DFS

최백준 choi@startlink.io

순열 사용하기

팩토리얼

Factorial

- 3! = 6
- 4! = 24
- 5! = 120
- 6! = 720
- 7! = 5,040
- 8! = 40,320
- 9! = 362,880
- 10! = 3,628,800
- 11! = 39,916,800
- 12! = 479,001,600
- 13! = 6,227,020,800

차이를 최대로

https://www.acmicpc.net/problem/10819

- 수 N개가 주어졌을 때 ($3 \leq N \leq 8$)
- $|A[0] - A[1]| + |A[1] - A[2]| + \ldots + |A[N-2] - A[N-1]|$
- 를 최대로 하는 문제

차이를 최대로

https://www.acmicpc.net/problem/10819

- N! = 8! = 40320
- 모든 경우를 다해봐도 된다.

- 수를 next_permutation을 이용해 모든 경우를 다 해본다

차이를 최대로

https://www.acmicpc.net/problem/10819

```
do {
    int temp = calculate(a);
    ans = max(ans,temp);
} while(next_permutation(a.begin(), a.end()));
```

차이를 최대로

https://www.acmicpc.net/problem/10819

- C++: https://gist.github.com/Baekjoon/fb602ec4b6778757d717
- Java: https://gist.github.com/Baekjoon/1b03fb9b88d7de8cd959

외판원 순회 2

https://www.acmicpc.net/problem/10971

- 영어로 Travelling Salesman Problem (TSP)
- 1번부터 N번까지 번호가 매겨져있는 도시가 있다
- 한 도시에서 시작해 N개의 모든 도시를 거쳐 다시 원래 도시로 돌아오려고 한다 (한 번 갔던 도시로는 다시 갈 수 없다)
- 이 때, 가장 적은 비용을 구하는 문제

- W[i][j] = i -> j 비용

외판원 순회 2

https://www.acmicpc.net/problem/10971

- $2 \leq N \leq 10$
- N! = 10! = 3628800
- 모든 경우를 다해봐도 시간 안에 나온다

외판원 순회 2

https://www.acmicpc.net/problem/10971

- $2 \leq N \leq 10$
- N! = 10! = 3628800
- 모든 경우를 다해봐도 시간 안에 나온다
- 모든 경우 = N!
  - 비용 계산 = N
- 시간복잡도: O(N*N!)

외판원 순회 2

https://www.acmicpc.net/problem/10971

```
do {
    bool ok = true;
    int sum = 0;
    for (int i=0; i<n-1; i++) {
        if (w[d[i]][d[i+1]] == 0) ok = false;
        else sum += w[d[i]][d[i+1]];
    }
    if (ok && w[d[n-1]][d[0]] != 0) {
        sum += w[d[n-1]][d[0]];
        if (ans > sum) ans = sum;
    }
} while (next_permutation(d.begin(), d.end()));
```

외판원 순회 2

https://www.acmicpc.net/problem/10971

- O(N*N!)
- C++: https://gist.github.com/Baekjoon/a62f0b1263752c8d1a75
- Java: https://gist.github.com/Baekjoon/a5450f44bc19da72f9ac

- O(N!)
- C++: https://gist.github.com/Baekjoon/3eeee9003b22cffb2a76
- C++ 2: https://gist.github.com/Baekjoon/45c47a211c3be61e054a
- Java: https://gist.github.com/Baekjoon/88bfb6c2e54bb399beb2

재귀호출 사용하기

재귀함수 사용하기

Recursion

- 재귀함수를 잘 설계해야 한다

1, 2, 3 더하기

https://www.acmicpc.net/problem/9095

- 정수 n을 1, 2, 3의 조합으로 나타내는 방법의 수를 구하는 문제
- n = 4
- 1+1+1+1
- 1+1+2
- 1+2+1
- 2+1+1
- 2+2
- 1+3
- 3+1

1, 2, 3 더하기

https://www.acmicpc.net/problem/9095

- go(count, sum, goal)
- 숫자 count개로 합 sum을 만드는 경우의 수

1, 2, 3 더하기

https://www.acmicpc.net/problem/9095

- go(count, sum, goal)
- 숫자 count개로 합 sum을 만드는 경우의 수
- 불가능한 경우
  - count > 10
  - sum > goal
- 가능한 경우
  - sum == goal

1, 2, 3 더하기

https://www.acmicpc.net/problem/9095

- go(count, sum, goal)
- 숫자 count개로 합 sum을 만드는 경우의 수
- 다음 경우
  - 1을 사용하는 경우
    - go(count+1, sum+1, goal)
  - 2를 사용하는 경우
    - go(count+1, sum+2, goal)
  - 3을 사용하는 경우
    - go(count+1, sum+3, goal)

1, 2, 3 더하기

https://www.acmicpc.net/problem/9095

```
int go(int count, int sum, int goal) {
    if (count > 10) return 0;
    if (sum > goal) return 0;
    if (sum == goal) return 1;
    int now = 0;
    for (int i=1; i<=3; i++) {
        now += go(count+1, sum+i, goal);
    }
    return now;
}
```

1, 2, 3 더하기

https://www.acmicpc.net/problem/9095

- C++: https://gist.github.com/Baekjoon/3235f76fe44c1ad17648
- Java: https://gist.github.com/Baekjoon/bdeba307e9e6d1e80fc7

암호 만들기

https://www.acmicpc.net/problem/1759

- 암호는 서로 다른 L개의 알파벳 소문자들로 구성되며 최소 한 개의 모음과 최소 두 개의 자음으로 구성되어 있다
- 암호를 이루는 알파벳이 암호에서 증가하는 순서로 배열되었어야 한다
- 암호로 사용할 수 있는 문자의 종류는 C가지
- 가능성 있는 암호를 모두 구하는 문제

암호 만들기

https://www.acmicpc.net/problem/1759

- L = 4, C = 6
- 사용 가능한 알파벳: a t c i s w
- 가능한 암호
- acis
- acit
- aciw
- acst
- acsw
- actw
- aist
- aisw
- aitw
- astw
- cist
- cisw
- citw
- istw

암호 만들기

https://www.acmicpc.net/problem/1759

- go(n, alpha, password, i)
  - n: 만들어야 하는 암호의 길이
  - alpha: 사용할 수 있는 알파벳
  - password: 현재까지 만든 암호
  - i: 사용할지 말지 결정해야 하는 알파벳의 인덱스

암호 만들기

https://www.acmicpc.net/problem/1759

- go(n, alpha, password, i)
  - n: 만들어야 하는 암호의 길이
  - alpha: 사용할 수 있는 알파벳
  - password: 현재까지 만든 암호
  - i: 사용할지 말지 결정해야 하는 알파벳의 인덱스
- 언제 답인지 아닌지 확인해야 하나?
  - n == password.length()
- 다음
  - i번째 알파벳을 사용하는 경우
  - i번째 알파벳을 사용하지 않는 경우

암호 만들기

https://www.acmicpc.net/problem/1759

- 다음
  - i번째 알파벳을 사용하는 경우
    - go(n, alpha, password+alpha[i], i+1)
  - i번째 알파벳을 사용하지 않는 경우
    - go(n, alpha, password, i+1)

암호 만들기

https://www.acmicpc.net/problem/1759

```
void go(int n, vector<char> &alpha, string password, int i) {
    if (password.length() == n) {
        if (check(password)) {
            cout << password << '\n';
        }
        return;
    }
    if (i >= alpha.size()) return;
    go(n, alpha, password+alpha[i], i+1);
    go(n, alpha, password, i+1);
}
```

암호 만들기

https://www.acmicpc.net/problem/1759

```
bool check(string &password) {
    int ja = 0;
    int mo = 0;
    for (char x : password) {
        if (x == 'a' || x == 'e' || x == 'i' || x == 'o' || x ==
'u') {
            mo += 1;
        } else {
            ja += 1;
        }
    }
    return ja >= 2 && mo >= 1;
}
```

암호 만들기

https://www.acmicpc.net/problem/1759

- C++: https://gist.github.com/Baekjoon/dff42ddf0ae028f6b7f1
- Java: https://gist.github.com/Baekjoon/e92cfec2c020cd62b8ef

알파벳

https://www.acmicpc.net/problem/1987

- 세로 R칸, 가로 C칸으로 된 표 모양의 보드가 있다
- 보드의 각 칸에는 대문자 알파벳이 하나씩 적혀 있고, 좌측 상단 칸 (1행 1열) 에는 말이 놓여 있다
- 말은 상하좌우로 인접한 네 칸 중의 한 칸으로 이동할 수 있다
- 같은 알파벳이 적힌 칸을 두 번 지날 수 없다
- 좌측 상단에서 시작해서, 말이 최대한 몇 칸을 지날 수 있는지를 구하는 문제

알파벳

https://www.acmicpc.net/problem/1987

- go(board, check, x, y, cnt)
  - board: 보드
  - check: 방문한 알파벳
  - x, y: 현재 위치
  - cnt: 방문한 칸의 수

알파벳

https://www.acmicpc.net/problem/1987

- go(board, check, x, y, cnt)
  - board: 보드
  - check: 방문한 알파벳
  - x, y: 현재 위치
  - cnt: 방문한 칸의 수
- 새로운 칸 nx, ny로 이동할 수 있는 경우
  - go(board, check, nx, ny, cnt+1)
    - 이 때, check는 변경해 줘야함

알파벳

https://www.acmicpc.net/problem/1987

```
void go(vector<string> &board, vector<bool> &check, int x, int y, int
cnt) {
    if (cnt > ans) ans = cnt;
    for (int k=0; k<4; k++) {
        int nx = x+dx[k];
        int ny = y+dy[k];
        if (nx >= 0 && nx < board.size() && ny >= 0 && ny <
board[0].size()) {
            if (check[board[nx][ny]-'A'] == false) {
                check[board[nx][ny]-'A'] = true;
                go(board, check, nx, ny, cnt+1);
                check[board[nx][ny]-'A'] = false;
            }
        }
    }
}
```

알파벳

https://www.acmicpc.net/problem/1987

- go(board, check, x, y)
- board: 보드
- check: 방문한 알파벳
- x, y: 현재 위치
- 리턴 값: 방문할 수 있는 칸의 최대 개수
- 의미: (x, y)에서 이동을 시작하고, 방문한 알파벳이 check일 때, 방문할 수 있는 칸의 최대 개수

알파벳

https://www.acmicpc.net/problem/1987

```
int go(vector<string> &board, vector<bool> &check, int x, int y) {
    int ans = 0;
    for (int k=0; k<4; k++) {
        int nx = x+dx[k], ny = y+dy[k];
        if (nx >= 0 && nx < board.size() && ny >= 0 && ny <
board[0].size()) {
            if (check[board[nx][ny]-'A'] == false) {
                check[board[nx][ny]-'A'] = true;
                int next = go(board, check, nx, ny);
                if (ans < next) ans = next;
                check[board[nx][ny]-'A'] = false;
            }
        }
    }
    return ans + 1;
}
```

알파벳

https://www.acmicpc.net/problem/1987

- C++: https://gist.github.com/Baekjoon/f412bcc16f3b3f0cbffd
- Java: https://gist.github.com/Baekjoon/411767759d38830b5911

부분집합의 합

https://www.acmicpc.net/problem/1182

- 서로 다른 N개의 정수로 이루어진 집합이 있을 때, 이 집합의 공집합이 아닌 부분집합 중에서 그 집합의 원소를 다 더한 값이 S가 되는 경우의 수를 구하는 문제
- $1 \leq N \leq 20$

부분집합의 합

https://www.acmicpc.net/problem/1182

- C++: https://gist.github.com/Baekjoon/5d90f9d1582559c619ad2821b126ac16
- Java: https://gist.github.com/Baekjoon/923eddd3d8d3bef43372433c83afb6cf

비트마스크 사용하기

부분집합의 합

https://www.acmicpc.net/problem/1182

- 서로 다른 N개의 정수로 이루어진 집합이 있을 때, 이 집합의 공집합이 아닌 부분집합 중에서 그 집합의 원소를 다 더한 값이 S가 되는 경우의 수를 구하는 문제
- $1 \leq N \leq 20$

부분집합의 합

https://www.acmicpc.net/problem/1182

- 모든 집합의 개수 = 2^N
- 모든 집합을 구해보면 된다!

부분집합의 합

https://www.acmicpc.net/problem/1182

- 전체 집합 = (1<<N) – 1

```
for (int i=0; i<(1<<n); i++) {

}
```

부분집합의 합

https://www.acmicpc.net/problem/1182

- 전체 집합 = (1<<N) – 1
- 공집합은 제외해야 한다

```
for (int i=1; i<(1<<n); i++) {

}
```

부분집합의 합

https://www.acmicpc.net/problem/1182

- 전체 집합 = (1<<N) – 1
- 공집합은 제외해야 한다
- 집합에 무엇이 포함되어 있는지 확인하기

```
for (int i=1; i<(1<<n); i++) {
    for (int k=0; k<n; k++) {
        if (i&(1<<k)) {
        }
    }
}
```

부분집합의 합

https://www.acmicpc.net/problem/1182

```
for (int i=1; i<(1<<n); i++) {
    int sum = 0;
    for (int k=0; k<n; k++) {
        if (i&(1<<k)) {
            sum += a[k];
        }
    }
    if (sum == s) {
        ans += 1;
    }
}
```

부분집합의 합

https://www.acmicpc.net/problem/1182

- C++: https://gist.github.com/Baekjoon/f4154089addcd1adacc5
- Java: https://gist.github.com/Baekjoon/bddda372acf45d698817

째로탈출 2

https://www.acmicpc.net/problem/13460

- 보드의 상태가 주어졌을 때, 최소 몇 번 만에 빨간 구슬을 구멍을 통해 빼낼 수 있는지 구하는 문제
- 만약, 10번 이내에 움직여서 빨간 구슬을 구멍을 통해 빼낼 수 없으면 -1을 출력

째로탈출 2

https://www.acmicpc.net/problem/13460

- C++: https://gist.github.com/Baekjoon/c9dbf1e5eae35c2f4501f410482c1469
- Java: https://gist.github.com/Baekjoon/d462fd8f86659be5c7244d67113c5ff6