

Mathematics Basic - 2

수학 기초 2

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

Mathematics

- 이번 시간에는 게임을 해야하는 상황에서 누가 이기는지 판단하는 문제들에 대한 접근법을 소개하겠습니다.

Mathematics

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

問題文

N 個の石が積まれた山があり、2 人のプレイヤーが交互に石を取り合います。
各プレイヤーは1回のターンで、以下のいずれかの操作をすることができます。

- 山から A 個の石を取り除く。
- 山から B 個の石を取り除く。

先に石を取り除けなくなった方が負けです。両者が最善を尽くしたとき、先手と後手どちらが勝ちますか。

制約

- N, A, B は整数
- $2 \leq N \leq 100000$
- $1 \leq A < B \leq N$

- N 개의 돌이 쌓여있는 탑이 있고, 2명의 플레이어가 교대로 돌을 제거하고자 합니다.
- 각 플레이어는 턴마다 다음 행동들 중 하나를 수행해야 합니다.
 - 탑에서 정확히 A 개의 돌을 제거한다.
 - 탑에서 정확히 B 개의 돌을 제거한다.
- 먼저 자기 차례에 돌을 제거할 수 없는 경우 그 사람이 집니다. 두 플레이어 모두 최선을 다할때, 선공, 후공 중 누가 이기는 지 출력하는 프로그램을 작성하세요.

Mathematics


- 먼저 이 게임에서 존재하는 base case들을 생각해봅시다.
- 일단, 표를 돌탑에 남아있는 돌의 개수가 n 일때, 해당 턴의 플레이어가 이기는지, 지는지 여부를 저장한다고 합시다.
- $A = 2, B = 3$ 이라고 해봅시다. (문제 입력1 예시)
- 문제 조건에서 $A < B$ 이므로 현재 돌탑에 남아있는 돌의 개수가 $0 \sim A - 1$ 개인 경우 그때의 플레이어가 집니다. (제거를 하지 못하므로)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
n	L	L																	

Mathematics

- 그 다음, $n = 2$ 일때를 생각해봅시다.
- $n = 2$ 일때 해당 턴의 플레이어는 2개의 돌을 제거하거나 3개의 돌을 제거하는 행동 중 하나를 해야하는데, 돌탑에는 돌이 2개밖에 없습니다.
- 플레이어들은 항상 최선의 행동을 합니다. 따라서 해당 턴의 플레이어는 2개의 돌을 반드시 제거해야합니다.
- 이때 $n = 2 - 2 = 0$ 은 지는 상황이므로 $n = 2$ 일때의 플레이어는 이기게 됩니다.


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
n	L	L	W																



Mathematics

- 그 다음, $n = 3$ 일때를 생각해봅시다.
- $n = 3$ 일때 해당 턴의 플레이어는 2개의 돌을 제거하거나 3개의 돌을 제거하는 행동 중 하나를 해야하는데, 돌탑에는 3개의 돌이 있으므로 두 행동 모두 선택할 수는 있습니다.
- 플레이어들은 항상 최선의 행동을 합니다. 따라서 해당 턴의 플레이어는 최대한 다음 턴의 플레이어가 지도록 유도해야하므로, $n - 2, n - 3$ 중에 하나라도 L이라면 W, 아니라면 L을 적으면 됩니다.
- 이때 $n = 3 - 2 = 1$ 은 지는 상황이고 $n = 3 - 3 = 0$ 도 지는 상황이므로, $n = 3$ 일때의 플레이어는 이기게 됩니다.


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
n	L	L	W	W															



Mathematics

- 그 다음, $n = 4$ 일때를 생각해봅시다.
- $n = 4$ 일때 해당 턴의 플레이어는 2개의 돌을 제거하거나 3개의 돌을 제거하는 행동 중 하나를 해야하는데, 돌탑에는 4개의 돌이 있으므로 두 행동 모두 선택할 수 있습니다.
- 이때 $n = 4 - 2 = 2$ 은 이기는 상황이고 $n = 4 - 3 = 1$ 은 지는 상황입니다.
- 해당 턴의 플레이어는 내가 돌을 제거해서 다음 턴의 플레이어가 지게 해야하므로, 3개를 빼서 다음 턴 플레이어를 지게 할 수 있습니다. 따라서 W가 됩니다.


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
n	L	L	W	W	W														



Mathematics

- 그 다음, $n = 5$ 일때를 생각해봅시다.
- $n = 5$ 일때 해당 턴의 플레이어는 2개의 돌을 제거하거나 3개의 돌을 제거하는 행동 중 하나를 해야하는데, 돌탑에는 5개의 돌이 있으므로 두 행동 모두 선택할 수 있습니다.
- 이때 $n = 5 - 2 = 3$ 도 이기는 상황이고 $n = 5 - 3 = 2$ 는 이기는 상황입니다.
- 해당 턴의 플레이어는 내가 돌을 제거해서 다음 턴의 플레이어가 지게 해야하는데, 뭘 하든 다음 턴의 플레이어가 이기게 됩니다. 따라서 L이 됩니다.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
n	L	L	W	W	W	L													



Mathematics

- 나머지 표를 채우면 다음과 같습니다.
- 근데, 이거 DP랑 비슷하지 않나요?

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
n	L	L	W	W	W	L	L	W	W	W	L	L	W	W	W	L	L	W	W

Mathematics

- 이를 코드로 구현하면 다음과 같습니다. ($i - B \geq 0$ 인지 체크 필수!!)

```
bool dp[100005];
int main() {
    fastio();
    int N, A, B;
    cin >> N >> A >> B;
    for(int i=A; i<=N; i++) {
        if(i>=B) {
            if(dp[i-A] && dp[i-B]) {
                dp[i] = false;
            } else {
                dp[i] = true;
            }
        } else if(i>=A) {
            if(dp[i-A]) {
                dp[i] = false;
            } else {
                dp[i] = true;
            }
        }
    }
    cout << (dp[N] ? "First" : "Second");
}
```

Mathematics

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

問題文

石の山が N 個あり、山 i ($1 \leq i \leq N$) には A_i 個の石が積まれています。

このゲームでは、2 人のプレイヤーが交互に次の操作を行います。

- 好きな石の山を 1 つ選び、選んだ山から 1 個以上の石を取る。

すべての石がなくなり、操作を行えなくなった方が負けです。

両者が最善を尽くしたとき、先手と後手どちらが勝ちますか。

制約

- 入力は全て整数
- $2 \leq N \leq 100000$
- $1 \leq A_i \leq 10^9$

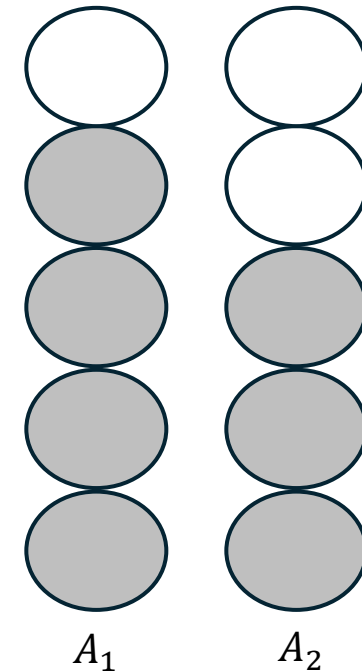
- A_i 개의 돌이 쌓여있는 탑이 N 개 있고, 2명의 플레이어가 교대로 돌을 제거하고자 합니다.
- 각 플레이어는 자기 턴마다 다음 행동들 중 하나를 수행해야 합니다.
 - N 개의 탑 중 원하는 탑 하나를 골라서 탑에서 정확히 1개 이상의 돌을 제거한다.
- 먼저 자기 차례에 돌을 제거할 수 없는 경우 그 사람이 집니다. 두 플레이어 모두 최선을 다할때, 선공, 후공 중 누가 이기는 지 출력하는 프로그램을 작성하세요.

Mathematics

- 먼저 base case들을 보겠습니다.
- 돌탑이 1개일때는, 선공이 모든 돌을 가져가면 후공이 돌을 가져가지 못하므로, 선공이 반드시 이깁니다.
- 돌탑이 2개일때는 조금 복잡한데, 선공이 가져간 돌의 개수만큼 후공도 그만큼 가져가는 전략을 선택하면 후공이 반드시 이기게 됩니다.
- 뭔가 애매하니 그림으로 살펴보겠습니다.

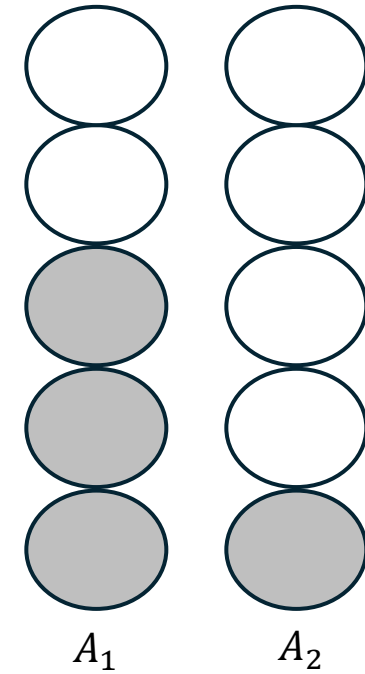
Mathematics

- 탑 2개에 각각 $A_1 = 5, A_2 = 5$ 개의 돌이 있는 경우를 봅시다.
- 먼저 선공이 1개, 후공이 선공과 다른 개수(ex. 2개)의 돌을 가져가는 경우를 봅시다.
- 선공 $A_1 \rightarrow -1 = 4$, 후공 $A_2 \rightarrow -2 = 3$



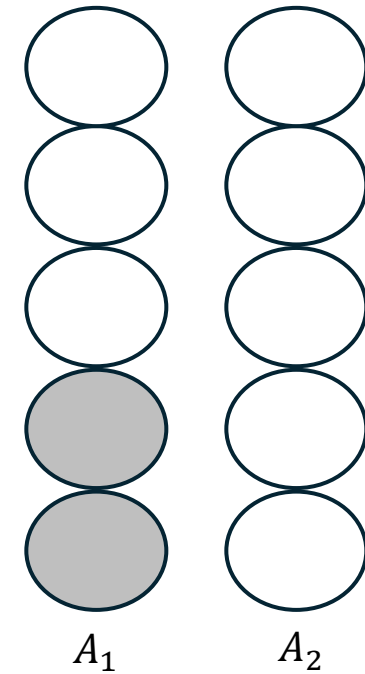
Mathematics

- 선공 $A_1 \rightarrow -1 = 3$, 후공 $A_2 \rightarrow -2 = 1$



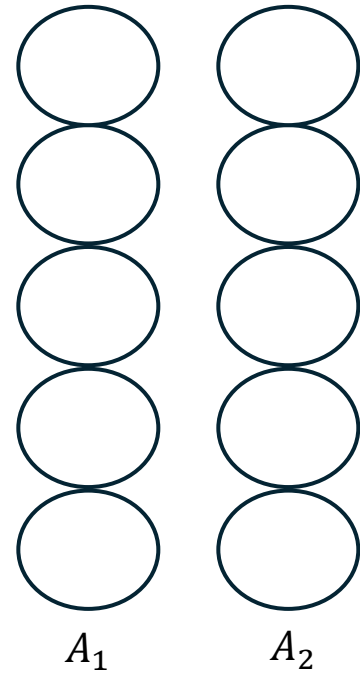
Mathematics

- 선평 $A_1 \rightarrow -1 = 2$, 후공 A_2 or $A_1 \rightarrow -1 = 0$



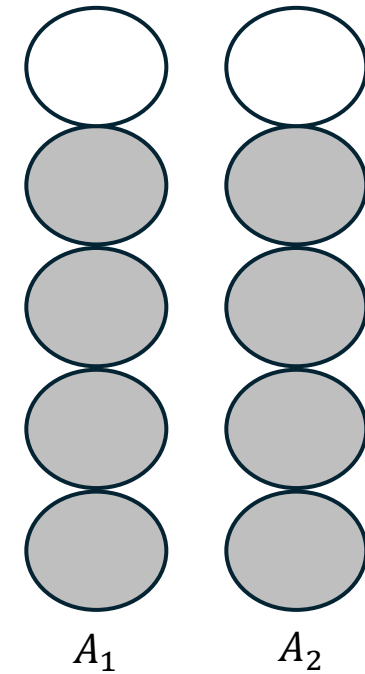
Mathematics

- 선공 $A_1 \rightarrow -2 = 0$, 후공 $A_1 = A_2 = 0$ 이므로 후공 패배



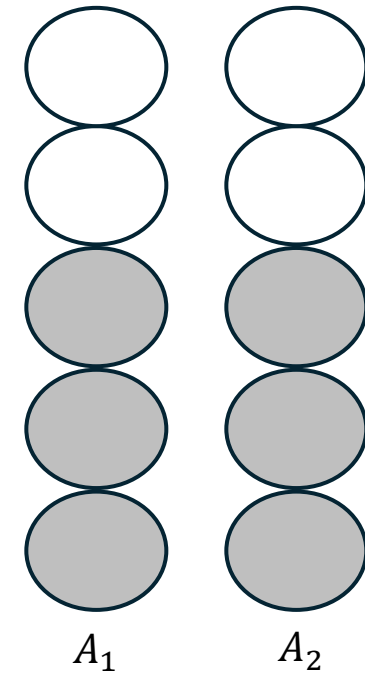
Mathematics

- 이제 선공이 1개, 후공이 선공과 같은 개수의 돌을 가져가는 경우를 봅시다.
- 선공 $A_1 \rightarrow -1 = 4$, 후공 $A_2 \rightarrow -1 = 4$



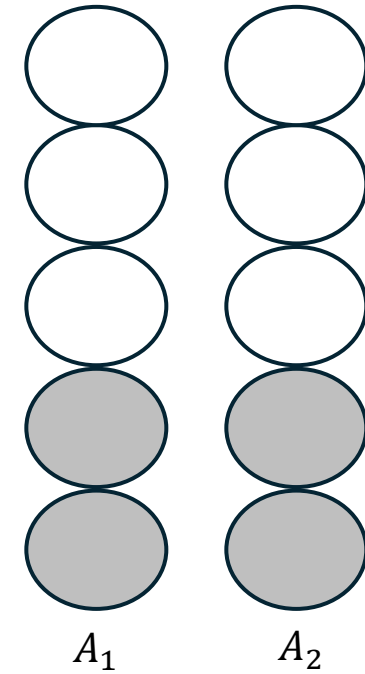
Mathematics

- 선공 $A_1 \rightarrow -1 = 3$, 후공 $A_2 \rightarrow -1 = 3$



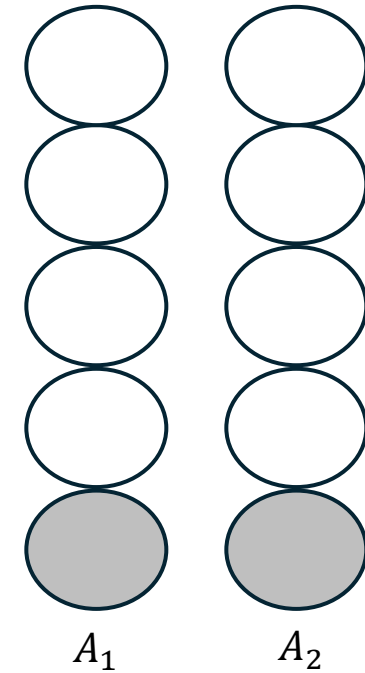
Mathematics

- 선공 $A_1 \rightarrow -1 = 2$, 후공 $A_2 \rightarrow -1 = 2$



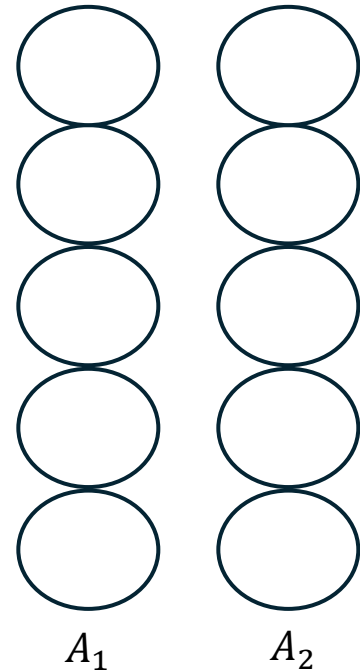
Mathematics

- 선공 $A_1 \rightarrow -1 = 1$, 후공 $A_2 \rightarrow -1 = 1$
- 참고로 여기서 갑자기 선공이 2개의 공을 가져가도 후공이 2개의 공을 가져가면 후공이 이깁니다.



Mathematics

- 선공 $A_1 \rightarrow -1 = 0$, 후공 $A_2 \rightarrow -1 = 0$
- 이제 선공이 가져갈 돌이 없으므로 후공 승리
- 이렇게 선공이 가져간 돌의 개수만큼 후공이 똑같이 가져가는 것을 모방 전략(흉내 전략(원서 직역))이라고 합니다.
- 이 전략이 통하려면 $A_1 = A_2$ 여야 하며, $A_1 \neq A_2$ 인 경우 선공이 항상 이기게 됩니다.
- 왜냐하면, 선공이 $A_1 = A_2$ 가 되도록 돌을 가져가면 선공이 후공에 대해 모방 전략을 이용해서 돌을 가져가면 선공이 이기기 때문입니다.



Mathematics

- 지금까지는 탑의 개수가 1, 2개인 경우만 보았는데, 탑이 3개 이상인 경우에는 어떻게 구할 수 있을까요?
- 결론적으로 얘기하면 $A_1 \text{ XOR } A_2 \text{ XOR } \dots \text{ XOR } A_N$ 을 님 합이라고 정의하였을때, 님 합이 0이면 후공이 이기고, 0이 아니면 선공이 이긴다고 판정하는 판정법을 사용하면 됩니다.
- 필승법에 따르면 다음 차례의 플레이어가 지는 상황이 하나라도 있다면 그 상황으로 가야 내가 이기게 됩니다.
- 님 합이 0이라면 0이 아닌 곳 반드시 가게 되고, 0이 아니었다면 0인 곳(2)으로 갈 수 있습니다.

Mathematics

- 이제 (1), (2)에 대한 증명을 살펴보겠습니다. 원서에 따르면 다음과 같이 증명할 수 있습니다.
- $N = 4, (A_1, A_2, A_3, A_4) = (4, 5, 6, 7)$ 를 이용해 (1)부터 증명해봅시다.
- 먼저, $4 \text{ XOR } 5 \text{ XOR } 6 \text{ XOR } 7 = 100_2 \text{ XOR } 101_2 \text{ XOR } 110_2 \text{ XOR } 111_2 = 000_2 = 0$ 입니다.
- 즉, 이진법 표기의 모든 자리에 대해 1의 개수가 짝수개 임을 뜻합니다. ($1 \text{ XOR } 1 = 0$)

	4	2	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
1 비트 개수	4개	2개	2개
XOR	0	0	0

Mathematics

- 이 문제에서는 돌을 추가하는 것이 아닌 제거하기 때문에 반드시 어떤 자리들에 대해 1의 개수가 변하게 됩니다.
- 이 경우 짝수 → 홀수가 되므로 XOR한 결과에 대해 그 비트는 켜지게 되므로 XOR 값이 0이 아니게 됩니다.
- 따라서 님 합이 0인 경우에는 돌을 어떻게 제거하더라도 님 합이 0이 아닌 상태로 변하게 됩니다.

	4	2	1
4	1	0	0
5	1	0	1
4	1	0	0
7	1	1	1
1 개수	4개	1개	2개
XOR	0	1	0

Mathematics

- 이제 $N = 4$, $(A_1, A_2, A_3, A_4) = (4, 6, 8, 9)$ 를 이용한 (2)에 대한 증명을 봐봅시다.
- $4 \text{ XOR } 6 \text{ XOR } 8 \text{ XOR } 9 = 0100_2 \text{ XOR } 0110_2 \text{ XOR } 1000_2 \text{ XOR } 1001_2 = 0011_2 = 3$ 입니다.
- 님 합을 0으로 만들기 위해서는 A_1 을 7로 만들거나, A_2 를 5로 만들거나, A_3 를 11로 만들거나, A_4 를 10으로 만들면 됩니다.
- 하지만, 이 문제에서는 돌이 증가하도록 행동을 취할 수 없습니다. 따라서 A_2 를 5로 만들면 됩니다.

	8	4	2	1
4	0	1	0	0
6	0	1	1	0
8	1	0	0	0
9	1	0	0	1
1개수	2개	2개	1개	1개
XOR	0	0	1	1

	8	4	2	1
4	0	1	0	0
5	0	1	0	1
8	1	0	0	0
9	1	0	0	1
1개수	2개	2개	0개	2개
XOR	0	0	0	0

Mathematics

- 근데 어떤 돌탑을 골라 몇 개의 돌을 제거해야할까요?
- XOR값에서 1인 가장 왼쪽 자리를 선택하고, 그 자리에 1이 있는 돌탑들 중 하나를 선택하면 됩니다.
- $A_2 = 6$ 에 대해 해당 자리는 0으로 바꾸고, 그 자리의 오른쪽 자리들에 대해선 1의 개수가 홀수면 짝수로, 짝수면 그대로 냅두면 됩니다.
- 이렇게 돌을 제거하면 1개 이상 제거한것과 같고, 님 합은 0이 됩니다.

	8	4	2	1
4	0	1	0	0
6	0	1	1	0
8	1	0	0	0
9	1	0	0	1
1개수	2개	2개	1개	1개
XOR	0	0	1	1

	8	4	2	1
4	0	1	0	0
5	0	1	0	1
8	1	0	0	0
9	1	0	0	1
1개수	2개	2개	0개	2개
XOR	0	0	0	0

Mathematics

- 모든 돌탑에 돌이 없는 경우는 님 합이 0입니다.
- 따라서, 선공이 님 합을 0으로 만들지 못하면(1) 후공이 님 합을 0으로 만들 수 있기 때문에(2) 후공이 승리하고, 선공이 님 합을 0으로 만들 수 있으면(2) 후공이 님 합을 0으로 만들 수 없기 때문에(1) 선공이 승리하게 됩니다.
- 즉, 간단하게 님 합이 0인지 아닌지를 통해 누가 승리하는지 간단하게 판단할 수 있습니다.

Mathematics

- 이를 코드로 구현하면 다음과 같습니다.

```
int arr[100005];
int main() {
    fastio();
    int N;
    cin >> N;
    for(int i=0; i<N; i++) {
        cin >> arr[i];
    }
    int nim = arr[0];
    for(int i=1; i<N; i++) {
        nim ^= arr[i];
    }
    cout << (nim==0 ? "Second" : "First");
}
```

Mathematics

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

問題文

石の山が N 個あり、山 $i (1 \leq i \leq N)$ には A_i 個の石が積まれています。

このゲームでは、2 人のプレイヤーが交互に次の操作を行います。

- 好きな石の山を 1 つ選び、選んだ山から X 個または Y 個の石を取る。

すべての山にある石の数が X 個未満になり、操作を行えなくなった方が負けです。

両者が最善を尽くしたとき、先手と後手どちらが勝ちますか。

制約

- 入力は全て整数
- $1 \leq N \leq 100000$
- $1 \leq X < Y \leq 100000$
- $1 \leq A_i \leq 100000$

- A_i 개의 돌이 쌓여있는 탑이 N 개 있고, 2명의 플레이어가 교대로 돌을 제거하고자 합니다.
- 각 플레이어는 자기 턴마다 다음 행동들 중 하나를 수행해야 합니다.
 - N 개의 탑 중 원하는 탑 하나를 골라서 탑에서 정확히 X 개 혹은 Y 개의 돌을 제거한다.
- 먼저 자기 차례에 돌을 제거할 수 없는 경우 그 사람이 집니다. 두 플레이어 모두 최선을 다할때, 선공, 후공 중 누가 이기는 지 출력하는 프로그램을 작성하세요.

Mathematics

- 아까 푼 문제와 달리 제거할 수 있는 돌의 개수가 X, Y 로 정해져있습니다.
- 원서에서도 이 파트에 대해서는 자세하게 증명하지 않았으므로, 본 ppt에서도 자세하게 증명하지는 않겠습니다.
- (스프라그-그런디 정리, 조금 더 자세한 증명을 원하시면 [링크](#))
- 그런디(Grundy)수는 게임 현황을 나타내는 정수로서, 다음과 같이 정의합니다.
- 한번의 행동을 통해 x_1, x_2, \dots, x_k 상태로 이동할 수 있을 때, 현재 상태의 그런디 수는 x_1, x_2, \dots, x_k 가 아닌 가장 작은 음이 아닌 정수로 정의합니다.
- 예를 들어 x_1, x_2, x_3 가 1, 2, 3이라면 그런디 수는 0이고, 0, 1, 3이면 그런디 수는 2입니다.

Mathematics


- 아까 필승법에서 본 것과 비슷하게 각 돌탑에 대해 그런디 수 테이블을 채워봅시다.
- 먼저 $X = 2, Y = 3$ 으로 표를 채워봅시다.
- 일단, $A_i < X$ 일때는 아무런 행동을 취할 수 없으므로 그런디 수가 0입니다.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
A_i	0	0																	

Mathematics

- 그 다음, $A_i = 2$ 일때는 돌을 2개 제거하는 행동만 할 수 있으므로 그런디 수는 정의에 따라 $A_i = 0$ 에서의 그런디 수인 0이 아니면서 음수가 아닌 최소의 정수인 1이 됩니다.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
A_i	0	0	1																



Mathematics


- 그 다음, $A_i = 3$ 일때는 돌을 2개 혹은 3개를 제거할 수 있으므로 그런디 수는 정의에 따라 $A_i = 0, A_i = 1$ 에서의 그런디 수들인 0이 아니면서 음수가 아닌 최소의 정수인 1이 됩니다.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
A_i	0	0	1	1															

Mathematics

- 그 다음, $A_i = 4$ 일때는 돌을 2개 혹은 3개를 제거할 수 있으므로 그런디 수는 정의에 따라 $A_i = 1, A_i = 2$ 에서의 그런디 수들인 0, 1이 아니면서 음수가 아닌 최소의 정수인 2가 됩니다.


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
A_i	0	0	1	1	2														



Mathematics

- 그 다음, $A_i = 5$ 일때는 돌을 2개 혹은 3개를 제거할 수 있으므로 그런디 수는 정의에 따라 $A_i = 2, A_i = 3$ 에서의 그런디 수들인 1이 아니면서 음수가 아닌 최소의 정수인 0이 됩니다.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
A_i	0	0	1	1	2	0													



Mathematics

- 이를 끝까지 채우면 다음과 같게 됩니다.
- 이 표를 이용하면 A_i 에서 더 이상 해당 돌탑에서 행동을 취하지 못하도록 하는 상태로 이동할 수 있는지 여부를 알 수 있습니다.
- 더 이상 행동을 취하지 못하는 상태는 0인데, 만약 그런디 수가 0이 아니라면 행동을 취하고 난 후의 그런디 수에 0이 포함되어 있는 것이기 때문에 0으로 만들 수 있다고 판단이 가능해집니다.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
A_i	0	0	1	1	2	0	0	1	1	2	0	0	1	1	2	0	0	1	1

Mathematics

- 이제 $G[A_i]$ 는 위 과정을 통해 구할 수 있게 되었으므로 다음을 계산해서 승패를 판정하면 됩니다.
- $G[A_1] XOR G[A_2] XOR \dots XOR G[A_N] = 0$ 이면 후공 승리
- $\dots \neq 0$ 이면 선공 승리
- 아까 님 합 증명과 비슷하게 내가 0을 못만들면 진다라는 아이디어 정도로만 알아두시면 될 것 같습니다.
- 혹은 승리할 수 있는 경우가 홀수개고 내가 선공이면 반드시 내가 이긴다?

Mathematics

- 이를 코드로 구현하면 다음과 같습니다.

```
int grundy[100005];
int arr[100005];
int main() {
    fastio();
    int N, A, B;
    cin >> N >> A >> B;
    for(int i=0; i<N; i++) {
        cin >> arr[i];
    }
    for(int j=A; j<=100000; j++) {
        bool grundy_check[3] = {false, false, false};
        if(j >= B) {
            grundy_check[grundy[j-B]] = true;
            grundy_check[grundy[j-A]] = true;
        } else if(j >= A) {
            grundy_check[grundy[j-A]] = true;
        }
        for(int t=0; t<=2; t++) {
            if(grundy_check[t] == false) {
                grundy[j] = t;
                break;
            }
        }
    }
    int grundy_xor = grundy[arr[0]];
    for(int i=1; i<N; i++) {
        grundy_xor ^= grundy[arr[i]];
    }
    cout << (grundy_xor != 0 ? "First" : "Second");
}
```

Mathematics

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

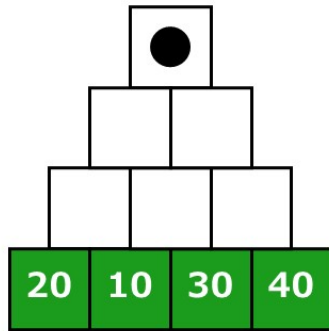
問題文

下図のような N 段のピラミッドがあり、最下段には左から順に整数 A_1, A_2, \dots, A_N が書かれています。また、最上段には1つのコマが置かれています。

太郎君と次郎君は、このピラミッドを使ってゲームをします。コマが最下段に到達するまで、各プレイヤーは交互に以下のいずれかの操作を行います(太郎君が先手です)。

- コマを左下方向に1つ移動させる。
- コマを右下方向に1つ移動させる。

ゲーム終了時のコマの位置に書かれた整数を **スコア** とします。太郎君はスコアを最大化し、次郎君はスコアを最小化するとき、スコアはいくつになりますか。



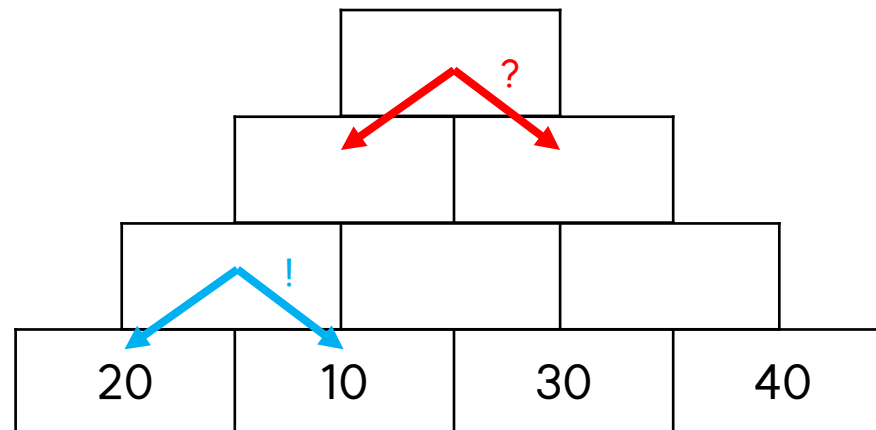
制約

- 入力は全て整数
- $2 \leq N \leq 2000$
- $1 \leq A_i \leq 100000$

- 그림과 같이 N 층짜리 피라미드가 있고, 최하단에는 정수 A_1, A_2, \dots, A_N 이 적혀 있습니다.
- 또한, 최상단에는 1개의 점이 놓여 있습니다.
- 타로군과 치로군은 이 피라미드를 이용해서 게임을 하는데, 점이 최하단으로 도달할 때 까지 각 플레이어는 교대로 아래 행동 중 하나를 수행합니다. (타로군이 먼저)
 - 점을 왼쪽 아래방향에 한칸 이동시킨다.
 - 점을 오른쪽 아래방향에 한칸 이동시킨다.
- 게임이 종료될때의 점이 위치한 곳에 써져있는 점수를 '스코어'라고 합시다.
- 타로군은 스코어를 최대화하고, 치로군은 스코어를 최소화 하고 싶어할때, 스코어는 얼마가 되는지 구하는 프로그램을 작성하세요.

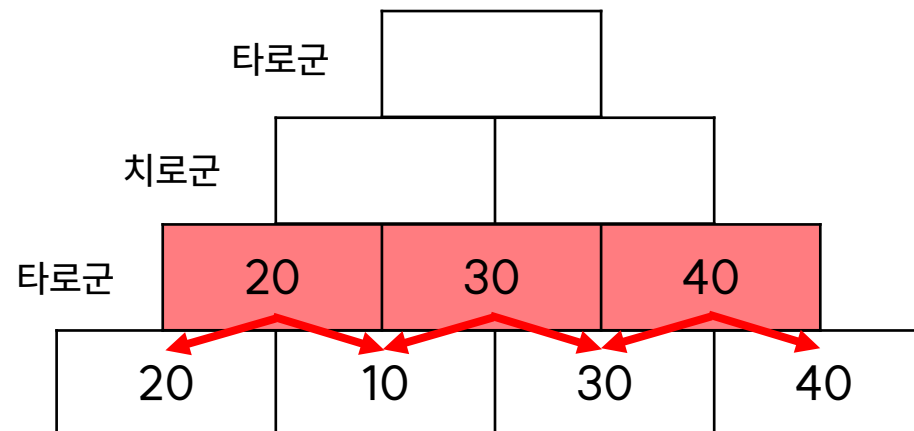
Mathematics

- 먼저 위에서 아래로 내려가면서 최대화, 최소화되는 방향이 어딘지는 구하기가 힘들어 보입니다.
- 하지만, 가장 아래에서 위로 올라가는 것은 쉽지 않을까요?



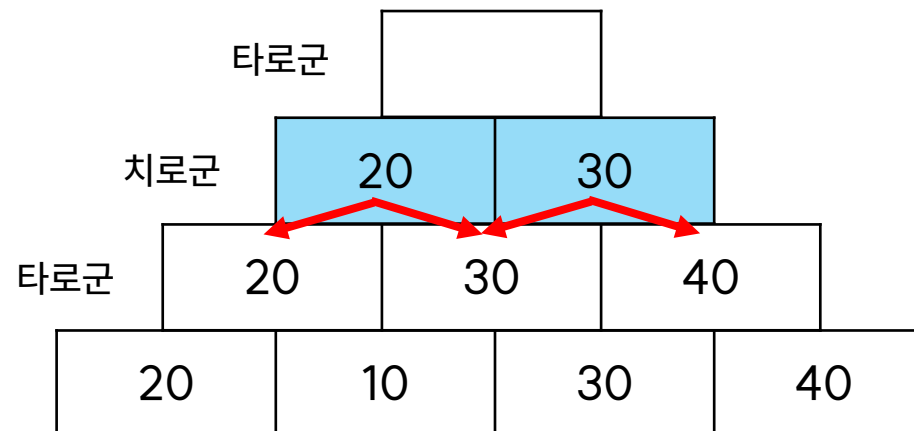
Mathematics

- 타로군은 최대화, 치로군은 최소화하길 원하기 때문에 각 행에 따라 최댓값을 위로 가져올지, 최솟값을 가져올지 정해주면서 피라미드를 채우면 됩니다.
- 타로군이 게임을 먼저 시작하므로, $1, 3, 5, \dots, 2k + 1 \leq N - 1$ 행은 아래 두 값중 최댓값, $2, 4, 6, \dots, 2k \leq N - 1$ 행은 아래 두 값중 최솟값을 넣어주면 됩니다.



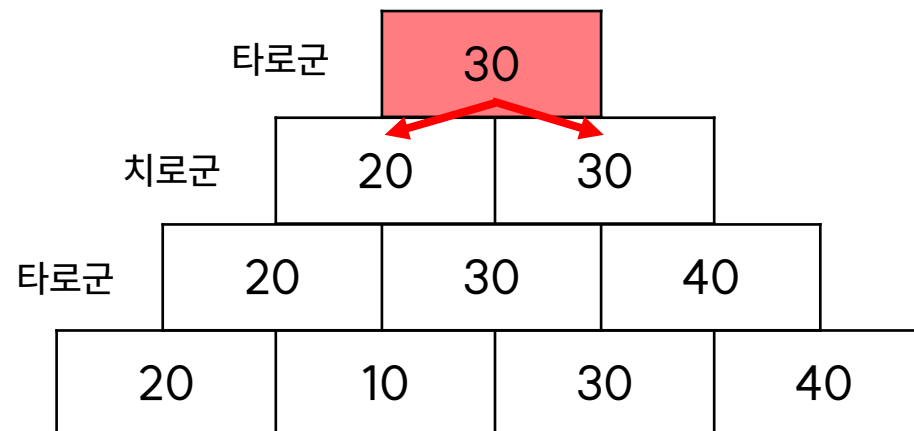
Mathematics

- 타로군은 최대화, 치로군은 최소화하길 원하기 때문에 각 행에 따라 최댓값을 위로 가져올지, 최솟값을 가져올지 정해주면서 피라미드를 채우면 됩니다.
- 타로군이 게임을 먼저 시작하므로, $1, 3, 5, \dots, 2k + 1 \leq N - 1$ 행은 아래 두 값중 최댓값, $2, 4, 6, \dots, 2k \leq N - 1$ 행은 아래 두 값중 최솟값을 넣어주면 됩니다.



Mathematics

- 타로군은 최대화, 치로군은 최소화하길 원하기 때문에 각 행에 따라 최댓값을 위로 가져올지, 최솟값을 가져올지 정해주면서 피라미드를 채우면 됩니다.
- 타로군이 게임을 먼저 시작하므로, $1, 3, 5, \dots, 2k + 1 \leq N - 1$ 행은 아래 두 값중 최댓값, $2, 4, 6, \dots, 2k \leq N - 1$ 행은 아래 두 값중 최솟값을 넣어주면 됩니다.



Mathematics

- 이를 코드로 구현하면 다음과 같습니다.

```
int dp[2005][2005];
int main() {
    fastio();
    int N;
    cin >> N;
    for(int i=1; i<=N; i++) {
        cin >> dp[N][i];
    }
    for(int i=N-1; i>=1; i--) {
        for(int j=1; j<=i; j++) {
            if(i%2==0) {
                dp[i][j] = min(dp[i+1][j], dp[i+1][j+1]);
            } else {
                dp[i][j] = max(dp[i+1][j], dp[i+1][j+1]);
            }
        }
    }
    cout << dp[1][1];
}
```

Mathematics Basic - 2

끝.

다음주에는 Thinking Techniques - 1로 찾아뵙겠습니다.