

2024년 8월 31일, 14시 ~ 17시

Code Master 2024

Official Solutions

송도고등학교 주최
전국 중고등학생 대상 알고리즘 대회

Thanks to

출제

차형준 *cherub8128*
송도고등학교 교사

김도훈 *dohoon*
송도고등학교 졸업생

백진서 *jinseo*
송도고등학교 재학생

임준세 *junse2*
송도고등학교 재학생

김창연 *weekend*
송도고등학교 재학생

검수

노준혁 *gggkik*
한양대학교

박신욱 *bnb2011*
DGIST

가지 *kiwiyou*
연세대학교

이병민 *ibm2006*
군대

김형진 *plast*
연세대학교

한정환 *pizzaroot*
세종대학교

이동현 *lisifu*
서울대학교

白崎杏子 *cologne*
開拓団訓練所

이주형 *iccodly*
경기과학고등학교

sjhi00
서울대학교

sait2000
—

장래오 *leo020630*
포항공과대학교

후원

STARTLINK

SOLVED.AC

해설에 앞서

문제를 풀어주신 여러분,
정말 수고하셨습니다!

이제 백준에 문제가 공개되었으니 풀어보실 수 있습니다.

각 문제에 사용되는 알고리즘들에 대한 자료를 적어 놓았으니
공부해보시면 좋을 것 같습니다.

궁금한 점은 dohoon@kakao.com으로 질문 주시면
답변해드릴 수도 있습니다.

감사합니다!

주의

모범 코드들은 **참고**만 해주세요.

공부의 목적이라고 해도 코드를 베끼는 것은 문제가 됩니다.
직접 작성하신 코드만 제출해주세요.

배껴서 제출하신 코드가 **치팅**으로 판단되면
해당 문제에 제출한 **모든 소스가 삭제**됩니다.

또한 3회 이상 치팅 적발 시 해당 계정은
약 70년간 제출 금지 처분을 받습니다.

자세한 내용은 https://help.acmicpc.net/rule에서 확인하세요.

문제 목록

#	문제	난이도
A	코드마스터 2024	**Easy**
B	찬물 샤워	**Easy**
C	광선 다각형	**Easy**
D	반려동물 준세	**Medium**
E	코드마스터, 슬라이딩 퍼즐 마스터, 보드게임 마스터	**Medium**
F1	연강은 힘들어(Easy)	**Medium**
F2	연강은 힘들어(Hard)	**Hard**
G	도시농부	**Challenging**
H	송도고 레일 정비 사업	**Challenging**
I1	물탱크 알바(Easy)	**Hard**
I2	물탱크 알바(Hard)	**Challenging**
Σ	**11문제**	

A. 코드마스터 2024

난이도 : **Easy** 태그 : ***math*** 출제 : **김도훈** *dohoon* 세팅 : **임준세** *junse2*

간단한 사칙연산

k대의 컴퓨터에 m개의 에디터를 설치하기에

그 이후에는 노트북 한대에 별도로 m개의 에디터를 한번만 설치하면 됩니다.

$(k + 1) \times m$

코멘트

✓ 가장 마지막에 정한 문제
✓ 참가해주신 모든 분들 감사드립니다!

공부하면 좋은 자료
✓ https://solved.ac/problems/sprout

모범 코드

✓ C++ (**김도훈** *dohoon*)

```cpp
#include <bits/stdc++.h>
using namespace std;
int main() {
    int n, m, k;
    cin >> n >> m >> k;
    cout << m * (k + 1);
}
```

✓ Python (**김도훈** *dohoon*)

```python
n, m, k = map(int, input().split())
print(m*(k+1))
```

B. 찬물 샤워

난이도 : **Easy** 태그 : ***math, for*** 출제 : **김도훈** *dohoon* 세팅 : **임준세** *junse2*

지문 독해

처음에는 어려워 보일 수 있으나 끈기를 가지고 읽으신다면 i초의 물의 온도는 모두 $d_i + k$로 같다는 사실을 알 수 있습니다. 위 사실을 확인하고 구하고자 하는 불쾌함 지수를 구하는 식 $\left|T_i - k\right|$ 을 보면 문제의 답은

$$\sum_{i=1}^{n} \left|d_i\right|$$

로 깔끔하게 떨어집니다.

이렇게 하지 않고 직접 문제 내용에 맞게 시뮬레이션을 거쳐도 해결할 수 있습니다.

코멘트

✓ 코딩보다 독해가 어렵다
✓ 샤워를 너무 자주해도 좋지 않습니다.

참고
✓ 2024 9모 국어 (아님)

모범 코드

✓ C++ (**김도훈** *dohoon*)

```cpp
#include<bits/stdc++.h>
using namespace std;
int main() {
    int n, k, x, s=0;
    cin >> n >> k >> x;
    for (int i = 1; i ≤ n; ++i) {
        int a; cin >> a;
        s += a;
    }
    cout << s;
}
```

✓ Python (**한정환** *pizzaroot*)

```python
input()
print(sum(list(map(int, input().split()))))
```

C. 광선 다각형 만들기

난이도 : **Easy** 태그 : ***math, geometry*** 출제 : **김창연** *weekend* 세팅 : **김창연** *weekend*

다각형의 내각의 합

n각형의 내각의 합은 $180° \times (n - 2)$라는 사실은 많이 알려져있습니다.

주어진 각 n개의 입사각의 두 배는 볼록 다각형의 한 내각과 같습니다. 꼭짓점이 $n + 1$개인 볼록 다각형에서 n개의 내각을 알고 있다면 나머지 한 개의 각은 쉽게 구할 수 있겠죠

코멘트

✓ 지문을 이해하기 쉽게 정말 많이 갈아엎었습니다.
✓ 그럼에도 풀지 못한 모든 분들께 죄송합니다.

참고
✓ 초등 4학년 2학기 6단원 다각형 단원

모범 코드

✓ C++ (**김도훈** *dohoon*)

```cpp
#include <bits/stdc++.h>
using namespace std;

int main() {
    int n, s=0; cin >> n;
    for (int i = 1; i ≤ n; ++i) {
        int a; cin >> a;
        s += a;
    }
    cout << 180*(n-1) - 2*s;
}
```

✓ Python (**김창연** *weekend*)

```python
n=int(input())
angle_sum = 180 * (n-1)
l = list(map(int,input().split()))
for i in l:
    x=i
    angle = (x*2)
    angle_sum -= angle
print(angle_sum)
```

D. 반려동물 준세

난이도 : **Medium** 태그 : ***implementation*** 출제 : **임준세** *junse2* 세팅 : **임준세** *junse2*

준세는 무조건 잠을 잘 수 있다

버블 정렬의 아이디어를 이용합니다.

매 시행마다 배열의 맨 뒤에는 0이 차곡차곡 쌓입니다. 따라서 $n + 1$번의 시행 내로 배열은 모든 원소가 0인 상태에 도달하게 됩니다.

이유는, 다음과 같습니다.

- ✓ 한 번의 시행 내로 맨 마지막 원소가 0으로 바뀝니다.
- ✓ 0이 아닌 맨 마지막 원소 뒤에는 0들만이 존재합니다.
- ✓ 이 원소는 0보다 작았다면, 시행에 의해 그 뒤 원소의 개수로 바뀝니다. 이는 0 이상의 값입니다.
- ✓ 0 이상의 값은 뒤에 있는 모든 0보다 작거나 같으므로 한 번의 시행 내로 0으로 바뀝니다.

정리하면, 위 논리에 의해 맨 뒤에 연속한 0의 개수는 매 단계, 최소 하나씩 늘어납니다.

따라서 문제의 내용을 그대로 구현만 해도 $O(n^3)$의 시간복잡도에 문제를 해결할 수 있음이 보장됩니다.

코멘트

- ✓ 실제로 준세는 10번 정도 문제를 반려 당했습니다.
- ✓ chatGPT는 시간복잡도를 잘 모르는 채로 그냥 시뮬레이션을 짜줍니다. 아무튼 그걸 내도 맞힐 수 있도록 의도했습니다.

공부하면 좋은 자료

- ✓ 버블정렬 튜토리얼

모범 코드

✓ C++ (**김도훈** *dohoon*)

```cpp
#include <bits/stdc++.h>
#define rep(i,a,b) for (int i = (a); i ≤ (b); ++i)
using namespace std;

const int N = 203;
int n, a[N], b[N];

int main() {
    cin.tie(0)→sync_with_stdio(0);
    cin >> n;
    rep(i,1,n) cin >> a[i];

    rep(ans,1,n+1) {
        fill(b+1,b+n+1,0);
        rep(i,1,n) rep(j,i+1,n) {
            b[i] += a[i] < a[j];
        }
        if (equal(a+1,a+n+1,b+1)) cout << ans, exit(0);
        copy(b+1,b+n+1,a+1);
    }
    assert(0);
}
```

✓ Python (**이동현** *lisifu*)

```python
n = int(input())
a = list(map(int, input().split()))
ans = 1
while any(a):
    ans += 1
    a = [sum(a[j] > a[i] for j in range(i + 1, n)) for i in range(n)]
print(ans)
```

E. 코드마스터, 슬라이딩 퍼즐 마스터, 보드게임 마스터

난이도 : **Medium** 태그 : ***game theory*** 출제 : **백진서** *jinseo* 세팅 : **백진서** *jinseo*

게임을 이기는 방법

게임의 상태 그래프를 아시나요?

서로에게 공평하고, 랜덤 요소가 없으며, 게임의 상태를 두 플레이어 모두 공유하는 게임은 보통 상태 그래프를 그려서 분석할 수 있습니다. 그래프의 각 정점은 승리할 수 있는 상태, 혹은 패배할 수 밖에 없는 상태로 표현할 수 있습니다. 각각 Win, Lose라고 해봅시다.

- ✓ 현재 정점에서 갈 수 있는 모든 상태 정점이 Win이라면, 현재 정점은 Lose입니다. 왜냐하면 다음 상태 정점으로 이동하면 플레이어가 상대로 바뀌는데, 무슨 짓을 해도 상대 플레이어가 Win이라면 현재 플레이어는 Lose인 것이죠.
- ✓ 반대로, 현재 정점에서 갈 수 있는 Lose 상태 정점이 하나라도 존재하면, 현재 정점은 Win입니다. 왜냐하면 그 정점으로 이동함으로써 상대 플레이어는 Lose로 게임을 해야 하기 때문입니다.

따라서 게임의 기저 상태를 잘 결정한 뒤, 게임의 상태 그래프를 규칙에 따라 잘 채워나가다 보면 게임의 승패에 대한 패턴을 비교적 쉽게 찾을 수 있게 됩니다.

L	*W*	*W*	*W*
W	*L*	*W*	*W*
W	*W*	*L*	*W*
W	*W*	*W*	*L*

필승 규칙

말의 위치가 (x, x), 즉 주대각선 위에 올려져있다면 후공이 무조건 이기는 필승 전략이 존재합니다. 선공이 다른 아무곳으로 말을 옮겨도 후공은 그저 다시 주대각선 위로 다시 말을 옮겨주면 됩니다.

위의 성질을 이용한다면 말의 초기 위치가 주대각선 위가 아니라면 선공이 무조건 이길 수 있는 필승 전략이 존재합니다. 선공이 말을 주대각선 위로 옮긴다면 선공과 후공이 뒤바뀐 위의 경우와 같습니다.

스프라그 그런디

이 정리에 대해 공부하셨었다면 더욱 쉽게 규칙을 찾으실 수 있습니다. 그저 문제 상황을 그대로 그런디 수로 적어보면 되기 때문이죠. 게임이론에 흥미가 있으신 분이시면 한번 공부해보시길 추천드립니다.

코멘트

- ✓ hard 문제(말이 여러개 존재 가능한 버전)가 존재했으나 단순히 스프라그 그런디를 알고있는지에 가까워 대회에 넣지 않았습니다.
- ✓ 진서가 코드마스터고 도훈이가 슬라이딩 퍼즐 마스터인 이유...
  - https://boj.kr/songdo2023 — 코드마스터 2023
  - https://www.acmicpc.net/problem/27526 — 슬라이딩 퍼즐 마스터

공부하면 좋은 자료

- ✓ https://solved.ac/problems/tags/sprague_grundy

모범 코드

✓ C++ (**김도훈** *dohoon*)

```cpp
#include <bits/stdc++.h>

int main() {
    int t; scanf("%d", &t);
    while (t--) {
        int n, m; scanf("%d%d", &n, &m);
        if (n == m)
            puts("dohoon");
        else
            puts("jinseo"),
            printf("%d %d\n", std::min(n, m), std::min(n, m));
    }
}
```

✓ Python (**김도훈** *dohoon*)

```python
t = int(input())
for _ in range(t):
    r, c = map(int, input().split())
    if r == c:
        print('dohoon')
    else:
        print('jinseo')
        print(f'{min(r,c)} {min(r,c)}')
```

F1. 연강은 힘들어(Easy)

난이도 : **Medium** 태그 : ***dp*** 출제 : **차형준** *cherub8128* 세팅 : **백진서** *jinseo*

DP

먼저 상태를 정의해 봅시다.

- ✓ k번 연속하는 수업이 존재하는 경우와 존재하지 않는 경우를 나누어 줍시다.
- ✓ dp1[i] ≔ i교시 수업이 비어있을 때 i교시 수업까지 경우의 수 중에 k번 연속하는 수업 존재 X
- ✓ dp2[i] ≔ i교시 수업이 비어있을 때 i교시 수업까지 경우의 수 중에 k번 연속하는 수업 존재 O
- ✓ 초기 상태를 정의 dp1[0] = 1

dp[i]에 dp[$i - a$]의 값을 더한다는 의미는 무엇일까요?

i교시 수업과 $i - a$교시 수업이 비어있기에 그 사이 $i - 1$부터 $i - a + 1$교시 수업이 모두 있는 경우를 더한다고 생각할 수 있고 이를 통해 점화식을 세울 수 있습니다.

점화식

이제 점화식을 세워봅시다.

$$\text{dp1}[i] = \text{dp1}[i-1] + \text{dp1}[i-2] + \cdots + \text{dp1}[\max(i-k, 0)]$$

$$\text{dp2}[i] = \text{sum}\begin{cases} \text{dp2}[i-1] + \text{dp2}[i-2] + \cdots + \text{dp2}[\max(i-k, 0)] \\ \text{dp1}[i-k-1] \text{ if } i \geq k+1 \\ \text{dp2}[i-k-1] \text{ if } i \geq k+1 \end{cases}$$

- ✓ 이를 나이브하게 $O(n^2)$으로 해결할 수 있습니다.

코멘트

- ✓ dp 초깃값을 잘 지정하고 수업 k번 연속해서 dp2로 넘어가는 과정을 주의

공부하면 좋은 자료

- ✓ https://www.acmicpc.net/problem/2560

모범 코드

✓ C++ (백진서 *jinseo*)

```cpp
#include <bits/stdc++.h>

using namespace std;
typedef long long ll;

const ll MOD = 1e9+7;
ll n, k;
ll dp[3030][2];

int main()
{
    ios::sync_with_stdio(0);
    cin.tie(0);

    cin >> n >> k;

    int a;
    for (int i=0; i<n; i++) cin >> a;

    dp[0][0] = 1;
    for (ll i=1; i≤n+1; i++){
        for (ll j=1; j≤min(i, k); j++){
            dp[i][0] = (dp[i][0] + dp[i-j][0]) % MOD;
            dp[i][1] = (dp[i][1] + dp[i-j][1]) % MOD;
        }
        if (i ≥ k+1){
            dp[i][1] = (dp[i][1] + dp[i-k-1][0]) % MOD;
            dp[i][1] = (dp[i][1] + dp[i-k-1][1]) % MOD;
        }
    }

    cout << dp[n+1][1];

    return 0;
}
```

F2. 연강은 힘들어(Hard)

난이도 : **Hard** 태그 : ***dp*** 출제 : **차형준** *cherub8128* 세팅 : **백진서** *jinseo*

누적합 최적화

$$dp1[i] = dp1[i-1] + dp1[i-2] + \cdots + dp1[\max(i-k, 0)]$$

$$dp2[i] = sum\begin{cases} dp2[i-1] + dp2[i-2] + \cdots + dp2[\max(i-k, 0)] \\ dp1[i-k-1] \text{ if } i \geq k+1 \\ dp2[i-k-1] \text{ if } i \geq k+1 \end{cases}$$

다시 한 번 점화식을 보겠습니다.

$dp1[i-1]$부터 $dp1[\min(i-k, 0)]$을 그냥 더할 뿐이라면 누적합을 이용할 수 있을 것 같습니다.

다시 점화식

- ✓ 초기 상태를 정의 $dp1[0] = 1, sum1[0] = 1$
- ✓ 식의 편의를 위해서 i번째 교시면 $i+1$로 뒤로 미뤘습니다.

$$sum1[i] = sum1[i-1] + dp1[i]$$

$$sum2[i] = sum2[i-1] + dp2[i]$$

$$dp1[i] = sum1[i-1] - sum1[\max(i-k-2, 0)]$$

$$dp2[i] = sum\begin{cases} sum2[i-1] - sum2[\max(i-k-1, 0)] \\ dp1[\max(i-k-1, 0)] \end{cases}$$

이미 지정된 수업

이 조건은 크게 영향을 미치지 않습니다. 그저 if문을 통해 지정되어 있다면 건너뛰면 됩니다.

코멘트

- ✓ dp문제에서 자주 볼 수 있는 최적화 방법
- ✓ deque를 통한 풀이 방법도 존재합니다.
- ✓ (k 이하로 사용) - ($k-1$ 이하로 사용) 방식으로 구할 수도 있습니다.

공부하면 좋은 자료

- ✓ https://www.acmicpc.net/problem/2560

모범 코드

✓ C++ (김도훈 *dohoon*)

```cpp
#include <bits/stdc++.h>
using namespace std;

const int N = 1e6+5, K = 1e6+5, MOD = 1e9+7;
int n, k;
long long dp[N][2], pfx[N][2];
int a[N];

int sum(int l, int r, int k) {
    return ( pfx[r][k] - (l==0 ? 0 : pfx[l-1][k]) ) % MOD;
}

int main() {
    cin.tie(0)->sync_with_stdio(0);
    cin >> n >> k;

    for (int i = 2; i <= n+1; ++i) cin >> a[i];
    dp[0][0] = pfx[0][0] = 1;

    for (int i = 1; i <= n+1; ++i) {
        dp[i][0] += sum(i-min(i,k), i-1, 0);
        dp[i][1] += sum(i-min(i,k), i-1, 1);

        if (i >= k+1 && !a[i-k])
            dp[i][1] += dp[i-k-1][0] + dp[i-k-1][1];

        dp[i][0] %= MOD;
        dp[i][1] %= MOD;

        pfx[i][0] = ( pfx[i-1][0] + (a[i+1]?0:dp[i][0]) ) % MOD;
        pfx[i][1] = ( pfx[i-1][1] + (a[i+1]?0:dp[i][1]) ) % MOD;
    }
    cout << (dp[n+1][1]%MOD + MOD) % MOD;
}
```

✓ Python (**이동현** *lisifu*)

```python
import sys, os

input = sys.stdin.read
data = input().split()

n = int(data[0])
k = int(data[1])
a = [0] + list(map(int, data[2:2+n]))
mod = 10**9 + 7

def sol(len):
    dp = [1] + [0] * n
    s, j = 0, 0
    for i in range(1, n + 1):
        if not a[i]:
            s = (s + dp[i - 1]) % mod
        if i - j - 1 > len:
            if not a[j + 1]:
                s = (s - dp[j]) % mod
            j += 1
        dp[i] = (s + (i ≤ len)) % mod
    return dp[n]

print((sol(k) - sol(k - 1)) % mod)
```

G. 도시농부

난이도 : **Challenging** 태그 : ***greedy*** 출제 : **백진서** *jinseo* 세팅 : **백진서** *jinseo*

그리디하게

먼저 할 수 있는 관찰들을 생각해 봅시다.

- ✓ A 영양제를 사용한다면 하나의 수확 가능한 상추와 아닌 상추 두 개를 조합해 수확 가능 상추 2개를 만들 수 있습니다.
- ✓ 위 관찰과 C 영양제를 조합한다면 기준 하한이라고 생각 가능한 C 사용 후 A만 계속 사용을 관찰할 수 있습니다.
- ✓ 영양제 B를 사용할 때 3개의 상추가 모두 $\frac{m}{2}$ 이상이라면 3개의 수확 가능 상추가 만들어집니다.

수확 가능한 상추가 초기에 존재한다면 B로 $\frac{m}{2}$ 이상의 상추를 최대로 많이 묶으면 되니 중요한 케이스는 초기에 수확 가능한 상추가 존재하지 않을 때입니다. 영양제 B의 사용에 따라 세부 케이스가 나뉠 듯합니다.

위와 연결되어 $\frac{m}{2}$ 이상이며 m보다 작은 상추가 얼마나 생기는지도 중요하겠죠.

결론

모든 사실들을 조합해 보면 각 시행에서 수확 가능한 상추를 많이 만드는 것을 우선하되 $\frac{m}{2}$ 이상 상추를 만드는 것을 2순위로 생각하는 그리디로 문제가 풀릴 것 같습니다. 실제로 케이스워크를 해보면 그럴 것 같다는 확신이 듭니다.

풀이

수확 가능한 상추가 있는 경우

1. 수확 가능한 상추가 존재할 경우 $\frac{m}{2}$ 이상인 상추들을 최대한 3개씩 묶어 B를 쓴다. 그 다음에는 A 사용해서 2개씩

수확 가능한 상추가 없는 경우

2-1. 영양제 B로 상추 3개를 수확 가능한 상태로 만드는 경우, B 쓰고 1로 이동

2-2. 영양제 B로 상추 2개를 수확 가능한 상태로 만드는 경우, B 쓰고 1로 이동

2-3. 영양제 B로 상추 1개를 수확 가능하게 만들며 나머지 1개이상을 $\frac{m}{2}$ 이상 상추로 만드는 경우, B 쓰고 1로 이동

2-4. 영양제 C 쓰고 1로 이동

증명

케이스워크가 주이므로 증명은 직접 해보시길 바랍니다. 각 연산에 '효율'을 부여하는 것도 좋은 방법입니다.

코멘트

✓ 위의 풀이를 그대로 시뮬레이션하는 코드를 짜지 않고 식으로도 해결 가능합니다
✓ 밑에 코드를 참고

모범 코드

✓ C++ (백진서 *jinseo*)

```cpp
#include <bits/stdc++.h>

using namespace std;
const int MAX = 1e5+10;

int arr[MAX];

int n, m, ans=0;

void case1(){
    sort(arr, arr+n);

    for (int i=0; i<n-2; i++){
        if (arr[i] ≥ (m+1)/2 && arr[i+2] < m){
            arr[i] = m;
            arr[i+1] = m;
            arr[i+2] = m;
            ans++;
            i+=2;
        }
    }

    int cnt=0;
    for (int i=0; i<n; i++){
        if (arr[i] < m) cnt++;
    }
    ans += (cnt+1)/2;
}

void case2(){
    sort(arr, arr+n);

    if (arr[n-3]+arr[n-3] ≥ m) {
        arr[n-1] += arr[n-3];
        arr[n-2] += arr[n-3];
        arr[n-3] += arr[n-3];
    }
    else if (arr[n-2]+arr[n-3] ≥ m){
        arr[n-1] += arr[n-3];
        arr[n-2] += arr[n-3];
        arr[n-3] += arr[n-3];
    }
    else if (arr[n-1] ≥ (m+1)/2){
        int i, check = 0;
        for (i=0; i<n-1; i++){
```

```
            if (arr[i]+arr[i] ≥ (m+1)/2 && arr[i+1] < (m+1)/2 && arr[n-1]+arr[i] ≥ m){
                arr[i+1] += arr[i];
                arr[n-1] += arr[i];
                arr[i] += arr[i];
                break;
            }
        }

        if (i == n-1) arr[n-1] = m;
    }
    else arr[n-1] = m;

    ans++;

    case1();
}

int main()
{
    ios::sync_with_stdio(0);
    cin.tie(0);

    cin >> n >> m;

    int check=0;
    for (int i=0; i<n; i++){
        cin >> arr[i];

        if (arr[i] == m) check=1;
    }

    if (check == 0) case2();
    else case1();

    cout << ans;

    return 0;
}
```

✓ Python (**白崎杏子** *cologne*)

```python
def main():
    n, m = map(int, input().split())
    *s, = map(int, input().split())
    S = sorted([v for v in s if 2*v < m], reverse=True)
    M = sorted([v for v in s if m ≤ 2*v < 2*m], reverse=True)
    L = [v for v in s if v == m]

    def score(x):
        if x ≥ m:
            return 0
        elif 2*x ≥ m:
            return 2
        else:
            return 3

    ans = n
    if len(L) ≥ 1 or len(M) ≥ 3:
        pts = 3*len(S) + 2*len(M)
        ans = min(ans, (pts+5)//6)

    if len(M) ≥ 2 and len(S) ≥ 1:
        pts = 3 * (len(S) - 1) + 2 * (len(M) - 2)
        M1, M2, S1 = M[0], M[1], S[0]
        pts += score(M1+S1) + score(M2+S1) + score(S1+S1)
        ans = min(ans, 1+(pts+5)//6)

    if len(M) ≥ 1 and len(S) ≥ 2:
        pts = 3 * (len(S) - 2) + 2 * (len(M) - 1)
        M1, S1, S2 = M[0], S[0], S[1]
        pts += score(M1+S2) + score(S1+S2) + score(S2+S2)
        ans = min(ans, 1+(pts+5)//6)

    if len(S) ≥ 1:
        pts = 3 * (len(S) - 1) + 2 * len(M)
        ans = min(ans, 1+(pts+5)//6)

    print(ans)


if __name__ == "__main__":
    main()
```

H. 송도고 레일 정비 사업

난이도 : **Challenging** 태그 : ***memoization, sparse table*** 출제 : **백진서** *jinseo* 세팅 : **백진서** *jinseo*

레일 따라가기

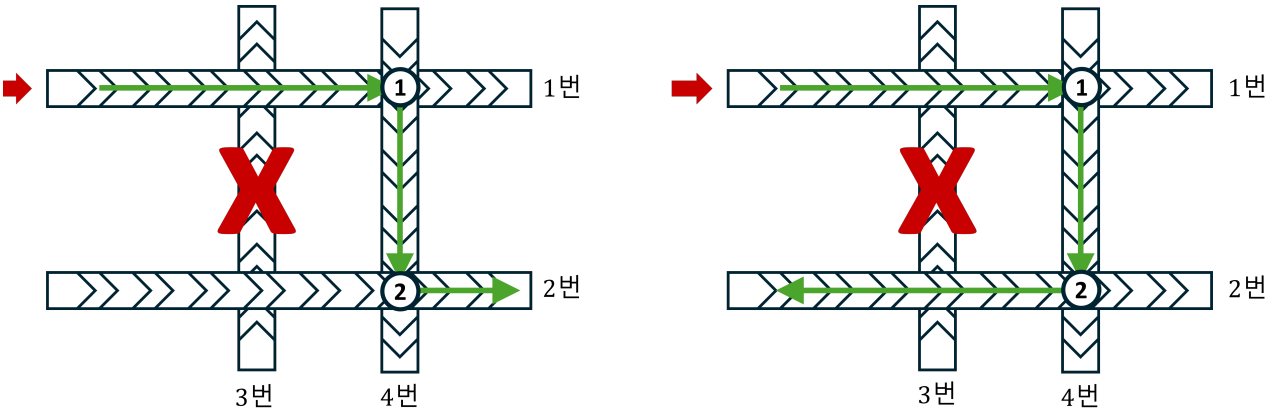
시간 제한은 무시하고 레일을 따라가는 동작을 생각해봅시다.

희소 배열을 이용한다면 $\log n$에 다음 레일을 찾을 수 있습니다. 동서, 남북, 이동 방향, 즉 4개의 희소 배열을 만듭니다. 동서 레일이고 움직이는 방향이 동쪽임을 가정하면 현재 레일에서 2^k만큼 동쪽에 있는 레일로 넘어갈 때 그 사이에 있는 레일 중 최소의 p_i 우선순위를 가지도록 희소 배열을 구성하면 범위를 줄여가며 가장 서쪽에 있는 현재 레일보다 p_i가 작은 레일을 $\log n$에 찾을 수 있습니다.

유의미한 교점의 개수

유의미한 교점의 상한이 충분히 작음을 증명해봅시다.

레일이 두 번 꺾이는 경우의 그림입니다.



각각의 레일에서 처음으로 꺾이는 연속된 두 개의 교차점을 그 레일이 가지는 교차점이라고 정의합시다.

첫 번째 케이스는 두 개보다 많은 교차점이 생겨도 2번 레일이 가지는 교차점입니다. 2번 레일의 교차점이 아니라면 2번 레일이 더 왼쪽에서 교차하는 점이 있어야 하지만 그런 3번 레일을 만들 경우 1번 레일에서 다른 경로로 이동해 모순이 생깁니다.

두 번째 케이스의 경우에는 더 이상 교차 레일을 만들 수 없습니다. 만든다면 모순이 생깁니다. 2번 레일보다 p_i가 낮아야 더 꺾이게 할 수 있지만 그런 3번 레일을 만든다면 1번 레일에서 다른 경로로 이동합니다.

레일 하나당 최대 2개의 유의미한 교점을 갖습니다. 결국 꺾이는 교차점의 상한은 $(n + m) \times 2$입니다.

그래서

위의 결과로 대부분의 경우 경로가 겹친다는 사실을 알 수 있습니다. 그래서 메모이제이션으로 한 번 지나간 경로는 생략합니다.

✓ $O((n + m) \log(n + m))$에 해결할 수 있습니다.

코멘트

✓ 유니온 파인드, 세그먼트 트리와 같은 자료구조로도 풀립니다.

공부하면 좋은 자료

✓ https://solved.ac/problems/tags/sparse_table

모범 코드

✓ C++ (**백진서** *jinseo*)

```cpp
#include <bits/stdc++.h>

using namespace std;

int n[2];
int a[100010][2][2];
pair<int, int> b[200020];
int sp[100010][17][2][2];
int check[200020];
int output[200020];

int solve(int x, int now, int p, int d, int fst){
    if (d==2 && fst==a[now][x][1]) return check[now*2+x];

    int tmp = p;
    for (int i=16; i≥0; i--){
        int nt = tmp + (1<<i)*(a[now][x][1]*2-1);
        if (sp[tmp][i][x^1][a[now][x][1]] > a[now][x][0])
            tmp = nt;

        if (tmp < 0 || tmp >= n[x^1]) break;
    }
    if (tmp ≥ 0 && tmp < n[x^1]) return solve(x^1, tmp, now, d+1, fst);
    else return now+n[0]*x+1;
}

int main()
{
    ios::sync_with_stdio(0);
```

```
    cin.tie(0);

    cin >> n[0] >> n[1];

    for (int i=0; i<2; i++){
        for (int j=0; j<n[i]; j++){
            cin >> a[j][i][0] >> a[j][i][1];
            a[j][i][1]--;

            b[i*n[0]+j] = {a[j][i][0], i*n[0]+j};
        }
    }

    for (int i=0; i<2; i++){
        for (int j=0; j<n[i]; j++){
            sp[j][0][i][0] = sp[j][0][i][1] = a[j][i][0];
        }
        for (int j=1; j<17; j++){
            for (int k=0; k<n[i]; k++){
                sp[k][j][i][0] = sp[k][j-1][i][0];
                sp[k][j][i][1] = sp[k][j-1][i][1];
                if (k-(1<<j-1) ≥ 0)
                    sp[k][j][i][0] = min(sp[k][j][i][0], sp[k-(1<<j-1)][j-1][i][0]);
                if (k+(1<<j-1) < n[i])
                    sp[k][j][i][1] = min(sp[k][j][i][1], sp[k+(1<<j-1)][j-1][i][1]);
            }
        }
    }

    sort(b, b+n[0]+n[1]);

    for (int i=0; i<n[0]+n[1]; i++){
        int x = 0, y;
        if (b[i].second ≥ n[0]) x = 1;
        y = b[i].second-n[0]*x;

        output[b[i].second] = check[y*2+x] = solve(x, y, a[y][x][1] ? 0:n[x^1]-1, 0, a[y]
[x][1]);
    }

    for (int i=0; i<n[0]+n[1]; i++)
        cout << output[i] << "\n";


    return 0;
}
```

I1. 물탱크 알바(Easy)

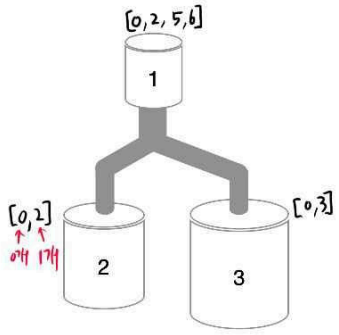
난이도 : **Hard** 태그 : ***dfs, tree*** 출제 : **김창연** *weekend* 세팅 : **김창연** *weekend*

넘칠까 안 넘칠까

이 문제의 포인트는 물이 넣기 시작한 물탱크가 가득 찬다고 멈추지 않고, 부모 물탱크로 올라간다는 점입니다. 따라서 *A* 물탱크에서 물이 넘치지 않는다 할 때, A와 A의 자식 물탱크만 체크를 해주면 되는 것이죠.

물이 들어온다면?

따라서 우리는 물탱크에 얼마나 많은 물이 들어와야 자신과 하위 물탱크가 *x*개 차는지 체크해주면 됩니다.



(예시)

이런 식으로 각 물탱크마다 리스트를 만들어 체크를 해주고, 주어진 물의 양 *m*을 넘어가는 경우에만 체크를 하여 max 값을 출력하면 됩니다.

코멘트

- ✓ '물탱크 문제'로 유명한 Which Tank Will Be Full First? 에서 영감을 얻었습니다.
- ✓ 혹시라도 자세한 풀이를 원하시는 분은 https://weekend07.tistory.com/73
- ✓ 실제로 오프라인 대회 다음 날, 잔나비 콘서트를 갔다 왔습니다🤘.
- ✓ 도훈: 흘러넘치는 dfs를 직접 구현해서 풀 수도 있고, 절반씩 나뉜다는 관찰을 잘 이용해서 시뮬레이션을 잘 돌릴 수도 있다고 합니다. 화이팅!

모범 코드

✓ C++ (**김도훈** *dohoon*)

```cpp
#include <bits/stdc++.h>
using namespace std;

const int N = 1003;
int n, m, sub1[N], sub2[N], c[N], ans;
vector<int> adj[N], v[N];

void dfs0(int s) {
    sub1[s] = 1, sub2[s] = c[s];
    for (auto u : adj[s]) dfs0(u), sub1[s] += sub1[u], sub2[s] += sub2[u];
}
int get(int x, int y) {
    return upper_bound(begin(v[x]), end(v[x]), y) - begin(v[x]);
}
void solve(int s) {
    for (auto u : adj[s]) solve(u);
    if (size(adj[s]) == 1) swap(v[s], v[adj[s][0]]);
    if (size(adj[s]) == 2) {
        int u1 = adj[s][0], u2 = adj[s][1];
        if (sub2[u1] ≤ m) ans = max(ans, sub1[u1] + get(u2, m-sub2[u1]));
        if (sub2[u2] ≤ m) ans = max(ans, sub1[u2] + get(u1, m-sub2[u2]));
        if (sub2[u1] > sub2[u2]) swap(u1,u2);

        for (int i=0, j=0; j < size(v[u2]);) {
            if (i < size(v[u1])) {
                if (v[u1][i] ≤ v[u2][j])
                    v[s].push_back(v[u1][i++]*2);
                else
                    v[s].push_back(v[u2][j++]*2);
            }
            else v[s].push_back(sub2[u1]*2 + v[u2][j++]-sub2[u1]);
        }
    }
    v[s].push_back(sub2[s]);
    ans = max(ans, get(s, m));
}

int main() {
    cin.tie(0)->sync_with_stdio(0);
    cin >> n >> m;
    for (int i = 1; i ≤ n; ++i) cin >> c[i];
    for (int i = 2; i ≤ n; ++i) { int p; cin >> p; adj[p].push_back(i); }
    dfs0(1), solve(1);
    cout << ans;
}
```

✓ Python (**김창연** *weekend*)

```python
import sys
input = sys.stdin.readline
from bisect import bisect_right
sys.setrecursionlimit(10**8)
n,m = map(int,input().split())
a = list(map(int,input().split()))
parent = list(map(int,input().split()))
p = [[] for _ in range(n+1)]
for i in range(0,n-1):p[parent[i]-1].append(i+1)
root = 0
answer=0
tank = [[] for _ in range(n+1)]
def dp(x):
    global answer
    if(len(p[x])==0):#리프노드
        tank[x].append(0)
        tank[x].append(a[x])


    elif(len(p[x])==1):#자식노드 하나
        y1 = p[x][0]
        dp(y1)
        for i in tank[y1]:tank[x].append(i)
        tank[x].append(tank[x][-1]+a[x])


    elif(len(p[x])==2):#자식노드 둘
        y1,y2 = p[x][0],p[x][1]
        dp(y1)
        dp(y2)
        tank[x].append(0)
        i,j=1,1
        while(i<len(tank[y1]) and j<len(tank[y2])):
            if(tank[y1][i]==tank[y2][j]):
                tank[x].append(tank[y2][j]*2)
                tank[x].append(tank[y1][i]*2)
                i+=1
                j+=1
            elif(tank[y1][i]>tank[y2][j]):
                tank[x].append(tank[y2][j]*2)
                j+=1
            elif(tank[y1][i]<tank[y2][j]):
                tank[x].append(tank[y1][i]*2)
                i+=1
        if(j<len(tank[y2])):
            tank[x].append(tank[x][-1] // 2 + tank[y2][j])
            for k in range(j,len(tank[y2])-1):
```

```
                tank[x].append(tank[x][-1]+ (tank[y2][k+1]-tank[y2][k]))
        elif(i<len(tank[y1])):
            tank[x].append(tank[x][-1] // 2 + tank[y1][i])
            for k in range(i,len(tank[y1])-1):
                tank[x].append(tank[x][-1]+ (tank[y1][k+1]-tank[y1][k]))
        tank[x].append(tank[x][-1]+a[x])


    if(tank[x][-1]≥m or x==root):#m을 넘어가거나 root 노드인경우
        pb=bisect_right(tank[x],m)-1
        b1,b2=0,0
        if(len(p[x])≥1):#자식노드 1개 이상
            b1 = bisect_right(tank[y1],m)-1
        if(len(p[x])≥2):#자식 노드 2개
            b2 = bisect_right(tank[y2],m)-1
            if(tank[y2][-1]<m):b2 += (bisect_right(tank[y1],m-tank[y2][b2]) - 1)
            if(tank[y1][-1]<m):b1 += (bisect_right(tank[y2],m-tank[y1][b1]) - 1)
        b = max([pb,b1,b2])
        if(b > answer):answer = b
dp(root)
print(answer)
```

I2. 물탱크 알바(Hard)

난이도 : **Challenging** 태그 : ***slope trick, small to large*** 출제 : **김창연** *weekend* 세팅 : **김도훈** *dohoon*

접근

일단 문제 상황에 따라, 정점에 들어온 물은 세 가지 방법으로 흘려줄 수 있습니다.

- ✓ 그냥 현재 정점에 흘림. 반씩 자식 정점으로 흘러들어감.
- ✓ 왼쪽 정점에 우선적으로 흘림. 이는 왼쪽 정점에 물을 투입하다가 오른쪽 정점으로 흘러넘치는 상황과 동치임.
- ✓ 오른쪽 정점에 우선적으로 흘림. 이는 오른쪽 정점에 물을 투입하다가 왼쪽 정점으로 흘러넘치는 상황과 동치임.

그럼 각 정점에서 일종의 계단 모양 함수를 관리한다는 느낌으로 접근합시다.

$f_u(x)$ ≔ 각 정점 u에 물을 x만큼 부었을 때 꽉 차게 되는 정점의 개수

계단 모양 함수의 꼭짓점들은, 서브트리 속 정점의 개수 이하로 구성될 것입니다.

small to large

몇 가지 case work를 해보면, 필요한 연산은,

- ✓ $f_u(x)$를 가져오는 것
- ✓ 그래프의 앞쪽 일부에 $+\delta$를 취하는 것
- ✓ 그래프의 뒷쪽 일부에 $\times 2$를 취하는 것

정도로 좁혀집니다.

여기서, **서브트리 용량 총합**을 기준으로 small to large를 적용하면, $O\left(n \log \sum c_i\right)$에 해결 가능하다는 것을 알 수 있습니다.

코멘트

- ✓ 이 외에도, 물이 절반씩 흘려내려간다는 성질에 의해 특정 시뮬레이션 횟수가 적다는 관찰을 조합하여 lazy propagation을 지원하는 persistent segtree를 이용하여 문제를 해결할 수도 있습니다. 시간복잡도는 동일하게 $O\left(n \log \sum c_i\right)$입니다.
- ✓ 난이도가 높고, 구현량이 많은 문제인 만큼, 코드를 올리지는 않겠습니다. 감사합니다.

풀어보면 좋은 문제

- ✓ https://www.acmicpc.net/problem/21795