BruteForce

브루트포스 알고리즘 정의

브루트포스 알고리즘(완전탐색)이란?

모든 경우의 수를 다 해보는 방법

?.. 이게 끝?

브루트포스 알고리즘 정의

특정한 구현법이나 스킬 같은 것이 굳어 있는게 아니다.

문제에서 주어진 상황에서
순열(nPr)을 사용하던
반복문을 사용하던
재귀를 사용하던
BFS/DFS를 사용하던 모든 경우의 수를 탐색할 수 있으면
브루트포스 알고리즘이다!

브루트포스 알고리즘 정의

특정한 구현법이나 스킬 같은 것이 굳어 있는게 아니다.

문제에서 주어진 상황에서
순열과 조합을 사용하던
반복문을 사용하던
재귀를 사용하던
BFS/DFS를 사용하던 모든 경우의 수를 탐색할 수 있으면
브루트포스 알고리즘이다!

브루트포스 알고리즘 중요한 점

이렇게 간단하면 다 브루트포스를 쓰면 되지 않을까?
-> 브루트포스 알고리즘을 사용하면 간단하게 구현이 가능하지만
시간복잡도가 크게 나와 시간초과가 나올 수도 있다.

그렇다면 **핵심**은?
문제를 보았을 때
브루트포스 알고리즘을 사용해도 되는지 **생각**하는게 중요하다!!!

어떻게 접근? 예)

우리가 값이 변화하지 않는 숫자들의 배열의
특정 구간 (L ≤ i ≤R) 에서의 최대값을 찾고싶다고 해봅시다.
배열의 길이는 N입니다.

가장 쉬운방법부터 생각해봅시다. 단순히 [L, R] 구간의 모
든 숫자들을 한 번씩 보면서 그 중 최대값을 찾으면 됩니다.
이 방법은 시간복잡도가 O(N) 입니다.

어떻게 접근? 예)

이제 문제가 바뀌어서 q번의 구간들에 대해서 최대값을 찾으라고 묻는다면 시간복잡도는 O(QN)이 될 것입니다.

> *[제한]*
>
> $1 \le N \le 100, 1 \le Q \le 1000$
> *시간 제한: 1s*

문제의 제한이 다음과 같을 때 **최악의 경우**를 생각해보면,
100*1000 = 100000입니다.

어떻게 접근? 예)

PS를 할 때 알면 좋은 규칙

보통 1초에 1억번의 연산을 한다고 가정하고 풀면 맞는다고 함

어떻게 접근? 예)

이제 문제가 바뀌어서 q번의 구간들에 대해서 최대값을 찾으라고 묻는다면 시간복잡도는 O(QN)이 될 것입니다.

> *[제한]*
>
> $1 \le N \le 100, 1 \le Q \le 1000$
> *시간 제한: 1s*

문제의 제한이 다음과 같을 때 **최악의 경우**를 생각해보면,
100*1000 = 100000입니다.
이는 제한시간 1초니까 1억보단 작아서 브루트포스로 풀어도 된다!

어떻게 접근? 예)

즉 문제를 읽을 때

1. 문제를 보고
2. 가장 느린 풀이를 생각해보고 시간복잡도를 계산해본다.
3. 문제의 입력의 최악의 경우를 넣어본다.
4. 오 되네? 제한시간에 안걸릴 것 같네?
5. -> 브루트포스로 풀어도 될듯?
6. 코딩까지 잘해야한다

브루트포스에서 가장 중요한 점

즉 이게 브루트포스로 풀 수 있음? 가능?을 알아야한다.

문제풀이

블랙잭(2798)

블랙잭 다국어

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
1 초	128 MB	113780	54549	42106	46.829%

문제

카지노에서 제일 인기 있는 게임 블랙잭의 규칙은 상당히 쉽다. 카드의 합이 21을 넘지 않는 한도 내에서, 카드의 합을 최대한 크게 만드는 게임이다. 블랙잭은 카지노마다 다양한 규정이 있다.

한국 최고의 블랙잭 고수 김정인은 새로운 블랙잭 규칙을 만들어 상근, 창영이와 게임하려고 한다.

김정인 버전의 블랙잭에서 각 카드에는 양의 정수가 쓰여 있다. 그 다음, 딜러는 N장의 카드를 모두 숫자가 보이도록 바닥에 놓는다. 그런 후에 딜러는 숫자 M을 크게 외친다.

이제 플레이어는 제한된 시간 안에 N장의 카드 중에서 3장의 카드를 골라야 한다. 블랙잭 변형 게임이기 때문에, 플레이어가 고른 카드의 합은 M을 넘지 않으면서 M과 최대한 가깝게 만들어야 한다.

N장의 카드에 써져 있는 숫자가 주어졌을 때, M을 넘지 않으면서 M에 최대한 가까운 카드 3장의 합을 구해 출력하시오.

블랙잭(2798)

입력

첫째 줄에 카드의 개수 N(3 ≤ N ≤ 100)과 M(10 ≤ M ≤ 300,000)이 주어진다. 둘째 줄에는 카드에 쓰여 있는 수가 주어지며, 이 값은 100,000을 넘지 않는 양의 정수이다.

합이 M을 넘지 않는 카드 3장을 찾을 수 있는 경우만 입력으로 주어진다.

출력

첫째 줄에 M을 넘지 않으면서 M에 최대한 가까운 카드 3장의 합을 출력한다.

예제 입력 1 복사

```
5 21
5 6 7 8 9
```

예제 출력 1 복사

```
21
```

예제 입력 2 복사

```
10 500
93 181 245 214 315 36 185 138 216 295
```

예제 출력 2 복사

```
497
```

블랙잭(2798)

```java
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int N, M;
        N = sc.nextInt();    //카드 개수
        M = sc.nextInt();    //블랙잭 (넘지않고 가까워야 하는 수)
        int sum = 0;        //세 수의 합
        int tmp = 0;         //새로운 근사치가 나올때까지 이전 근사치 값

        int arr[] = new int[N];
        for(int i=0; i<N; i++) {
            arr[i] = sc.nextInt();    //카드에 적힌 수
        }

        for(int i=0; i<N; i++) {           //N만큼 반복(배열 arr을 순회)
            for(int j=i+1; j<N; j++) {       //i+1번째 위치한 arr부터 순회
                for(int k=j+1; k<N; k++) {    //j+1번째부터 arr 순회
                    sum = arr[i] + arr[j] + arr[k];    //세 수를 더한다.
                    //근사치 (tmp) 보다 크고 블랙잭보다 작거나 같으면 새로운 근사치
                    if(tmp < sum && sum <= M)
                        tmp = sum;
                }
            }
        }
        System.out.println(tmp);    //for문을 다 돌고 가장 근접한 근사치는 tmp에 있다.
    }
}
```

Colored by Color Scripter

```cpp
#include <iostream>
using namespace std;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    cout.tie(NULL);

    int n, m, result = 0;
    int arr[101] = {};
    cin >> n >> m;
    for (int i = 0; i < n; i++)
        cin >> arr[i];

    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            for (int k = j + 1; k < n; k++) {
                if (arr[i] + arr[j] + arr[k] <= m && arr[i] + arr[j] + arr
                    result = arr[i] + arr[j] + arr[k];
            }
        }
    }
    cout << result << '\n';
}
```

```python
n, m = map(int, input().split())
num = list(map(int, input().split()))
l = len(num)
ans = 0
for i in range(0, l-2):
    for j in range(i+1, l-1):
        for k in range(j+1, l):
            if(num[i] + num[j] + num[k] > m):
                continue
            else:
                ans = max(ans ,num[i] + num[j] + num[k])

print(ans)
```

분해합(2231)

분해합 다국어

☆ 한국어

브론즈 II

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
2 초	192 MB	96396	44191	34887	45.429%

문제

어떤 자연수 N이 있을 때, 그 자연수 N의 분해합은 N과 N을 이루는 각 자리수의 합을 의미한다. 어떤 자연수 M의 분해합이 N인 경우, M을 N의 생성자라 한다. 예를 들어, 245의 분해합은 256(=245+2+4+5)이 된다. 따라서 245는 256의 생성자가 된다. 물론, 어떤 자연수의 경우에는 생성자가 없을 수도 있다. 반대로, 생성자가 여러 개인 자연수도 있을 수 있다.

자연수 N이 주어졌을 때, N의 가장 작은 생성자를 구해내는 프로그램을 작성하시오.

입력

첫째 줄에 자연수 N(1 ≤ N ≤ 1,000,000)이 주어진다.

출력

첫째 줄에 답을 출력한다. 생성자가 없는 경우에는 0을 출력한다.

예제 입력 1 복사

```
216
```

예제 출력 1 복사

```
198
```

분해합(2231)

입력 숫자보다 작은 숫자 만큼 비교하여 같아질 때 까지 반복

분해합(2231)

```cpp
#include <iostream>
#include <cstring>
#include <string>
using namespace std;

int main(int argc, char *argv[])
{
	int num;
	int sum;
	int part;

	cin >> num;

	for (int i = 1; i < num; ++i) {
		sum = i;
		part = i;

		while (part) {
			sum += part % 10;
			part /= 10;
		}

		if (num == sum) {
			cout << i << endl;
			return 0;
		}
	}

	cout << "0" << endl;

	return 0;
}
```

```python
n = int(input())

result = 0
for i in range(1,n+1):
    a = list(map(int,str(i)))
    result = i + sum(a)
    if result == n:
        print(i)
        break
    if i == n:
        print(0)
```

```java
import java.util.*;

public class Main {
    public static void main(String args[]) {
        Scanner input = new Scanner(System.in);
        int num = input.nextInt();
        int tot, len;
        boolean flag = false; // 생성자 찾음 유무 표시
        for(int i=1; i<num; i++){
            tot = i; // 분해합 과정 중 자기 자신 더하기
            len = 1;
            while(i/len > 0) { // 각 자릿수 더하기
                tot += (i/len)%10;
                len *= 10;
            }
            if(tot == num){
                System.out.println(i);
                flag = true; // 생성자 찾음
                break;
            }
        }
        if(!flag) // 생성자 못찾았을 때
            System.out.println(0);
    }
}
```

순열과 조합

= python

도영이가 만든 맛있는 음식(2961)

2961번 | 제출 | 맞힌 사람 | 숏코딩 | 재채점 결과 | 채점 현황 | 내 제출 | 질문 검색

도영이가 만든 맛있는 음식 다국어

☆ 한국어

실버 II

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
1 초	128 MB	7352	3673	2773	48.846%

도영이가 만든 맛있는 음식(2961)

문제

도영이는 짜파구리 요리사로 명성을 날렸었다. 이번에는 이전에 없었던 새로운 요리에 도전을 해보려고 한다.

지금 도영이의 앞에는 재료가 N개 있다. 도영이는 각 재료의 신맛 S와 쓴맛 B를 알고 있다. 여러 재료를 이용해서 요리할 때, 그 음식의 신맛은 사용한 재료의 신맛의 곱이고, 쓴맛은 합이다.

시거나 쓴 음식을 좋아하는 사람은 많지 않다. 도영이는 재료를 적절히 섞어서 요리의 신맛과 쓴맛의 차이를 작게 만들려고 한다. 또, 물을 요리라고 할 수는 없기 때문에, 재료는 적어도 하나 사용해야 한다.

재료의 신맛과 쓴맛이 주어졌을 때, 신맛과 쓴맛의 차이가 가장 작은 요리를 만드는 프로그램을 작성하시오.

입력

첫째 줄에 재료의 개수 N($1 \le N \le 10$)이 주어진다. 다음 N개 줄에는 그 재료의 신맛과 쓴맛이 공백으로 구분되어 주어진다. 모든 재료를 사용해서 요리를 만들었을 때, 그 요리의 신맛과 쓴맛은 모두 1,000,000,000보다 작은 양의 정수이다.

출력

첫째 줄에 신맛과 쓴맛의 차이가 가장 작은 요리의 차이를 출력한다.

도영이가 만든 맛있는 음식(2961)

예제 입력 1 복사

```
1
3 10
```

예제 출력 1 복사

```
7
```

예제 입력 2 복사

```
2
3 8
5 8
```

예제 출력 2 복사

```
1
```

예제 입력 3 복사

```
4
1 7
2 6
3 8
4 9
```

예제 출력 3 복사

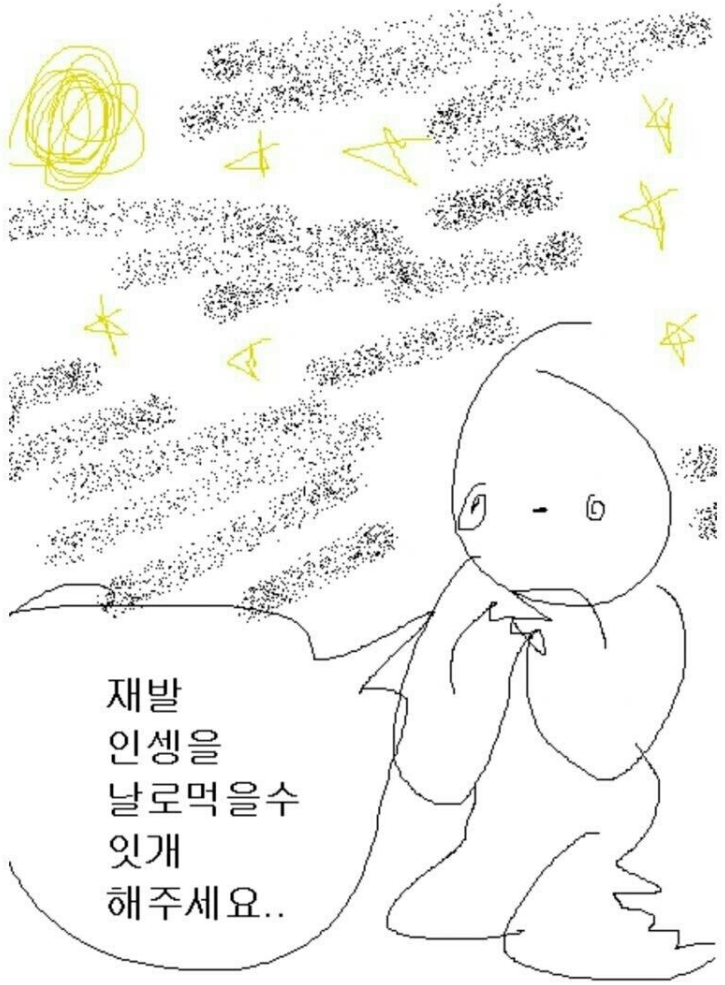
```
1
```

2, 3, 4번 재료를 사용한다면, 요리의 신맛은 2×3×4=24, 쓴맛은 6+8+9=23이 된다. 차이는 1이다.

도영이가 만든 맛있는 음식(2961)

조합으로 나열 후 최소 값 구하기

```python
import sys
from itertools import combinations

n = int(input())
arr = []

for _ in range(n):
    arr.append(list(map(int, input().split())))

com = []
for i in range(1, n+1):
    com.append(combinations(arr, i))

answer = 100000000

for i in com:
    for j in i:
        sour = 1
        bitter = 0
        for e in j:
            sour *= e[0]
            bitter += e[1]

        answer = min(answer, abs(sour-bitter))
print(answer)
```



도영이가 만든 맛있는 음식(2961)

조합으로 나열 후 최소 값 구하기

```cpp
#include <iostream>
#include <algorithm>
#include <cmath>
using namespace std;

int n;
pair<int, int> input[10];
//1<=n<=10
pair<int, int> result[10];
int answer = 987654321;
void combination(int idx, int cnt) {
	if (cnt > 0) {
		int s = result[0].first;
		int b = result[0].second;
		for (int i = 1; i < cnt; i++) {
			s *= result[i].first;
			b += result[i].second;
		}
		answer = min(answer, abs(s - b));
	}
	if (cnt == n) {
		return;
	}
	for (int i = idx; i < n; i++) {
		result[cnt] = { input[i].first,input[i].second};
		combination(i + 1, cnt + 1);
	}
}

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

	cin >> n;
	for (int i = 0; i < n; i++) {
		cin >> input[i].first;
		cin >> input[i].second;
	}
	combination(0,0);
	cout << answer;
	return 0;
}
```

도영이가 만든 맛있는 음식(2961)

조합으로 나열 후 최소 값 구하기

```
        // 조합
        private static void combination(int max, int start, int count) {
                // 경계
                // 뽑은 재료의 개수가 원하는 재료의 개수면 재귀 종료
                if (count == max) {
                        // 신맛
                        int t1 = 1;
                        // 쓴맛
                        int t2 = 0;

                        // 선택한 재료의 인덱스에 해당하는 맛의 점수 계산
                        for (int idx : material) {
                                t1 *= taste1[idx];
                                t2 += taste2[idx];
                        }

                        // 해당 재료로 만든 음식의 점수 차이가 더 작으면
                        if (minDiff > Math.abs(t1 - t2)) {
                                minDiff = Math.abs(t1 - t2);
                        }
                        return;
                }

                // 재귀
                for (int i = start; i < N; i++) {
                        material[count] = i;
                        combination(max, i + 1, count + 1);
                }

        }
```

```
public class Problem2961 {

        // 신맛 배열
        static int[] taste1;
        // 쓴맛 배열
        static int[] taste2;
        // 선택한 재료 배열
        static int[] material;
        // 재료의 개수
        static int N;
        // 맛의 차이
        static int minDiff = Integer.MAX_VALUE;

        // 2961. 도영이가 만든 맛있는 음식
        public static void main(String[] args) throws
NumberFormatException, IOException {

                BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));

                // 재료 개수
                N = Integer.parseInt(br.readLine());

                // 배열 크기 입력
                taste1 = new int[N];
                taste2 = new int[N];

                // 배열에 데이터 입력
                for (int i = 0; i < N; i++) {
                        StringTokenizer st = new
StringTokenizer(br.readLine());

                        int t1 = Integer.parseInt(st.nextToken());
                        int t2 = Integer.parseInt(st.nextToken());

                        taste1[i] = t1;
                        taste2[i] = t2;
                }

                // 재료 개수별 조합 실행
                for (int i = 1; i <= N; i++) {
                        material = new int[i];
                        combination(i, 0, 0);
                }

                // 결과 출력
                System.out.println(minDiff);

        }
```

앵콜 원해요?

우리가 좋아하는 DP에? 완전탐색?

고생하셨습니다…