#중급반 1주차

재귀적 문제 해결 방법

<Backtracking, Divide and Conquer>

#백트래킹
<Backtracking>

답이 될 수 있는 경우들을 재귀적으로 하나하나 탐색해보며,
답이 될 수 없는 상태가 되었을 때는 뒤로 되돌아가서 답을 찾는 기법을 의미합니다.

1. 뒤로 되돌아간다는 성질 때문에 Back + Tracking이라는 이름이 붙었습니다.
2. 답이 될 수 없는 경우들을 추가적인 탐색 없이 건너뛰게 되며, 이를 가지치기라고 부릅니다.

#백트래킹

<Backtracking> - 연습 문제

3 모든 순열 (BOJ #10974)

<문제 설명>

- 정수 N이 주어집니다.
- 길이 N의 순열을 모두 출력하면 됩니다.

<제약 조건>

- $1 \leq N \leq 8$

#백트래킹

<Backtracking> - 연습 문제

3 모든 순열 (BOJ #10974)

<문제 해설>

f(N, P)를 길이가 N이고, 앞부분이 P로 고정된 모든 순열이라고 생각해봅시다.

예를 들면, f(4, [1, 2])는 { [1, 2, 3, 4], [1, 2, 4, 3] }이 되고,

f(3, []) = { [1, 2, 3], [1, 3, 2], [2, 1, 3], [2, 3, 1], [3, 1, 2], [3, 2, 1] }이 됩니다.

이제 이걸 백트래킹을 통해, 재귀적으로 풀어봅시다.

#백트래킹

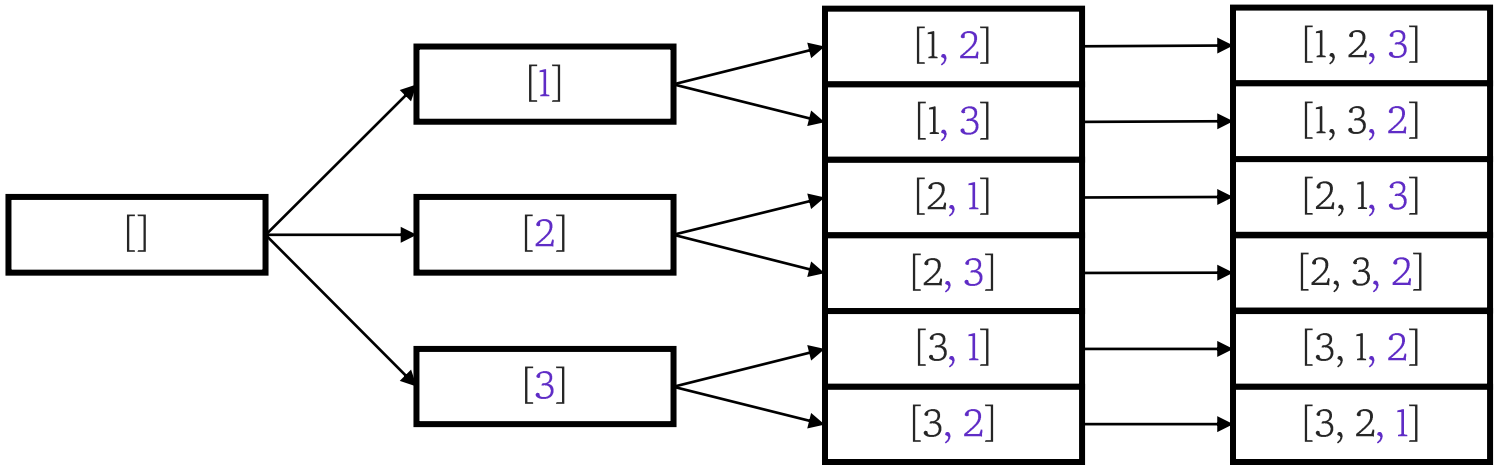
<Backtracking> - 연습 문제

3 모든 순열 (BOJ #10974)

<문제 해설>

f(N, P)에서 생각해볼 수 있는 경우는 아래 두 가지 경우가 있습니다.

- P가 길이 N의 순열이라면: P를 그대로 출력하면 됩니다.
- 아니라면: P에 아직 넣지 않은 수를 하나씩 골라서 넣어보면 됩니다.



#백트래킹

<Backtracking> - 연습 문제 구현

구현의 편의를 위해 P를 전역 변수로 옮기고, 대신 P의 길이 index를 함수로 넘겼습니다.
또한, '어떤 수를 사용했는가' 여부를 chk 배열로 관리하고 있습니다.

```
int P[8], N;
bool chk[8];
void f(int index){
    if (index == N){
        for (int i = 0; i < N; i++){ cout << P[i] << ' '; }
        cout << endl;
        return;
    }
    for (int x = 1; x <= N; x++){
        if (chk[x]){ continue; }
        chk[x] = 1; P[index] = x;
        f(index+1);
        chk[x] = 0; P[index] = 0;
    }
}
```

#백트래킹

<Backtracking> - 연습 문제 구현

탐색을 하는 부분은 아까 f(N, P)로 했던 것과 비슷합니다.

만약 P의 길이가 N과 동일하다면 P를 출력 후 탐색을 종료하고,

아니라면 아직 등장하지 않은 수 x를 P의 맨 뒤에 넣고 탐색을 진행합니다.

```
int P[8], N;
bool chk[8];
void f(int index){
    if (index == N){
        for (int i = 0; i < N; i++){ cout << P[i] << ' '; }
        cout << endl;
        return;
    }
    for (int x = 1; x <= N; x++){
        if (chk[x]){ continue; }
        chk[x] = 1; P[index] = x;
        f(index+1);
        chk[x] = 0; P[index] = 0;
    }
}
```

#분할 정복

<Divide and Conquer>

어떤 문제를 풀기 위해, 더 작은 부분 문제들로 나누고 (분할)
그 작은 문제들을 푼 뒤 합쳐서 (정복) 답을 알아내는 방법입니다.

1. 동적 계획법 (DP)과 다른 점은, 부분 문제끼리 중복되지 않는다는 점입니다.
   이는 메모이제이션을 굳이 하지 않아도 된다는 걸 의미하죠.
2. 하지만 그 점만 빼면, DP와 크게 다른 점이 없다고 봐도 됩니다.

#분할 정복

<Divide and Conquer> - 전체적인 틀

어떤 문제 P가 주어졌을 때...

1. 만약 P를 간단하게 풀 수 있다면, 그대로 푼다.
2. 아니라면, 부분문제들로 적절히 나눈 뒤, 이를 각각 푼다.
3. 부분문제들의 답들을 적절히 합해서 문제 P의 답 A를 알아낸다.

이렇게 두고 보면 안 어려워 보이지만, 결국 저 '적절히'를 어떻게 잡고 처리하는지가 문제의 관건이 됩니다.

그와는 별개로, 이렇게 틀을 잡고 보니 DP와의 차이가 별로 없죠?

#분할 정복

<Divide and Conquer> - 연습 문제

2 색종이 만들기 (BOJ #2630)

<문제 설명>

- 0과 1로 구성된 N × N 크기의 종이가 주어진다.
- 문제에서 주는 규칙에 따라 종이를 붙일 때,
  붙이게 되는 종이의 수를 출력한다.

<제약 조건>

- $N = 2^k, 1 \leq k \leq 7$

#분할 정복

<Divide and Conquer> - 연습 문제

2 색종이 만들기 (BOJ #2630)

<문제 해설>

정사각형 구간 [y1, y2] × [x1, x2]에서 주어진 문제를 푸는 함수 f(y1, y2, x1, x2)를 정의해봅시다.

그럼, 남은 건 문제에서 쓰여진 대로 재귀적인 구현을 해주면 됩니다.

#분할 정복

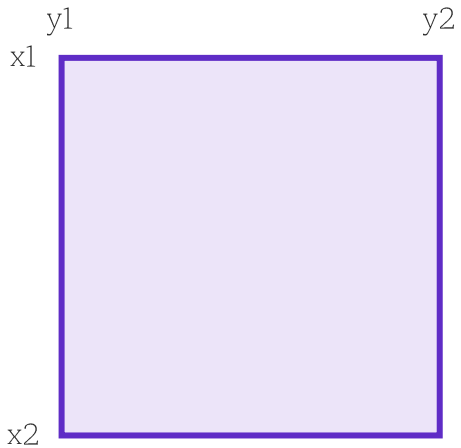
<Divide and Conquer> - 연습 문제

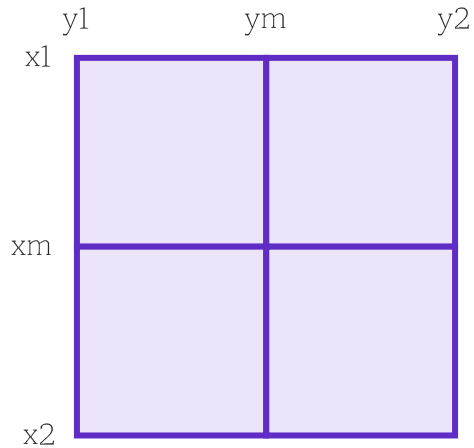
2 색종이 만들기 (BOJ #2630)

<문제 해설>

만약 [y1, y2] × [x1, x2]가 모두 같은 색으로 이루어져 있다면, 그 색깔의 종이를 하나 사용한 뒤 탐색을 종료합니다.

아니라면, y1과 y2의 중점 y'과, x1과 x2의 중점 x'을 기준으로 4조각으로 나눈 뒤, 재귀적으로 문제를 풀어주면 됩니다.





#분할 정복

<Divide and Conquer> - 연습 문제 구현

우선, 우리가 보고 있는 구간이 색깔 하나로만 나타낼 수 있다면, 이에 맞는 색깔의 종이 하나로 처리합니다.

남은 경우는, y와 x의 중간점을 찾은 뒤
그 중간점을 기준으로 4조각으로 나눠서 답을 구해주면 됩니다.

```
int a[130][130];

int ans0 = 0, ans1 = 0;
void f(int y1, int y2, int x1, int x2){
    bool c0 = true, c1 = true;
    for (int y = y1; y <= y2; y++){
        for (int x = x1; x <= x2; x++){
            if (a[y][x] != 0){ c0 = false; }
            if (a[y][x] != 1){ c1 = false; }
        }
    }
    if (c0){ ans0 += 1; return; }
    if (c1){ ans1 += 1; return; }
    int ym = (y1+y2) / 2, xm = (x1+x2) / 2;
    f(y1, ym, x1, xm); f(y1, ym, xm+1, x2);
    f(ym+1, y2, x1, xm); f(ym+1, y2, xm+1, x2);
}
```

#분할 정복

<Divide and Conquer> - 연습 문제

곱셈 (BOJ #1629)

<문제 설명>

- 세 정수 A, B, C가 주어집니다.
- $A^B \bmod C$의 값을 출력하면 됩니다.

<제약 조건>

- $1 \leq A, B, C \leq 2\,147\,483\,647$

#분할 정복

<Divide and Conquer> - 연습 문제

곱셈 (BOJ #1629)

<문제 해설>

A^B을 계산하는 건 쉽습니다. A를 B번 곱해주면 되죠.

하지만, 이대로는 시간 복잡도가 O(B)가 되니, 시간 초과를 받게 됩니다.

#분할 정복

<Divide and Conquer> - 연습 문제

곱셈 (BOJ #1629)

<문제 해설>

지수 법칙을 응용해봅시다.

$A^B = \left(A^{B/2}\right)^2 = A^{B/2} \times A^{B/2}$으로 나타낼 수 있습니다.

다만 B에 실수가 들어가게 된다면 복잡해질 것 같으니, B가 홀수일 때와 짝수일 때로 나눠서

- B가 짝수라면: $A^B = A^{B/2} \times A^{B/2}$
- B가 홀수라면: $A^B = A^{B-1} \times A$

를 계산하면 됩니다.

기저 조건은, B = 0일 때 $A^0 = 1$을 사용해주면 됩니다.

#분할 정복

<Divide and Conquer> - 연습 문제

곱셈 (BOJ #1629)

<문제 해설>

이렇게 하면, 시간 복잡도는 어떻게 될까요?

B가 짝수였다면, B가 절반으로 나눠집니다.

B가 홀수였다면, 함수를 한 번 거치면 짝수가 되고, 그 뒤로는 B가 절반으로 나눠집니다.

B가 무슨 수가 되든, 함수를 2번 거치면 절반으로 나눠지므로

시간 복잡도는 $O(2 \times \log B) = O(\log B)$가 됩니다.

#분할 정복

<Divide and Conquer> - 연습 문제 구현

요약하자면, B = 0일 때, B가 짝수일 때, B가 홀수일 때로 나눠서 해결해주면 됩니다.

문제 해설에서는 mod C 부분을 뺐었는데, 그냥 계산할 때마다 mod C로 나눈 나머지를 계산해주면 됩니다.
A × B mod C = ((A mod C) × (B mod C)) mod C이라는 식이 성립하므로, 계산 결과에 대해 걱정할 필요도 없습니다!

함수 이름을 pow로 지으면, Built-In 함수와 이름이 충돌되므로 주의하세요!

```
long long f(long long a, long long b, long long c){
    if (b == 0){ return 1; }
    if (b % 2 == 0){
        long long res = f(a, b/2, c);
        return res*res % c;
    }
    else{
        return f(a, b-1, c) * a % c;
    }
}
```

#연습 문제 도전

<Backtracking>

3 N과 M (1) (BOJ #15649)

다른 N과 M 문제도 풀어보세요! (16214번은 빼고)

1 연산자 끼워넣기 (BOJ #14888)

사칙연산도 알고 보면 어렵습니다!

4 N-Queen (BOJ #9663)

백트래킹에서 빠질 수 없는 문제죠. (가지치기에 대해 기억하시나요?)

2 부분 수열의 합 (BOJ #1182)

수열 전체 대신, 합만 들고 다닌다면?

5 단어 덧셈 (BOJ #4964)

SEND + MORE = MONEY는 해가 유일하답니다

4 좋은수열 (BOJ #2661)

가지치기의 강력함을 느껴봅시다.

#연습 문제 도전

<Divide and Conquer>

3 The Big Dance (BOJ #6012)

Global한 시대에는 영어에도 익숙해져야 하죠.

2 종이의 개수 (BOJ #1780)

4조각이 아니라 9조각으로 나눠야 한다면?

1 쿼드트리 (BOJ #1992)

4조각으로 나누는데, 각 조각별로 괄호를 쳐야 한다면?

1 Z (BOJ #1074)

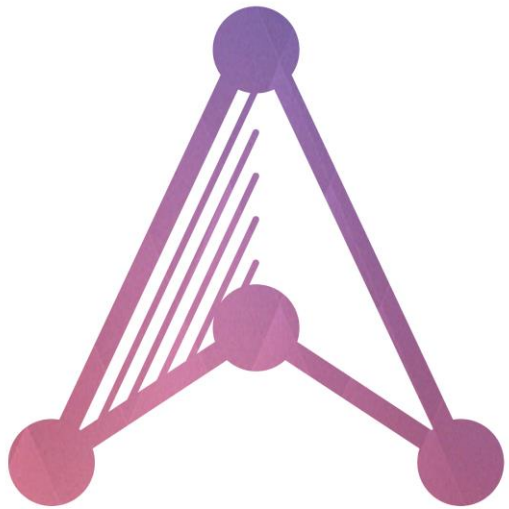
반드시 모든 조각을 탐색해야만 할까요?

5 별 찍기 - 10 (BOJ #2447)

for문 배울 때는 별 찍기가 쉬워 보였죠?

3 Messi Gimossi (BOJ #17297)

메시는 기분이 좋다



ALOHA

The algorithm club.