있던 칸으로 돌아오게 됩니다.
- 따라서 다른칸으로 이동하고 다시 오려면 $2(a+b)$번의 행동이 필요합니다.

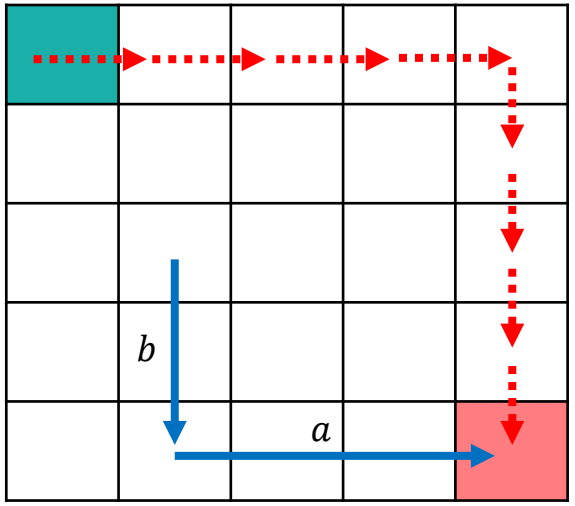




Thinking Techniques

- 모든 양의 정수에 짝수를 곱하면 그 수는 짝수가 되기 때문에 $2N - 2$에 0 이상인 아무 짝수 정수를 더해서 K가 되면 Yes가 됩니다.





Thinking Techniques

- 코드로 구현하면 다음과 같습니다.

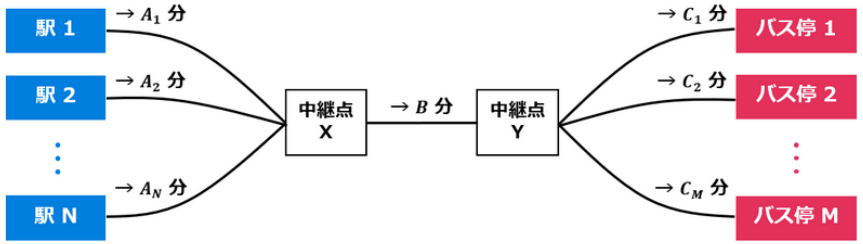
```cpp
int main() {
    fastio();
    int N, K;
    cin >> N >> K;
    if((K < 2*(N-1)) || ((K-2*(N-1))%2!=0)) {
        cout << "No";
    } else {
        cout << "Yes";
    }
}
```

Thinking Techniques

- 다음 문제를 풀어봅시다. (https://atcoder.jp/contests/tessoku-book/tasks/tessoku_book_ak)

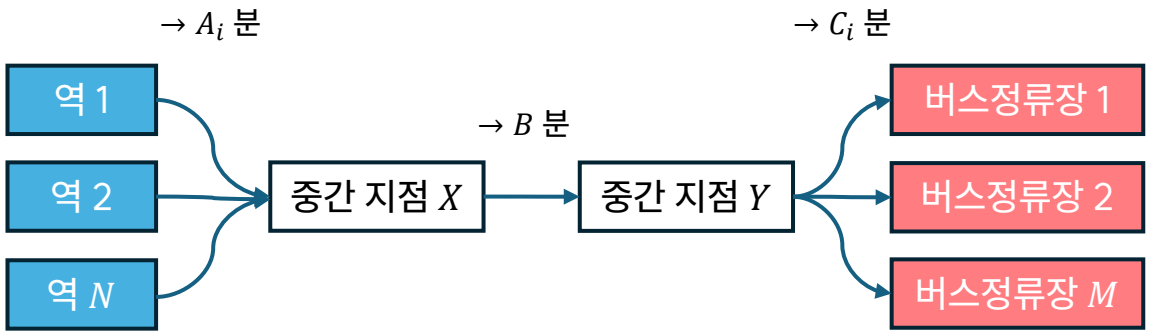
問題文

ALGO 市には N 個の駅と M 個のバス停があり、下図のように道路で結ばれています。すべての組 (i, j) に対して「駅 i からバス停 j までの所要時間」を足した値はいくつですか？



制約

- $1 \leq N, M \leq 2 \times 10^5$
- $1 \leq B \leq 100$
- $1 \leq A_i \leq 100\ (1 \leq i \leq N)$
- $1 \leq C_j \leq 100\ (1 \leq j \leq M)$
- 入力はすべて整数

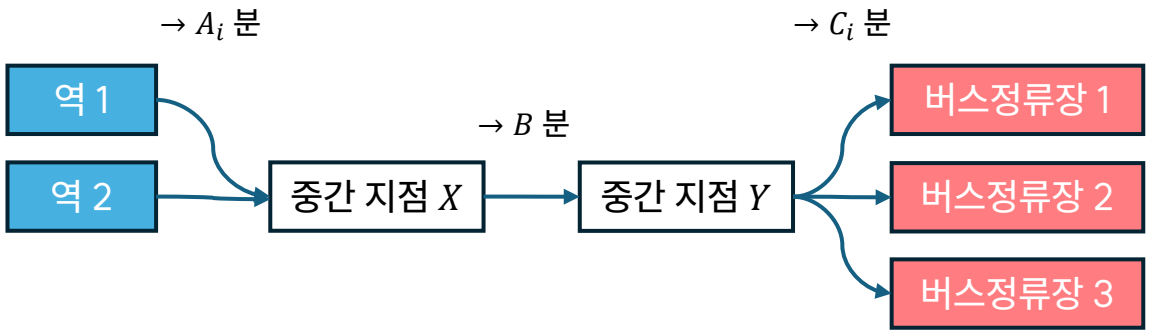


- ALGO시에는 N개의 역과 M개의 버스 정류장이 있고, 그림과 같이 도로로 연결되어 있습니다.
- 모든 쌍 (i, j)에 대해 역 i에서 버스정류장 j로 가는데에 걸리는 소요시간을 합한 값이 얼마인지 구하는 프로그램을 작성하세요.

Thinking Techniques

- $N, M \leq 200{,}000$이므로 $O(NM)$ 시간복잡도로는 풀 수 없어 보입니다.
- 일단, $N = 2, M = 3$일때 가능한 경우들을 모두 나열해봅시다.

- $(1,1) = A_1 + B + C_1$
- $(1,2) = A_1 + B + C_2$
- $(1,3) = A_1 + B + C_3$
- $(2,1) = A_2 + B + C_1$
- $(2,2) = A_2 + B + C_2$
- $(2,3) = A_2 + B + C_3$

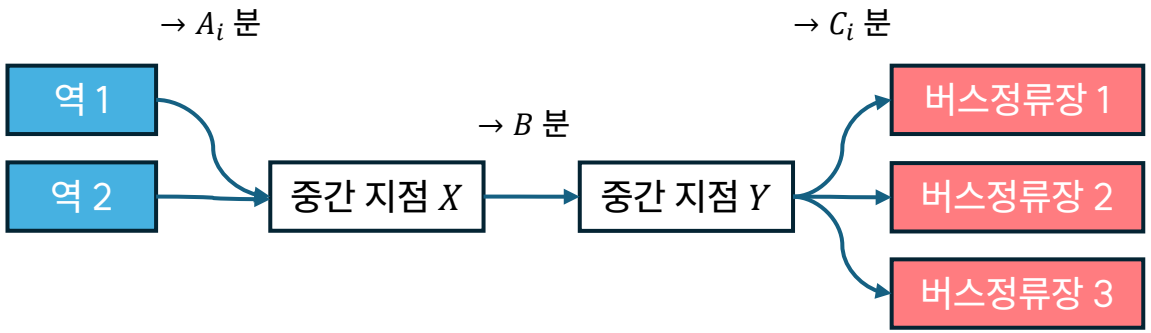


Thinking Techniques

- 이 값을 조금 정리해보면...

- $(1,1) = A_1 + B + C_1$
- $(1,2) = A_1 + B + C_2$
- $(1,3) = A_1 + B + C_3$
- $(2,1) = A_2 + B + C_1$
- $(2,2) = A_2 + B + C_2$
- $(2,3) = A_2 + B + C_3$

- $\Sigma = \sum_{i=1}^{n} MA_i + NMB + \sum_{j=1}^{m} NC_j$ 가 됩니다.



Thinking Techniques

- 코드로 구현하면 다음과 같습니다.

```cpp
int main() {
    fastio();
    ll N, M, B;
    ll sum = 0;
    cin >> N >> M >> B;
    for(int i=0; i<N; i++) {
        ll a;
        cin >> a;
        sum += M*a;
    }
    sum += N*M*B;
    for(int j=0; j<M; j++) {
        ll c;
        cin >> c;
        sum += N*c;
    }
    cout << sum;
}
```

Thinking Techniques

- 다음 문제를 풀어봅시다. (https://atcoder.jp/contests/tessoku-book/tasks/tessoku_book_al)

問題文

株式会社 KYOPRO-MASTER で働いている太郎君は、今後 D 日間の労働計画を立てることにしました。彼は今期の人事評価を上げるため、より多く働きたいと思っています。しかし、働きすぎると労働基準監督署に怒られてしまいます。具体的には、$i = 1, 2, ..., N$ に対して、以下の条件を満たす必要があります。

- 条件 i : L_i ～ R_i 日目について、最も多く働いた日でも H_i 時間以下

太郎君の D 日間の合計労働時間として考えられる最大値は何時間でしょうか。ただし、1 日は 24 時間であるものとします。

制約

- $1 \leq D \leq 365$
- $0 \leq N \leq 10000$
- $1 \leq L_i \leq R_i \leq D$
- $10 \leq H_i \leq 24$
- 入力はすべて整数

- 주식회사 KYOPRO-MASTER에서 일하는 타로군은 지금부터 D일간 노동계획을 세우고 있었습니다. 그는 이번 분기의 인사평가를 높게 받기 위해 많이 일할려고 합니다. 하지만, 너무 많이 일하면 노동기준감독관이 꾸짖습니다.
- 구체적으로는 $i = 1, 2, \cdots, N$에 대해서 아래 조건을 만족해야합니다.
  - 조건 $i : L_i \sim R_i$일에 대해서 최고로 많이 일한 날의 근로 시간이 H_i시간 이하
- 타로군의 D일간의 가능한 노동시간 합의 최댓값은 얼만지 구하는 프로그램은 작성하세요. 단, 1일은 24시간으로 칩니다.

Thinking Techniques

- $L_i \sim R_i$일까지는 H_i시간 이하로 일하면서 합이 최대가 되도록 해야합니다.

- 조건이 없을 때는 모든 일에 24시간동안 일할 수 있으므로, 조건들이 주어지면 그 날에 일할 수 있는 시간을 min을 취해나가면서 줄이고, $1, 2, \cdots, D$일에 대해서 일할 수 있는 시간을 다 더하면 됩니다.

- 따라서 $O(ND + D)$에 문제를 해결할 수 있습니다.

Thinking Techniques

- 코드로 구현하면 다음과 같습니다.

```cpp
int main() {
    fastio();
    int D, N;
    cin >> D >> N;
    for(int i=1; i<=D; i++) {
        arr[i] = 24;
    }
    for(int i=0; i<N; i++) {
        int L, R, H;
        cin >> L >> R >> H;
        for(int j=L; j<=R; j++) {
            arr[j] = min(arr[j], H);
        }
    }
    int ans = 0;
    for(int i=1; i<=D; i++) {
        ans += arr[i];
    }
    cout << ans;
}
```

Thinking Techniques

- 다음 문제를 풀어봅시다. (https://atcoder.jp/contests/tessoku-book/tasks/tessoku_book_an)

問題文

机の上に N 本の棒が置かれています。左から i 番目の棒（棒 i とする）の長さは A_i メートルです。

3 つの異なる棒を選んで**正三角形**を作る方法は何通りありますか。

制約

- $3 \leq N \leq 2 \times 10^5$
- $1 \leq A_i \leq 100$
- 入力は**すべて整数**

- 책상에 N개의 막대가 놓여있습니다. 왼쪽부터 i번째 막대라고 할때 그 막대의 길이는 A_i미터입니다. (A_i는 정수)
- 3개의 서로 다른 막대를 선택해서 정삼각형을 만들 수 있는 방법은 몇 개 있는지 출력하는 프로그램을 작성하세요.

Thinking Techniques

- 이 문제를 그냥 $O(N^3)$으로 풀기에는 $N \leq 200{,}000$이라서 누가 봐도 터질 것 같습니다.

- 그렇다면 이 문제를 어떻게 접근하는 것이 좋을까요?

Thinking Techniques

- 먼저 서로 다른 N개에서 3개를 고르는 경우의 수를 구해봅시다.
- 조합 공식에 의하면 $nC_3 = \frac{n(n-1)(n-2)}{3 \cdot 2 \cdot 1} = \frac{n(n-1)(n-2)}{6}$이 됩니다.
- 따라서 길이가 k인 막대 p개를 이용해서 정삼각형을 만드는 방법의 수는 $\frac{p(p-1)(p-2)}{6}$이 됩니다.
- $1 \leq A_i \leq 100$이므로, 길이가 A_i인 막대들의 개수를 세고 위 식을 이용해서 계산하면 됩니다.

Thinking Techniques

- 코드로 구현하면 다음과 같습니다.

```cpp
ll arr[105];
int main() {
    fastio();
    int N;
    cin >> N;
    for(int i=0; i<N; i++) {
        int x;
        cin >> x;
        arr[x]++;
    }
    ll ans = 0;
    for(int i=1; i<=100; i++) {
        if(arr[i]>=3) {
            ans += (arr[i]*(arr[i]-1)*(arr[i]-2))/6;
        }
    }
    cout << ans;
}
```

Thinking Techniques - 1

끝.

다음주에는 Thinking Techniques - 2로 찾아뵙겠습니다.