완전탐색

모든 경우를 다 탐색함

순열
재귀함수
비트마스킹

시간 복잡도란 특정 알고리즘이
어떤 문제를 해결하는데 걸리는 시간을 의미함.

빅-오 표기법
최악의 경우를 계산하는 방식을 빅-오(Big-O) 표기법 이라고함

```
1 for(int i=1; i<=n; i++){
2     printf("hello, world!");
3 }
```

O(N)

```
1 for(int i=1; i<=n; i++){
2     for(int j=1; j<=m; j++){
3         printf("hello, world");
4     }
5 }
```

O(NM)

```
1 for(int i=1; i<=(1<<n); i++){
2     printf("hello, world");
3 }
```

O(2^N)

```
1 for(int i=1; i<=n; i++){
2     int x = n;
3     while(x/=2){
4         printf("hello, world!");
5     }
6 }
```

O(NlogN)

기초유형 1

문제

N개의 정수로 이루어진 수열이 있을 때, 크기가 양수인 부분수열 중에서 그 수열의 원소를 다 더한 값이 S가 되는 경우의 수를 구하는 프로그램을 작성하시오.

입력

첫째 줄에 정수의 개수를 나타내는 N과 정수 S가 주어진다. (1 ≤ N ≤ 20, |S| ≤ 1,000,000) 둘째 줄에 N개의 정수가 빈 칸을 사이에 두고 주어진다. 주어지는 정수의 절댓값은 100,000을 넘지 않는다.

출력

첫째 줄에 합이 S가 되는 부분수열의 개수를 출력한다.

예제 입력 1 복사

```
5 0
-7 -3 -2 5 8
```

예제 출력 1 복사

```
1
```

N이 20이하라 O(2^N) 완전탐색이 가능하다

간편하게 재귀함수, 혹은 비트마스킹으로 풀수있다.

27을 2진수로 표현해보면 11011 이를 아까 문제에 응용해본다면
1은 해당 원소를 선택한 것, 0은 선택하지 않은 것으로 하여 풀 수 있음

2^N까지 반복문을 돌리면서 모든 경우를 확인해보면 된다.

기초유형 1

```cpp
#include "bits/stdc++.h"

using namespace std;
int n,m,arr[22],ans,x;
int main(){
    ios::sync_with_stdio(0);cin.tie(0);
    cin>>n>>m;
    for(int i=1; i<=n; i++) cin>>arr[i];
    for(int i=1; i<(1<<n); i++, x=0){
        for(int j=0; j<n; j++){
            if(i & (1<<j)){
                x += arr[j+1];
            }
        }
        ans += x == m;
    }
    cout<<ans<<'\n';
}

```

이 문제를 재귀함수를 이용해 푼다면 더 쉬운데,
func(bool flag, int idx, int sum)
함수를 정의해 쉽게 해결할 수 있다

기초유형 1

```cpp
#include "bits/stdc++.h"

using namespace std;
int n,m,arr[22];
int dfs(bool flag, int idx, int num){
    if(idx == n + 1){
        if(num == m && flag) return 1;
        return 0;
    }
    return dfs(true, idx+1, num+arr[idx]) + dfs(flag, idx+1,num);
}
int main(){
    ios::sync_with_stdio(0);cin.tie(0);
    cin>>n>>m;
    for(int i=1; i<=n; i++) cin>>arr[i];
    cout<<dfs(false, 1, 0);
}
```

기초유형 2

문제

N*M크기의 직사각형이 있다. 각 칸은 한 자리 숫자가 적혀 있다. 이 직사각형에서 꼭짓점에 쓰여 있는 수가 모두 같은 가장 큰 정사각형을 찾는 프로그램을 작성하시오. 이때, 정사각형은 행 또는 열에 평행해야 한다.

입력

첫째 줄에 N과 M이 주어진다. N과 M은 50보다 작거나 같은 자연수이다. 둘째 줄부터 N개의 줄에 수가 주어진다.

출력

첫째 줄에 정답 정사각형의 크기를 출력한다.

예제 입력 1 복사

```
3 5
42101
22100
22101
```

예제 출력 1 복사

```
9
```

N이 50이하라 O(N^3)으로 해결이 가능하다

모든 좌표 (y,x)에서 정사각형의 한 변의 길이로 가능한 k를 모두 시도해본다.

기초유형 2

```cpp
#include "bits/stdc++.h"

using namespace std;
int n,m,arr[51][51],ans;
bool in(int y,int x){return 1<=y && y<=n && 1<=x && x<=m;}
int main(){
    scanf("%d%d",&n,&m);
    for(int i=1; i<=n; i++){
        for(int j=1; j<=m; j++){
            scanf("%1d",arr[i]+j);
        }
    }
    for(int y=1; y<=n; y++){
        for(int x=1; x<=m; x++){
            for(int k=0; k<=50; k++){
                if(in(y+k,x+k) == false) continue;
                if(arr[y][x] != arr[y][x+k]) continue;
                if(arr[y][x] != arr[y+k][x]) continue;
                if(arr[y][x] != arr[y+k][x+k]) continue;
                ans = max(ans, (k+1)*(k+1));
            }
        }
    }
    printf("%d\n",ans);
}

```

기초유형 3

문제

N이 주어졌을 때, 1부터 N까지의 수로 이루어진 순열을 사전순으로 출력하는 프로그램을 작성하시오.

입력

첫째 줄에 N(1 ≤ N ≤ 8)이 주어진다.

출력

첫째 줄부터 N!개의 줄에 걸쳐서 모든 순열을 사전순으로 출력한다.

예제 입력 1 복사

```
3
```

예제 출력 1 복사

```
1 2 3
1 3 2
2 1 3
2 3 1
3 1 2
3 2 1
```

단순한 순열문제.
std::next_permutation 함수를 사용해보자

next_permutation함수는 [배열의 시작주소, 배열의 끝 주소)를 인자로 받아

다음 순열로 변경하고 true을 반환하거나 다음 순열이 없다면 false를 반환한다.

기초유형 3

```cpp
#include "bits/stdc++.h"

using namespace std;
int n,arr[8];
int main(){
    ios::sync_with_stdio(0);cin.tie(0);
    scanf("%d",&n);
    for(int i=0; i<n; i++) arr[i] = i+1;
    do{
        for(int i=0; i<n; i++) printf("%d ",arr[i]);
        puts("");
    }while(next_permutation(arr, arr+n));
}

```

문제푸는시간

D

7 난쟁이의 키가 100이 되는 걸 뽑아야 하니
모든 난쟁이의 키를 더한 뒤에 2중 포문으로 2 난쟁이를 뽑아
모두 키를 빼보며 100이 될 때 2 난쟁이를 제외한 7 난쟁이를
구하면 쉽게 해결이 가능하다

E

x 와 y가 각각 -999 이상 999 이하인 정수라고 문제에 나와 있기에 2000*2000 반복문을 돌리면 쉽게 해결이 가능

F

순열마다
|A[0] - A[1]| + |A[1] - A[2]| + ... + |A[N-2] - A[N-1]|
계산을 하고 정답을 갱신해 나가면 쉽게 해결이 가능

G

func(int idx, int sum) 을 정의해두고
전역으로 기호의 사용횟수를 관리해주며
idx와 N이 같아질때를 종료조건으로 잡아두고
재귀함수를 돌리면 쉽게 해결이 가능

H

S의 길이가 최대 10이니 10! 1초에 가능하므로
모든 순열에 행운의 문자열인지 아닌지를 판별하면
쉽게 해결이 가능하다

같은 자리에서 전구를 2번 만지는 건 의미가 없다
2번 만지는 거랑 안 만지는 거랑 차이가 없기 때문

모든 칸을 2중 포문으로 돌면서 이전 칸이 전구가 꺼져있으면
현재 칸의 전구를 만짐.

첫 행과 열을 2^n+m 완전 탐색하여 모든 경우에서 전구를
만져보고 모든 행, 열에 대해서 탐색을 하면 된다