2023 여름학기 | 동국대학교 SW 역량강화 캠프

4일차. DFS 1

구구단 문제

3 * 7 = 21
곱하는 2개의 수가 정해지면 식을 완성할 수 있다.

곱하는 2개의 수에 대한 완전탐색으로 구구단을 만들 수 있다. > 2중 반복문

```
for(int i=2; i<=9; i++){
    for(int j=1; j<=9; j++){
        System.out.println(i + " * " + j + " = " + (i*j));
    }
}
```

3장으로 하는 블랙잭

3개의 변수에 대한 완전 탐색이 필요하다. > 3중 반복문

```
int Answer = 0;
for (int i = 1; i <= N; i++) {
    for (int j = i + 1; j <= N; j++) { // j는 i+1부터 탐색
        for (int k = j + 1; k <= N; k++) { // k는 j+1부터 탐색
            int Total = arr[i] + arr[j] + arr[k]; // 총 카드의 합은 Total
            if (Total <= M && Total > Answer) { // Total이 M 이하이고, 지금까지 구한 최대값보다 크다면
                Answer = Total; // Answer를 Total로 갱신
            }
        }
    }
}
```

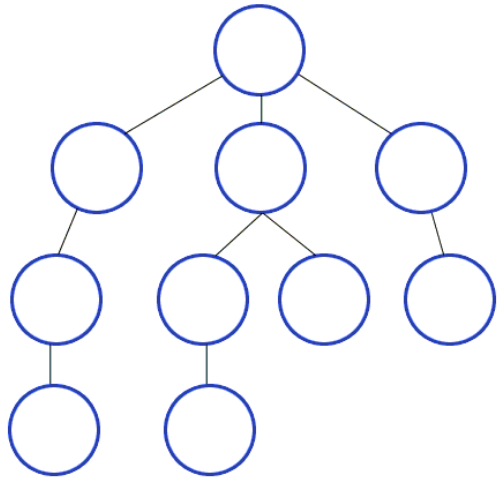
시작하기에 앞서

그렇다면 N개의 변수를 완전탐색하기 위해서는...?

미지수 N에 대하여 N중 반복문을 돌릴 수는 없다.
> 함수 하나가 반복문 하나를 가지고 함수를 N번 호출

● DFS (깊이 우선 탐색)

- ▶ 재귀함수를 통한 탐색(인접한 정점, 인접한 상태)
- ▶ 모든 경우의 수를 탐색하는 완전 탐색
- ▶ 중복 방문을 제한하는 경우 방문 여부를 저장하는 배열을 사용한다.



● 모든 순열 (4154)

문제

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

입력

첫째 줄에 N(1 ≤ N ≤ 8)이 주어진다.

출력

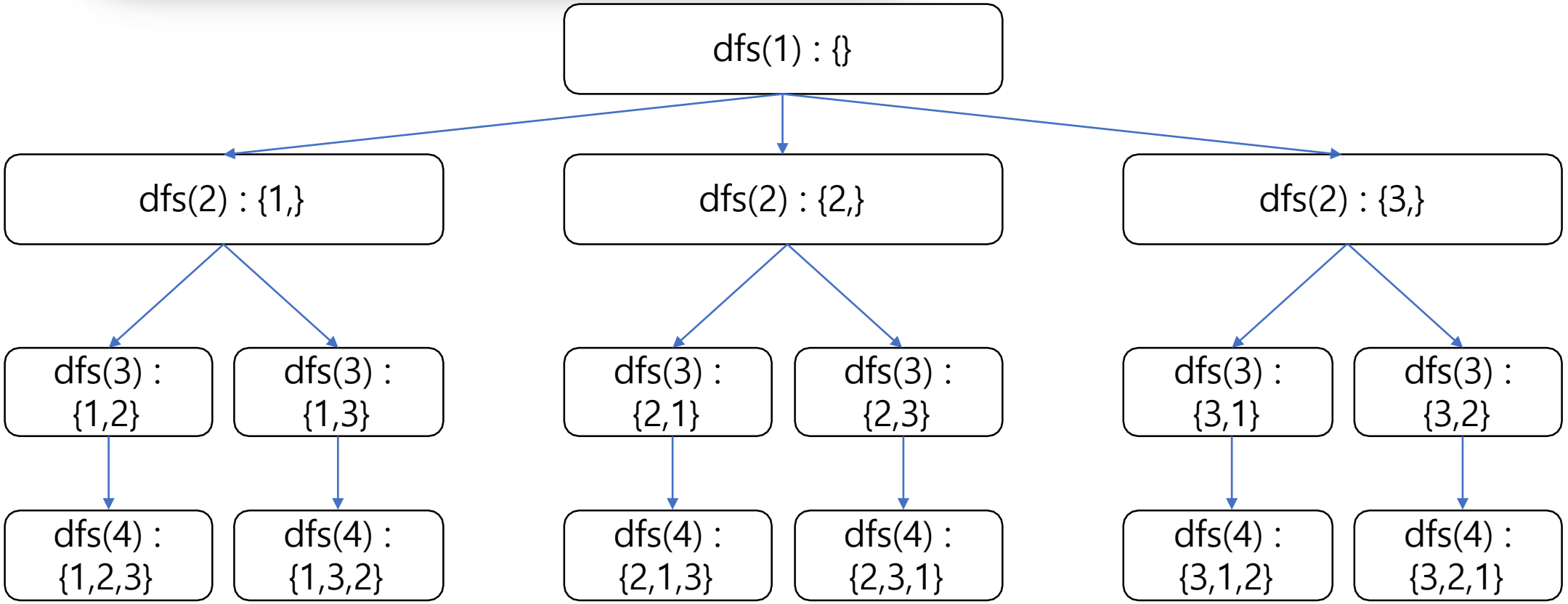
첫째 줄부터 N!개의 줄에 걸쳐서 모든 순열을 사전순으로 출력한다.

▶ 재귀함수를 이용한 풀이

▶ dfs 함수를 다음과 같이 정의하자

dfs(i) = arr[i]를 채우고 dfs(i+1)로 넘어간다. (1≤i ≤N)
dfs(i) = arr 배열을 출력한다. (i == N+1)

문제 해설


dfs(1) : {}
dfs(2) : {1,}
dfs(2) : {2,}
dfs(2) : {3,}
dfs(3) : {1,2}
dfs(3) : {1,3}
dfs(3) : {2,1}
dfs(3) : {2,3}
dfs(3) : {3,1}
dfs(3) : {3,2}
dfs(4) : {1,2,3}
dfs(4) : {1,3,2}
dfs(4) : {2,1,3}
dfs(4) : {2,3,1}
dfs(4) : {3,1,2}
dfs(4) : {3,2,1}

핵심 코드

```java
public static void main(String[] args) throws IOException {
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    N = Integer.parseInt(br.readLine());
    sb = new StringBuilder();

    arr = new int[N + 1];
    used = new boolean[N + 1];

    dfs(1);
    System.out.println(sb.toString());
}

public static void dfs(int x) {
    if(x == N + 1) {
        for(int i=1; i<=N; i++) sb.append(arr[i]).append(' ');
        sb.append('\n');
        return;
    }
    for (int i = 1; i <= N; i++) { // arr[x]에 i를 놓을 수 있는지 판단
        if(used[i] == true) continue;
        arr[x] = i; used[i] = true;
        dfs(x+1);
        arr[x] = 0; used[i] = false; // backtrack
    }
}
```

● N과 M 2 (4143)

문제

자연수 N과 M이 주어졌을 때, 아래 조건을 만족하는 길이가 M인 수열을 모두 구하는 프로그램을 작성하시오.

1. 1부터 N까지 자연수 중에서 중복 없이 M개를 고른 수열
2. 고른 수열은 오름차순이어야 한다.

입력

째 줄에 자연수 N과 M이 주어진다. $(1 \leq M \leq N \leq 8)$

▶ 재귀함수를 이용한 풀이

▶ dfs 함수를 다음과 같이 정의하자

dfs(i) = arr[i]를 채우고 dfs(i+1)로 넘어간다. (1≤i ≤N)
dfs(i) = arr 배열을 출력한다. (i == N+1)

dfs(i)에서 arr[i]를 채울 때, arr[i−1]+1 이상의 수로 채워야 한다.

핵심 코드

```
public static void dfs(int x) {
    if(x == M + 1) {
        for(int i=1; i<=M; i++) sb.append(arr[i]).append(' ');
        sb.append('\n');
        return;
    }
    for (int i = arr[x-1]+1; i <= N; i++) { // arr[x]에 i를 놓을 수 있는지 판단
        arr[x] = i;
        dfs(x+1);
        arr[x] = 0; // backtrack
    }
}
```

문제 해설

▶ 재귀함수를 이용한 풀이

▶ dfs 함수를 다음과 같이 정의하자

dfs(i, x) = x이상의 수로 arr[i]를 채우고 dfs(i+1, arr[i]+1)로 넘어간다. (1≤i ≤N)
dfs(i, x) = arr 배열을 출력한다. (i == N+1)

핵심 코드

```
    dfs(1, 1); // arr[1]은 1 이상의 수로 채운다.
    System.out.println(sb.toString());
}

public static void dfs(int x, int y) { // arr[x]를 y이상의 수로 채우고 다음 단계로
    if(x == M + 1) {
        for(int i=1; i<=M; i++) sb.append(arr[i]).append(' ');
        sb.append('\n');
        return;
    }
    for (int i = y; i <= N; i++) { // arr[x]에 i를 놓을 수 있는지 판단
        arr[x] = i;
        dfs(x+1, i+1);
        arr[x] = 0; // backtrack
    }
}
```

● 사과 나누기 (4752)

문제

민준이는 반 학생들에게 N 개의 무게를 알고 있는 사과를 나누어주고자 한다.
가지고 있는 사과를 적당히 나누어, 남학생들과 여학생들에게 각각 주고자 한다.
이 때, 학생들이 싸우는 것을 막기 위해서 남학생들에게 준 사과의 무게 합과, 여학생들에게 준 사과의 무게 합의 차이가 최소가 되도록 하고 싶다.
민준이를 위해, 가능한 무게 합의 차이의 최소를 출력해주자!

입력

첫번째 줄에는 사과의 수 N이 주어진다. $(1 \leq N \leq 20)$
두번째 줄에는, 각 사과의 무게 p_i 가 주어진다. $(1 \leq p_i \leq 10^9)$

문제 해설

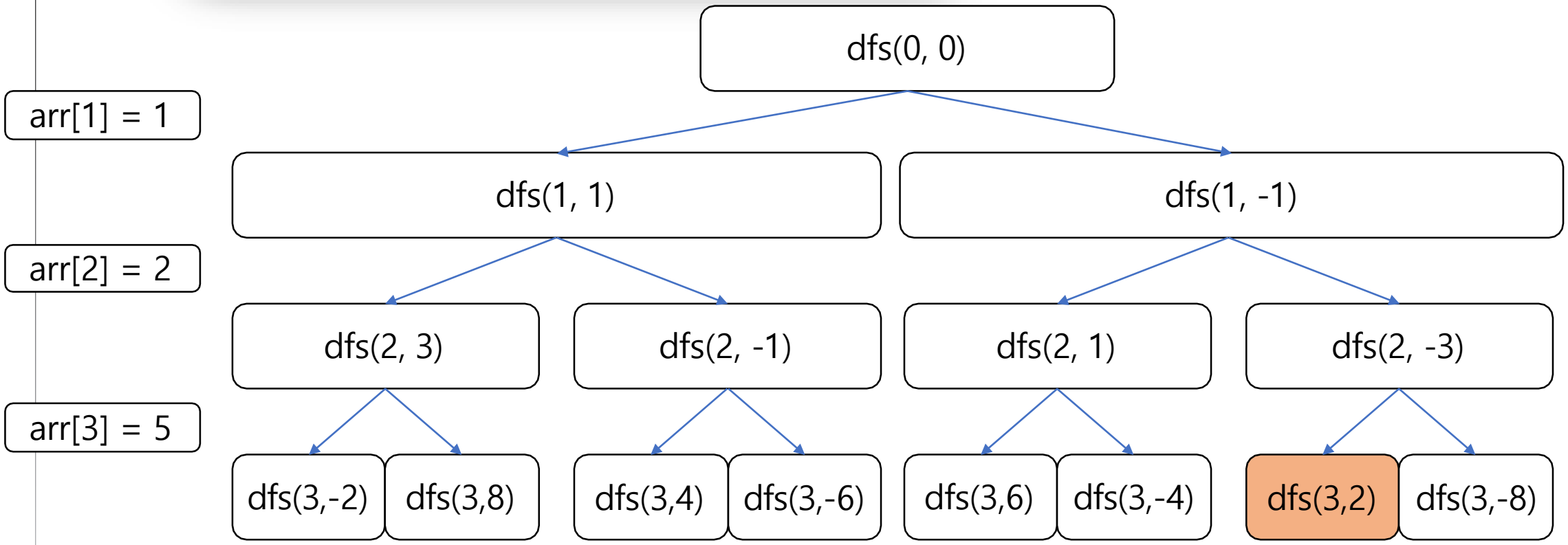
▶ 재귀함수를 이용한 풀이

▶ dfs 함수를 다음과 같이 정의하자

dfs(x, diff) = x번 사과까지 분배해서 두 그룹의 무게 차이가 diff인 상황, x+1번 사과를 분배하고 dfs(x+1, newdiff)로 넘어간다. (0≤x<N)

dfs(x,diff) = 모든 사과를 분배해서 두 그룹의 무게 차이가 diff, diff를 ans에 최소값 갱신(x == N)

문제 해설


dfs(0, 0)
arr[1] = 1
dfs(1, 1)
dfs(1, -1)
arr[2] = 2
dfs(2, 3)
dfs(2, -1)
dfs(2, 1)
dfs(2, -3)
arr[3] = 5
dfs(3,-2)
dfs(3,8)
dfs(3,4)
dfs(3,-6)
dfs(3,6)
dfs(3,-4)
dfs(3,2)
dfs(3,-8)

핵심 코드

```
    dfs(0, 0);

    System.out.println(ans);


public static void dfs(int x, long diff) { // x번까지 사과를 사용해서 두 그룹의 무게 차이가 diff 인 상황
    if (x == N) {
        if(Math.abs(diff) < ans) {
            ans = Math.abs(diff);
        }
        return ;
    }
    dfs(x+1, diff + arr[x+1]);
    dfs(x+1, diff - arr[x+1]);
}
```

● 0을 만들자 (2872)

문제

1부터 N까지의 숫자가 차례대로 적혀 있다.

수들 사이에 "+", "-"를 넣거나 아무것도 넣지 않은 후 계산한 결과가 0이 되도록 만드려고 한다. (아무것도 넣지 않는다면 두 수는 하나의 수로 연결된다)

단, 수의 제일 앞에는 "+", "-"가 있을 수 없다.

만약 N이 7일때는

```
1-2 3+4+5+6+7
```

과 같이 2, 3을 23으로 생각할 수 있다.

입력

양의 정수 N이 첫 줄에 입력된다.(3 <= N <= 9)

문제 해설

▶ 재귀함수를 이용한 풀이

▶ 만약 수를 이어붙힐 수 없었다면?

▶ dfs(x, res, form) = x까지 수를 사용해서 계산값이 res인 상황 (식은 form)

예) dfs(3, 2, “1−2+3”) → dfs(4, 6, “1−2+3+4”) , dfs(4, −2, “1−2+3−4”)

▶ 수를 이어붙히는 것을 변수로 어떻게 표현해야할까?

문제 해설

▶ 계산식을 확정된 부분과 만들고 있는 부분으로 분리하자

▶ 1 + 2 3 − 4 5 + 6 7

▶ dfs(x, res, tmp, form)
= x까지 숫자들을 사용해서 확정된 계산 결과 = res, 만들고 있는 수 = tmp, 식이 form인 상황

예)
dfs(5, 24, −45, "1+23−45")

→ dfs(6, 24, −456, "1+23−456")
→ dfs(6, −21, 6, "1+23−45+6")
→ dfs(6, −21, −6, "1+23−45−6")

▶ 초기 상태 = dfs(1, 0, 1, “1”)

핵심 코드

```
public static void dfs(int x, int res, int tmp, String form) {
    // x번까지 숫자들을 사용해서 확정된 계산 결과 = res, 만들고 있는 수 = tmp, 식이 form인 상황
    // 1 + 2 3 - 4 5 => dfs(5, 24, -45, "1+23-45")
    // 6이 이어붙혀진다면 1 + 2 3 - 4 5 6 => dfs(6, 24, -456, "1+23-456")
    // 6이 + 로 붙는다면 1 + 2 3 - 4 5 + 6 => dfs(6, -21, 6, "1+23-45+6")
    // 6이 - 로 붙는다면 1 + 2 3 - 4 5 - 6 => dfs(6, -21, -6, "1+23-45-6")
    if (x == N) {
        int result = res + tmp;
        if(result == 0) sb.append(form).append('\n');
        return ;
    }
    int newtmp;
    if(tmp > 0) newtmp = tmp*10 + (x+1); // 23 -> 234
    else newtmp = tmp*10 - (x+1); // -23 -> -234

    dfs(x+1, res, newtmp, form + ' ' + (char)((x+1) + '0'));
    dfs(x+1, res+tmp, x+1, form + '+' + (char)((x+1) + '0'));
    dfs(x+1, res+tmp, -(x+1), form + '-' + (char)((x+1) + '0'));
}
```