

2023년 6월 11일, 9시 ~ 13시

Code Master 2023

Official Solutions

송도고등학교 주최
인천지역 중고등학생 대상 알고리즘 대회

Thanks to

출제

차형준 *cherub8128*
송도고등학교 교사

박상진 *songdo_computer*
송도고등학교 교사

김도훈 *dohoon*
송도고등학교 재학생

백진서 *jinseo*
송도고등학교 재학생

김예준 *dpwns0827*
송도고등학교 졸업생

검수

신용명 *tlsdydaud1*
한양대학교

박진식 *pjshwa*
크리마

장래오 *leo020630*
포항공과대학교

문정후 *mjhmjh1104*
경기과학고등학교

박기윤 *parkky*
KAIST

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 2023	Easy
B	점심시간 레이스	Easy
C	정보 선생님의 야망	Medium
D1	배고파(Easy)	Medium
D2	배고파(Hard)	Medium
E1	수학 선생님의 고민(Easy)	Medium
E2	수학 선생님의 고민(Hard)	Medium
F	S리그	Hard
G	교실 불 끄기	Hard
H	조별 수업	Challenging
I	사람이 먼저 되라	Challenging
Σ	9문제	

A. 코드마스터 2023

난이도 : **Easy** 태그 : ***if/else, string*** 출제 : **김도훈** *dohoon* 세팅 : **김도훈** *dohoon*

조건문

`if` 문을 아시나요?

문자열

C++의 경우 C 스타일 `char` 배열이 아닌 `string`을 이용하면 편리하게 문자열을 다룰 수 있습니다!

python의 경우 C++보다 문자열을 다루는 것이 편리하다는 점도 기억해주세요!

코멘트

✓ '27889번: 특별한 학교 이름'에서 아이디어를 얻었어요.
✓ 풀어주신 모든 분들의 응원, 정말 감사합니다!

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

모범 코드

✓ C++ (**김도훈** *dohoon*)

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

int main() {
    string s; cin >> s;
    if (s == "2023") cout << "0611";
    if (s == "CODE") cout << "MASTER";
    if (s == "SONGDO") cout << "HIGHSCHOOL";
    if (s == "ALGORITHM") cout << "CONTEST";
}
```

✓ Python (**김도훈** *dohoon*)

```python
s = input()
if s == "CODE": print("MASTER")
if s == "2023": print("0611")
if s == "SONGDO": print("HIGHSCHOOL")
if s == "ALGORITHM": print("CONTEST")
```

B. 점심시간 레이스

난이도 : **Easy** 태그 : ***for-loop, arithmetics*** 출제 : **김예준** *dpwns0827* 세팅 : **백진서** *jinseo*

모두 오른쪽 계단으로

모두 오른쪽 계단으로 이동한 뒤 1층까지 그대로 내려갑니다. 이 두 길이를 더한 값이 제일 작은 반 중 번호가 제일 작은 반을 구하면 되겠습니다.

모범 코드

✓ C++ (**김도훈** *dohoon*)

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

const int N = 100'003, M = N, K = N;
int n, m, k;
int ans, cost = N+M;

int main() {
    cin.tie(0)->sync_with_stdio(0);
    cin >> n >> m >> k;
    for (int i = 1, f, d; i <= k; ++i) {
        cin >> f >> d;
        if (cost > m-d+f) {
            cost = m-d+f;
            ans = i;
        }
    }
    cout << ans;
}
```

✓ Python (**박기윤** *parkky*)

```python
n,m,k=map(int,input().split())
r=m=1e8
for i in range(1,1+k):
    f,d=map(int,input().split())
    if m>f-d:m=f-d;r=i
print(r)
```

C. 정보 선생님의 야망

난이도 : **Easy** 태그 : ***brute force*** 출제 : **차형준** *cherub8128* 세팅 : **백진서** *jinseo*

두 개의 요일

다섯 개의 요일 중 두 개의 요일의 쌍을 하나씩 다 시도한 뒤, 가장 많은 학생이 수강할 수 있는 요일 쌍을 고르면 됩니다.

모범 코드

✓ C++ (**김도훈** *dohoon*)

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

const int N = 1'000'003;
int n, a[N][6];
int ans, x, y;

int main() {
    scanf("%d", &n);
    for (int i = 1; i ≤ n; ++i)
        for (int j = 1; j ≤ 5; ++j)
            scanf("%d", &a[i][j]);
    for (int i = 1; i ≤ 5; ++i) for (int j = i+1; j ≤ 5; ++j) {
        int cnt = 0;
        for (int k = 1; k ≤ n; ++k) if (a[k][i] & a[k][j]) cnt++;
        if (ans ≤ cnt) x = i, y = j, ans = cnt;
    }
    printf("%d\n", ans);
    for (int i = 1; i ≤ 5; ++i)
        printf("%d ", i == x or i == y);
}
```

✓ Python (**박기윤** *parkky*)

```python
import sys
input=sys.stdin.readline
n=int(input())
D=[0]*10
for i in range(n):
    *L,=map(int,input().split())
    k=0
    for j in range(5):
        for i in range(j):
            D[k]+=L[i]&L[j]
            k+=1
x=0;y=1;z=D[0]
k=0
for j in range(5):
    for i in range(j):
        if D[k]≥z:x=i;y=j;z=D[k]
        k+=1
print(z)
L=[0]*5
L[x]=L[y]=1
print(*L)
```

D1. 배고파(Easy)

난이도 : **Easy** 태그 : ***math, bit*** 출제 : **김도훈** *dohoon* 세팅 : **김도훈** *dohoon*

이진수로 접근

$2^x + 2^y = m$인데, x, y, m은 양의 정수입니다.

✓ 그럼 x와 y의 범위는 $\log_2 m$ 이하입니다.
✓ 따라서 x, y 각각에 대해 0부터 60 사이만 전부 대입하여 확인하면 빠르게 풀 수 있습니다.

시간 복잡도는 $O(n \log^2 m)$이 됩니다.

32bit 정수형 int

C/C++에서 `int`는 -2^{31}(약 -2×10^9)부터 $2^{31} - 1$ (약 2×10^9)까지만 표현할 수 있습니다. 따라서 -2^{63}(약 -9×10^{18})부터 $2^{63} - 1$ (약 9×10^{18})까지 표현할 수 있는 `long long`을 사용해야 합니다.

✓ 참고로 python은 수를 다룰 수 있는 정해진 범위가 없습니다.

비트마스크(bitmask)

이진수를 다룰 때에는 비트마스크와 비트연산자가 큰 도움이 됩니다.

공부하면 좋은 자료
✓ https://rebro.kr/63: 이해하기 쉽고 명료합니다
✓ https://mathsciforstudent.tistory.com/202: 제 블로그입니다 ㅎㅎ 어려운 내용까지 담았습니다

코멘트

✓ 시도해보지 않았지만 `pow(x,y)` 함수를 잘 사용하면 비트연산자 없이도 할 수 있을 거에요.
✓ 하지만 보통은 `pow(x,y)`에 실수 오차가 있고, 비트연산자가 더 다루기 쉽기 때문에 지양합니다.

모범 코드

✓ C++ (**장래오** *leo020630*)

```cpp
#include <bits/stdc++.h>
using namespace std;
#define fi first
#define se second
typedef long long ll;
typedef pair<ll,ll> pll;

int main()
{
    ios::sync_with_stdio(0); cin.tie(0);
    ll n,m,i,j;
    cin>>n;
    while(n--)
    {
        cin>>m;
        for(i=0;i<60;i++) for(j=i;j<60;j++)
            if((1LL<<i)+(1LL<<j)==m) cout<<i<<' '<<j<<'\n';
    }
    return 0;
}
```

✓ Python (**박기윤** *parkky*)

```python
D={}
for y in range(61):
    for x in range(y+1):
      D[(1<<x)+(1<<y)]=x,y
for i in range(int(input())):
    print(*D[int(input())])
```

D2. 배고파(Hard)

난이도 : **Medium** 태그 : ***math, bit, greedy*** 출제 : **김도훈** *dohoon* 세팅 : **김도훈** *dohoon*

배고파(Easy) 확장하기

$2^x + 2^y = m$에서 해가 존재하려면 m은:
✓ 1이 아닌 한 개의 비트가 켜져있는 수거나
✓ 두 개의 비트가 켜져있는 수여야 합니다.

이를 바탕으로 두 가지 풀이를 찾을 수 있습니다.

Solution 1. $O(n \log^2 m)$

배고파(Easy)에서 봤던 모든 (x, y)쌍을 갖고 m을 대체할 값들의 후보를 전부 보면서 답을 구하면 됩니다.

색다른 관찰

Solution 2. $O(n)$

본래 의도한 풀이입니다. Solution 1이 더 쉽지만 개인적으로 Codeforces 스타일의 그리디 유형이라고 생각해서 익혀둘 가치가 있다고 봅니다.

m의 켜져있는 비트의 개수에 따라 케이스를 나눠 풀어봅시다.

✓ 켜진 비트가 1개:
- 1일 경우 2로 대체합니다.
- 나머지는 $2^{k-1} + 2^{k-1} = 2^k$꼴이 되므로 m을 그대로 둡니다.

✓ 켜진 비트가 2개:
- 당연히 $2^x + 2^y = m$꼴이므로 m을 그대로 둡니다.

✓ 켜진 비트가 3개 이상:
- m을 **줄일 경우**, 앞에서부터 두 번째로 등장한 1까지 남기고 나머지를 날려줍니다.
  - 즉, $101101_{(2)}$라고 하면 $101000_{(2)}$로 바꿉니다.
  - 이보다 덜 줄일 경우 두 번째로 등장한 1까지가 아직 남아있기 때문에 이것이 최적입니다.
- m을 **늘릴 경우**, 앞에서부터 두 번째로 등장한 1 뒤를 전부 채워서 올려줍니다.
  - 즉, $101101_{(2)}$라고 하면 $110000_{(2)}$로 바꿉니다.
  - 이보다 덜 늘릴 경우 두 번째로 등장한 1까지가 아직 남아있기 때문에 이것이 최적입니다.

이 모든 걸 gcc 내장 builtin 함수를 이용해서 구현하면 각 $O(1)$의 시간 복잡도만에 문제를 풀 수 있습니다.

모범 코드

✓ C++ (**장래오** *leo020630* , Solution 1)

```cpp
#include <bits/stdc++.h>
using namespace std;
#define fi first
#define se second
typedef long long ll;
typedef pair<ll,ll> pll;

int main()
{
    ios::sync_with_stdio(0); cin.tie(0);
    ll n,m,i,j;
    cin>>n;
    while(n--)
    {
        cin>>m;
        pair<pll,pll> ans={{2e18,2e18},{2e18,2e18}};
        for(i=0;i<60;i++) for(j=i;j<60;j++)
            ans=min(ans,pair<pll,pll>(pll(abs((1LL<<i)+(1LL<<j)-m),(1LL<<i)
+(1LL<<j)),pll(i,j)));
        cout<<ans.se.fi<<' '<<ans.se.se<<'\n';
    }
    return 0;
}
```

✓ Python (**박기윤** *parkky* , Solution 1)

```python
for i in range(int(input())):
    m=int(input())
    a=b=0
    c=1<<61
    for y in range(60):
        for x in range(y+1):
            t=abs((1<<x)+(1<<y)-m)
            if t<c:c=t;a=x;b=y
    print(a,b)
```

✓ C++ (**김도훈** *dohoon* , Solution 2)

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

const int log2(ll x) {
    assert(x > 0);
    return 63-__builtin_clzll(x);
}

const int N = 10003;
int n; ll m;

int main() {
    scanf("%d", &n);
    for (int i = 1; i ≤ n; ++i) {
        scanf("%lld", &m);
        if (m == 1) puts("0 0");
        else if (__builtin_popcountll(m) == 1) {
            printf("%d %d\n", log2(m)-1, log2(m)-1);
        } else if (__builtin_popcountll(m) == 2) {
            printf("%d %d\n", __builtin_ctzll(m), log2(m));
        } else {
            int firbit = log2(m);
            int secbit = log2(m - (1LL<<log2(m)));
            ll m1 = (1LL<<firbit)+(1LL<<secbit); // decrease
            ll m2 = (1LL<<firbit)+(2LL<<secbit); // increase
            if (m-m1 ≤ m2-m) printf("%d %d\n", secbit, firbit);
            else printf("%d %d\n", secbit+1, firbit);
        }
    }
}
```

✓ Python (**박기윤** *parkky* , Solution 2)

```python
for i in range(int(input())):
    m=int(input())
    if m≤2:print(0,0);continue
    y=(m-1).bit_length()-1
    m-=1<<y
    x=m.bit_length()
    if m-(1<<x-1)≤(1<<x)-m:x-=1
    print(x,y)
```

E1. 수학 선생님의 고민(Easy)

난이도 : **Medium** 태그 : ***math, number theory*** 출제 : **차형준** *cherub8128* 세팅 : **김도훈** *dohoon*

그저 인수분해

$nx^2 + (n+1)x - (n+2) = (ax+b)(cx+d)$로 인수분해한다고 할 때,

✓ $a = 1, 2, \ldots, n$를 돌리면서 n이 a로 나누어떨어진다면 $c = \frac{n}{a}$로,
✓ $b = 1, 2, \ldots, n+2$를 돌리면서 $n+2$가 b로 나누어떨어진다면 $d = -\frac{n+2}{b}$로

결정해서 $n + 1 = ac + bd$인지 확인해주면 됩니다. 시간 복잡도는 $O(n^2)$이 됩니다.

모범 코드

✓ C++ (**김도훈** *dohoon*)

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

int n;
ll a, b, c, d;

int main() {
    scanf("%d", &n);
    for (a = 1; a <= n; ++a) if (c = n/a; a*c == n) {
        for (b = 1; b <= n+2; ++b) if (d = (n+2)/b; b*d == (n+2)) {
            if (a*d-b*c == n+1) printf("%lld %lld %lld %lld",a,-b,c,d), exit(0);
            else if (-a*d+b*c == n+1) printf("%lld %lld %lld %lld",a,b,c,-d), exit(0);
        }
    }
    puts("-1");
}
```

✓ Python (**김도훈** *dohoon*)

```python
n = int(input())
for a in range(1,n+1):
    for b in range(1,n+3):
        c = n//a; d = (n+2)//b
        if a*c == n and b*d == n+2:
            if a*d-b*c == n+1: print(a,-b,c,d); exit()
            elif -a*d+b*c == n+1: print(a,b,c,-d); exit()
print(-1)
```

E2. 수학 선생님의 고민(Hard)

난이도 : **Medium** 태그 : ***math, number theory*** 출제 : **차형준** *cherub8128* 세팅 : **김도훈** *dohoon*

정수 계수

Solution 1. $O(n)$

수학 선생님의 고민(Easy)를 확장하는 풀이입니다. 약수에 대한 간단한 성질을 살펴봅시다.

Lemma 1. 자연수 n에 대해 $ab = n$이라면 $\min\{a, b\} \le \sqrt{n}$이다.

Proof. 귀류법을 사용하여 $a, b > \sqrt{n}$인 상황을 가정합시다.

$a \times b > \sqrt{n} \times \sqrt{n} = n$인데 $n = a \times b \neq n$이므로 모순입니다.

따라서 Lemma 1은 참입니다. □

단순히
✓ $a = 1, 2, \ldots, n$이 아니라 $a = 1, 2, \ldots, \sqrt{n}$을 돌리고,
✓ $b = 1, 2, \ldots, n + 2$가 아니라 $b = 1, 2, \ldots, \sqrt{n + 2}$를 돌리게 하면

시간 복잡도를 $O(n)$으로 줄일 수 있습니다.

Solution 2. $O(\log n)$

조금 더 수학적인 풀이입니다.

정수 범위 인수분해가 가능하려면 $nx^2 + (n + 1)x - (n + 2) = 0$의 근이 유리수여야 합니다.

✓ 근의 공식에 따라 $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$입니다.

✓ 유리수가 되기 위해서는 $b^2 - 4ac$가 제곱수가 되어야 합니다.

따라서 $(n + 1)^2 + 4n(n + 2)$가 제곱수인지 확인을 해주면 됩니다.

✓ `sqrt(x)` 함수를 이용하면 제곱근을 실수 형태로 구할 수 있습니다.
- 실수 오차를 고려해서 구한 제곱근의 ±3 정도는 실제 제곱근의 후보로 두는 것을 권장합니다.

구한 제곱근을 다시 제곱했을 때 원래의 값이 나온다면 제곱수가 맞다고 확인할 수 있습니다.

제곱수가 맞다고 확인을 했으면, 근의 공식에 따라 x를 계산합니다.

✓ $x = \frac{p}{q}$ 또는 $x = \frac{r}{s}$, 이렇게 두 가지가 나옵니다.

✓ 이를 기약분수 형태로 바꿔줍니다.
- 분모와 분자에 1이 아닌 공약수가 존재할 경우가 그렇지 않은 경우보다 정수 범위 인수분해가 가능하기에 유리하지 않기 때문입니다. (대충.. 무슨 뜻인지 아시겠죠..?)
- 이 과정에는 `gcd(x,y)` 함수를 이용하면 됩니다. 시간 복잡도는 $O(\log(x + y))$입니다.

✓ 분모/분자를 a, b, c, d에 부호 잘 맞춰서 넣은 뒤 전개하면 제대로 나오는지 확인하면 끝!

모범 코드

✓ C++ (**김도훈** *dohoon* , Solution 1)

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

int n;
ll a, b, c, d;

int main() {
    scanf("%d", &n);
    for (a = 1; a*a ≤ n; ++a) if (c = n/a; a*c == n) {
        for (b = 1; b*b ≤ n+2; ++b) if (d = (n+2)/b; b*d == (n+2)) {
            if (a*d-b*c == n+1) printf("%lld %lld %lld %lld",a,-b,c,d), exit(0);
            else if (-a*d+b*c == n+1) printf("%lld %lld %lld %lld",a,b,c,-d), exit(0);
        }
    }
    puts("-1");
}
```

✓ Python (**박기윤** *parkky* , Solution 1)

```python
n=int(input())
X=[]
for i in range(1,n+1):
    if i*i>n:break
    if n%i==0:X.append(i);X.append(n//i)
Y=[]
for i in range(1,n+3):
    if i*i>n+2:break
    if (n+2)%i==0:Y.append(i);Y.append((n+2)//i)
for a in X:
    for b in Y:
        c=n//a
        d=-(n+2)//b
        if a*d+b*c==n+1:print(a,b,c,d);exit()
print(-1)
```

✓ C++ (**김도훈** *dohoon* , Solution 2)

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

int n;

ll sq(ll x) {
    ll y = sqrt(x);
    for (ll i = y-10; i ≤ y+10; ++i) {
        if (i*i == x) return i;
    }
    return 0;
}

int main() {
    cin.tie(0)->sync_with_stdio(0);
    cin >> n;
    ll t = sq(5LL*n*n+10*n+1);
    if (t) {
        ll a, b, c, d, g1, g2;
        a = 2*n, b = -(n+1LL)+t;
        c = 2*n, d = -(n+1LL)-t;
        g1 = gcd(a,b), g2 = gcd(c,d);
        a /= g1, b /= g1;
        c /= g2, d /= g2;
        cout << a << ' ' << -b << ' ' << c << ' ' << -d;
    } else {
        cout << -1;
    }
}
```

✓ Python (**박기윤** *parkky* , Solution 2)

```python
import math
n=int(input())
m=5*n*(n+2)+1
s=math.isqrt(m)
if s*s!=m:print(-1);exit()
x=(-(n+1)+s)//2
y=(-(n+1)-s)//2
d=math.gcd(n,x)
print(n//d,-x//d,d,-y//(n//d))
```

보너스 문제!

주어진 $1 \le a \le b \le 10^{18}$에 대해 $nx^2 + (n+1)x - (n+2)$를 인수분해 가능케 하는 $n \in [a, b]$의 개수는?

- ✓ 이 문제를 푸는데 수학 선생님의 고민(Easy/Hard)를 풀어보는게 도움이 될 거에요!
- ✓ 차형준 선생님이 찾은 논문의 내용을 이용한 문제입니다. 대충 **Platinum** 상위로 예상됩니다.
- ✓ 정말 재밌고 신기한 문제입니다!
- ✓ 문제의 해답이 궁금한 분들은 개인적으로 연락 주시면 알려드리겠습니다!

F. S리그

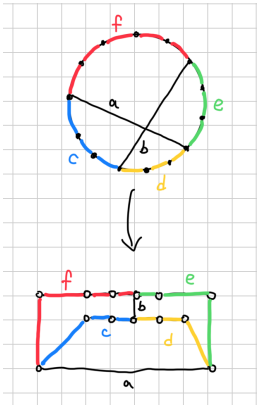
난이도 : **Hard** 태그 : ***ad hoc, constructive, geometry*** 출제 : **김도훈** *dohoon* 세팅 : **김도훈** *dohoon*

뭐가 됐든

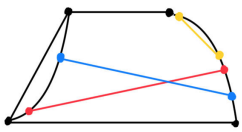
교차만 안 하면 됩니다. 구성하는 방법은 정말 다양하고요.. 예외 상황을 최대한 줄이는 것이 관건입니다.

딱히 설명할 게 없군요.

제가 의도한 방법은 대략 다음과 같습니다:



장래오 *leo020630* 님의 풀이는 대략 다음과 같습니다:



- ✓ 포물선 두 개를 나란히 만들고 끝점들을 잇습니다.
- ✓ 번호의 범위에 따라 남는 하나의 선분은 빨간색, 파란색, 노란색 형태로 나타날 수 있는데, 이는 기존의 선분들과 겹치지 않아서 올바릅니다.

그림에 구애받지 말고 본인만의 창의적인 방식으로 해결해보세요!

코멘트

- ✓ 풀어보지는 않았지만 '19552번: 데이터 제작'에서 아이디어를 얻었어요.
- ✓ 출제자지만 정해 코드를 작성해보지 않았어요 ㅋㅋ
- ✓ 제한을 늘리고 SPJ checker에 샤모스-호이를 짜려다가 접었어요.

모범 코드

✓ C++ (**장래오** *leo020630*)

```cpp
#include <bits/stdc++.h>
using namespace std;
#define fi first
#define se second
typedef long long ll;
typedef pair<ll,ll> pll;
int main()
{
    ios::sync_with_stdio(0); cin.tie(0);
    ll n,a,b,c,d,i,x=0,t=0,k,z=0;
    cin>>n>>a>>b>>c>>d;
    if(a>b) swap(a,b);
    if(c>d) swap(c,d);
    for(i=1;i≤n;i++)
    {
        cout<<i+z<<' '<<x<<'\n';
        if(i==b||i==d) t=1, k=1, z=1e8;
        if(t==0) x+=i;
        else x-=k, k++;
    }
    return 0;
}
```

✓ Python (**박진식** *pjshwa*)

```python
n = int(input())
a = list(map(int, input().split()))
for i in range(1, n):
    print(i, end=' ')
    if i in a[:2]: print(1)
    elif i in a[2:]: print(-1)
    else: print(0)

print(n, end=' ')
L = 10 ** 9
if n in a[:2]: print(L)
else: print(-L)
```

G. 교실 불 끄기

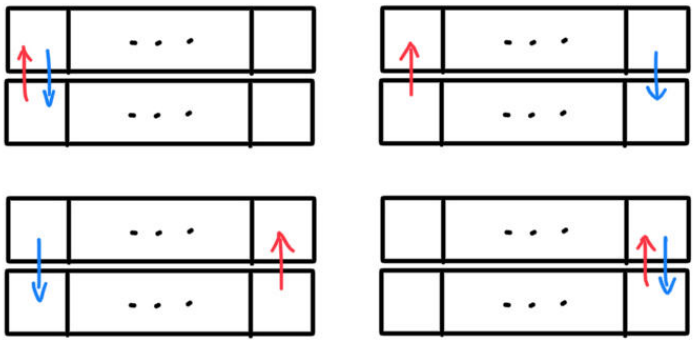
난이도 : **Hard** 태그 : ***dp*** 출제 : **백진서** *jinseo* 세팅 : **백진서** *jinseo*

계단?

최소한의 계단만을 사용하기 위해서는 n층까지 올라가는 와중에 내려가지 않고, 1층까지 내려가는 와중에 올라가지 않아야 합니다. 즉, 이 조건은 계단을 통해 오르내리는 방향이 일정해야 함을 의미합니다.

층과 층 사이

인접한 층을 연결하는 계단은 양쪽 두 개밖에 없습니다. 그 인접한 층 사이에서 계단 사용은 한 번 올라가고, 한 번 내려가는 경우밖에 없습니다. 따라서 다음과 같이 윗쪽 화살표, 아랫쪽 화살표를 분배하는 네 가지 상태들로 나눌 수 있습니다.



✓ 순서대로 상태를 x, y, z, w라고 합시다.

✓ $?(i) \coloneqq i$층과 $i+1$층을 연결하는 상태가 ?일때 소비한 시간의 최솟값

✓ 정의에 잘 맞아들어가게 $x(0) = y(0) = 0;\ z(0) = w(0) = \infty$로 정합시다.

점화식

✓ 이제 상태 표현은 잘 되었으니 점화식을 세워주면 됩니다.
✓ 살짝 양이 많은 느낌이 있지만(..), 대칭적인 부분이 많아서 힘들지 않습니다.

$$x(i) = \min\begin{cases} x(i-1) + 2a_{i,k_i} \\ y(i-1) + m + 1 \\ z(i-1) + m + 1 \\ w(i-1) + 2m + 2 \end{cases}$$

$$y(i) = \min\begin{cases} x(i-1) + m + 1 \\ y(i-1) + 2m + 2 + 2\min_{j=0}^{k_i}\left\{a_{i,j} - a_{i,j+1}\right\} \\ z(i-1) + 2m + 2 \\ w(i-1) + m + 1 \end{cases}$$

$$z(i) = \min\begin{cases} x(i-1) + m + 1 \\ y(i-1) + 2m + 2 \\ z(i-1) + 2m + 2 + 2\min_{j=0}^{k_i}\left\{a_{i,j} - a_{i,j+1}\right\} \\ w(i-1) + m + 1 \end{cases}$$

$$w(i) = \min\begin{cases} x(i-1) + 2m + 2 \\ y(i-1) + m + 1 \\ z(i-1) + m + 1 \\ w(i-1) + 2m + 2 - 2a_{i,1} \end{cases}$$

✓ 문제의 답은 $\min\{x(n-1) + 2m, y(n-1) + m + 1, z(n-1) + m + 1, w(n-1) + 2m\}$가 됩니다.
  – n층은 무조건 다 지나가야 하기 때문이죠.

코멘트

✓ 본 대회날 새벽 4시에 오류를 발견하고 1시간동안 출제자과 검수진이 머리를 맞대고 해결했습니다.
✓ 보통의 dp와는 다르게 형태별로 정보를 저장해 나간다는 점에서 신선한 문제라고 생각합니다.

공부하면 좋은 자료

✓ https://m.blog.naver.com/PostSearchList.naver?blogId=kks227 에서 “동적 계획법” 검색

모범 코드

✓ C++ (김도훈 *dohoon*)

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

const int N = 1e5+3, M = N;
int n, m;
ll k[N], c[N], dp[N][4];
vector<ll> a[N];

int main() {
    scanf("%d%d", &n, &m);
    rep(i,1,n) {
        scanf("%lld", &k[i]);
        a[i].push_back(0);
        rep(j,1,k[i]) {
            ll v; scanf("%lld", &v);
            a[i].push_back(v);
        }
        a[i].push_back(m+1);
    }
    rep(i,1,n) {
        c[i] = 1e9;
        rep(j,0,k[i]) c[i] = min(c[i],a[i][j]-a[i][j+1]);
    }
    dp[0][0] = dp[0][1] = 0;
    dp[0][2] = dp[0][3] = 1e18;
    rep(i,1,n-1) {
        dp[i][0] = 2+min({
            dp[i-1][0]+2*a[i][k[i]],dp[i-1][1]+m+1,dp[i-1][2]+m+1,dp[i-1][3]+2*m+2
        });
        dp[i][1] = 2+min({
            dp[i-1][0]+m+1,dp[i-1][1]+2*(m+1+c[i]),dp[i-1][2]+2*m+2,dp[i-1][3]+m+1
        });
        dp[i][2] = 2+min({
            dp[i-1][0]+m+1,dp[i-1][1]+2*m+2,dp[i-1][2]+2*(m+1+c[i]),dp[i-1][3]+m+1
        });
        dp[i][3] = 2+min({
            dp[i-1][0]+2*m+2,dp[i-1][1]+m+1,dp[i-1][2]+m+1,dp[i-1][3]+2*(m+1-a[i][1])
        });
    }
    printf("%lld", min({dp[n-1][0]+2*m,dp[n-1][1]+m+1,dp[n-1][2]+m+1,dp[n-1][3]+2*m}));
}
```

✓ Python (**박기윤** *parkky*)

```python
n,m=map(int,input().split())
X=[0]*n;Y=[0]*n;Z=[0]*n
for i in range(n):
    k,*L=map(int,input().split())
    L=[0]+L+[m+1]
    X[i]=L[-2]
    Y[i]=m+1-L[1]
    Z[i]=m+1-max(L[i+1]-L[i]for i in range(k+1))
A=[0]*n;B=[0]*n;D=[0]*n
A[n-1]=D[n-1]=2*m;B[n-1]=m+1
for i in range(n-2,-1,-1):
    A[i]=min(A[i+1]+X[i]*2,B[i+1]+m+1,D[i+1]+2*m+2)
    B[i]=min(A[i+1]+m+1,B[i+1]+Z[i]*2,D[i+1]+m+1)
    D[i]=min(A[i+1]+2*m+2,B[i+1]+m+1,D[i+1]+Y[i]*2)
print(min(A[0],B[0])+2*n-2)
```

H. 조별 수업

난이도 : **Challenging** 태그 : ***combinatorics, dp, sqrt*** 출제 : **김도훈** *dohoon* 세팅 : **김도훈** *dohoon*

접근

✓ dp 냄새가 강하게 납니다.
✓ 1...(i - 1)에 대한 문제를 바탕으로 1...i에 대한 문제를 푸는 방법을 고민해봅시다.
- i를 어디다가 집어넣어야 할까 고민해봅시다.
- 생각해보면 i는 1...(i - 1) 중 무엇보다도 큽니다. 그럼 가능한 경우는:
  - i를 최솟값으로 하는 구간을 새로 만들어 넣기
  - 기존의 구간의 빈자리에 넣기
- 가 됩니다.

✓ 그럼 인덱스 정보랑 빈자리 정보 정도만 있으면 dp가 완성될까요?
- 사실 빼먹은 것이 있습니다.
- i를 최솟값으로 하는 구간을 '어디에' 새로 만들어 넣을 지에 대한 결정이 필요합니다.
- 따라서 i까지 오면서 나눈 구간의 개수도 함께 관리해야 합니다.

dynamic programming

✓ $\mathrm{dp}(i, j, k) \coloneqq i$까지 봤을 때 j개의 빈자리가 존재하고, k개로 쪼개진 상태인 경우의 수
- $\mathrm{dp}(0, 0, 0) = 1$.
- $\forall j, k > 0, \mathrm{dp}(0, j, k) = 0$.

점화식은 다음과 같습니다:

$$\mathrm{dp}(i, j, k) = \mathrm{dp}(i - 1, j - i + 1, k - 1) \times i \times k + \mathrm{dp}(i - 1, j + 1, k) \times (j + 1)$$

하나씩 살펴보자면,

✓ $\mathrm{dp}(i - 1, j - i + 1, k - 1) \times i \times k$
- 이 부분은 i를 새로 쪼갠 구간의 최솟값이 되게끔 넣는 방법을 나타냅니다.
- 크기가 i니까 빈자리는 i - 1개 증가하고 넣는 위치는 i가지 자리가 있으니까 i를 곱합니다.
- 또한, 새로 쪼갠 구간이 총 k개의 자리 중 하나니까 k를 곱합니다.
- 쪼갠 구간의 개수는 1개 증가합니다.

✓ $\mathrm{dp}(i - 1, j + 1, k) \times (j + 1)$
- 이 부분은 i를 원래 남은 부분에 넣는 방법을 나타냅니다.
- 넣는 위치가 $j + 1$가지 자리가 있으니까 $j + 1$을 곱합니다.
- 빈자리는 1개 감소합니다.

문제에서 요구하는 답은 $\sum_{k \le n} \mathrm{dp}(n, 0, k)$가 됩니다.

제곱근 최적화

지금까지 말한대로 문제를 풀게 된다면 시간 복잡도는 $O(n^3)$이 될 것입니다. 아직 문제를 해결하기에는 부족한 시간 복잡도죠.

여기서 유명한 아이디어를 하나 이용하면 최적화할 수 있습니다.

Lemma 1. 서로 다른 수들의 합이 n일 때, 이 수들의 개수는 $O(\sqrt{n})$개이다.

Proof. 엄밀하지는 않습니다. 느낌만 생각해주세요.

서로 다른 수들의 개수를 최대화하는 방법을 고려해봅시다.

작은 수들부터 차곡차곡 더해야 최대화할 수 있겠죠?

따라서 $1 + 2 + \cdots + (k-1) + k + (k+\alpha) = n$꼴이 최적입니다. (단, $\alpha < k + 3$)

그럼 $k^2 = O(n) \to k = O(\sqrt{n})$이라는 사실을 알 수 있습니다.

따라서 서로 다른 수의 상한은 $O(\sqrt{n})$입니다. □

- ✓ 각 구간의 길이는 구간 내 번호의 최솟값입니다.
- ✓ 각 번호는 서로 다른 값을 가집니다.
- ✓ 따라서 위 상황에 부합합니다.

직접 계산해보면 구간의 개수의 상한은 넉넉하게 50으로 잡을 수 있습니다.

따라서 $O(n^2\sqrt{n})$에 문제를 풀 수 있습니다.

별해

- ✓ 문제가 주어진 M으로 나눈 나머지를 구하는 것이 아니라 $10^9 + 7$이라는 상수로 나눈 나머지를 구하기를 요구하기 때문에 최적화 없이 세제곱을 짜고서 precomputation을 하여 풀 수도 있을 것으로 예상됩니다.
- ✓ https://solved.ac/search?query=%23precomputation

코멘트

- ✓ 앞으로 조별 수업에는 이런 조 배정을 도입해보세요!

모범 코드

✓ C++ (김도훈 *dohoon*)

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

const int N = 1003, M = 1e9+7;
int n;
ll dp[N][N][51];

int main() {
    scanf("%d", &n);
    dp[0][0][0] = 1;
    rep(i,1,n) {
        rep(j,i-1,n) rep(k,1,min(50,i))
            dp[i][j][k] = (dp[i][j][k] + dp[i-1][j-i+1][k-1]*i*k)%M;
        rep(j,0,n) rep(k,0,min(50,i))
            dp[i][j][k] = (dp[i][j][k] + dp[i-1][j+1][k]*(j+1))%M;
    }
    ll ans = 0;
    rep(k,1,50) ans = (ans+dp[n][0][k])%M;
    printf("%lld",ans);
}
```

✓ Python (김도훈 *dohoon*)

```python
N = 1003; M = int(1e9+7)
dp = [[[0 for i in range(51)] for j in range(N)] for k in range(N)]
dp[0][0][0] = 1
n = int(input())

for i in range(1,n+1):
    for j in range(i-1,n+1):
        for k in range(1,min(50,i)+1):
            dp[i][j][k] = (dp[i][j][k] + dp[i-1][j-i+1][k-1]*i*k)%M
    for j in range(n+1):
        for k in range(min(50,i)+1):
            dp[i][j][k] = (dp[i][j][k] + dp[i-1][j+1][k]*(j+1))%M

ans = 0
for k in range(50): ans = (ans+dp[n][0][k])%M
print(ans)
```

I. 사람이 먼저 되라

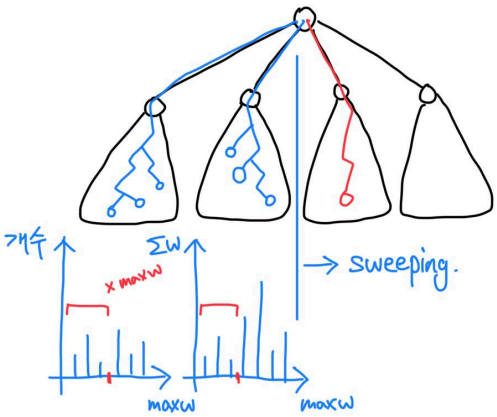
난이도 : **Challenging** 태그 : ***tree, centroid, segtree*** 출제 : **김도훈** *dohoon* 세팅 : **김도훈** *dohoon*

정리하면

트리의 모든 경로에 대해 가중치 합 × 가중치 최댓값을 더하라는 문제입니다.

'모든 경로'에 대한 문제를 풀 때 주로 centroid decomposition이 잘 먹힌다고 합니다. 그럼 한 번쯤 centroid decomposition이라고 생각하고 문제 해결을 시도해 볼 수 있겠죠?

centroid decomposition + segment tree sweeping



- ✓ 그림과 같이 센트로이드를 루트로 잡았다고 생각하고 문제를 풉니다.
- ✓ 센트로이드에 연결된 각 서브트리들을 스위핑하면서
  - 루트부터 정점까지의 가중치 최댓값을 인덱스로 하여
    - 1번 세그트리에 개수를 1 증가시킵니다.
    - 2번 세그트리에는 루트부터 정점까지의 가중치 합만큼 증가시킵니다.
- ✓ 그럼 추가되지 않은 각 정점들을 볼 때,
  - 루트부터 자신까지의 가중치 최댓값보다 작은 인덱스들에 대해
    - 1번 세그트리를 통해 개수 총합을 구하여 가중치 최댓값을 곱한 값을 더해줍니다.
    - 2번 세그트리를 통해 가중치 총합을 구하여 더해줍니다.
- ✓ 이 과정을 반대쪽 방향으로 스위핑을 한 번 더 해주면 문제를 풀 수 있습니다.
  - 한쪽 방향에서 >로 고려했다면 반대쪽 방향에서는 ≥로 고려해야 하는 디테일이 필요합니다.

코멘트

- ✓ 구현량이 꽤 돼요.. 그래서 모범 코드는 적지 않았어요.
- ✓ 기본 시간 제한이 5초라 python 시간 제한은 17초에요..!

공부하면 좋은 자료

- ✓ segment tree
  - https://m.blog.naver.com/kks227/220791986409
  - https://book.acmicpc.net/ds/segment-tree
- ✓ centroid decomposition
  - https://www.youtube.com/watch?v=7v65lpxSdIY
  - https://koosaga.com/145