마무리

- 혹시...
  - 함수 모르는 사람?
  - 비트 연산 모르는 사람?
  - 수학적 귀납법 모르는 사람?
- 과제 관련 질문은 슬랙 #qna 에 올리면 됩니다.
  - 이거 어떻게 풀어요? (x)
  - https://www.acmicpc.net/blog/view/45 읽어보세요.
    - 원하는 답변을 얻기 위해 "잘" 질문하는 방법
  - https://www.acmicpc.net/blog/view/55 이것도 읽어보세요.
    - BOJ 작동 원리
  - https://www.acmicpc.net/blog/view/70 이것도 읽어보세요.
    - 자주 틀리는 이유

2022-1학기 스터디 #2

나정휘

https://justiceHui.github.io/

목차

- 재귀함수
- 순열
- 비트마스크

재귀함수

재귀함수

- 재귀함수: 자기 자신을 호출하는 함수
  - ex. f(n) = f(n-1) + f(n-2)
  - 점화식 계산, 완전 탐색, 반복문으로 반복하기 힘든 작업, ...

  - 어려움
  - 평소와 다른 방식으로 생각해야 함
    - 귀납적 사고
    - 익숙해지는데 2년 걸림

재귀함수 – 예시 1

- 배열 A의 모든 원소의 합을 구하는 작업
  - f(l, r) : A[l] + A[l+1] + … + A[r]을 구하는 함수
  - f(0, N-1)을 구해야 함

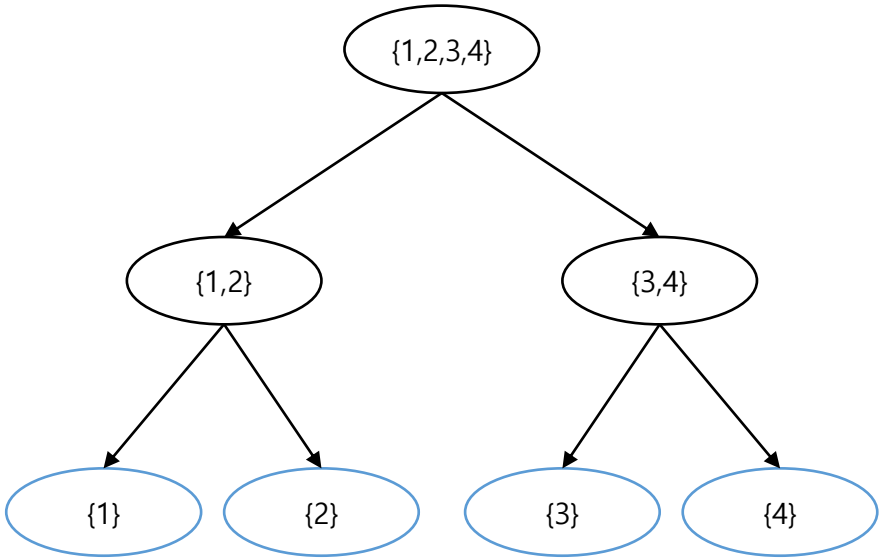
  - 일을 혼자 하는 건 어려우니까 직원 2명을 고용해서
  - 배열을 반으로 잘라서 나눠주면?
  - f(l, r) = f(l, (l+r)/2) + f((l+r)/2+1, r)

재귀함수 – 예시 1

- 배열 A의 모든 원소의 합을 구하는 작업
  - f(l, r) : A[l] + A[l+1] + … + A[r]을 구하는 함수

  - 일을 혼자 하는 건 어려우니까 직원 2명을 고용해서
  - 배열을 반으로 잘라서 나눠주면?
  - f(l, r) = f(l, (l+r)/2) + f((l+r)/2+1, r)

  - 재귀 호출 과정을 살펴보면…

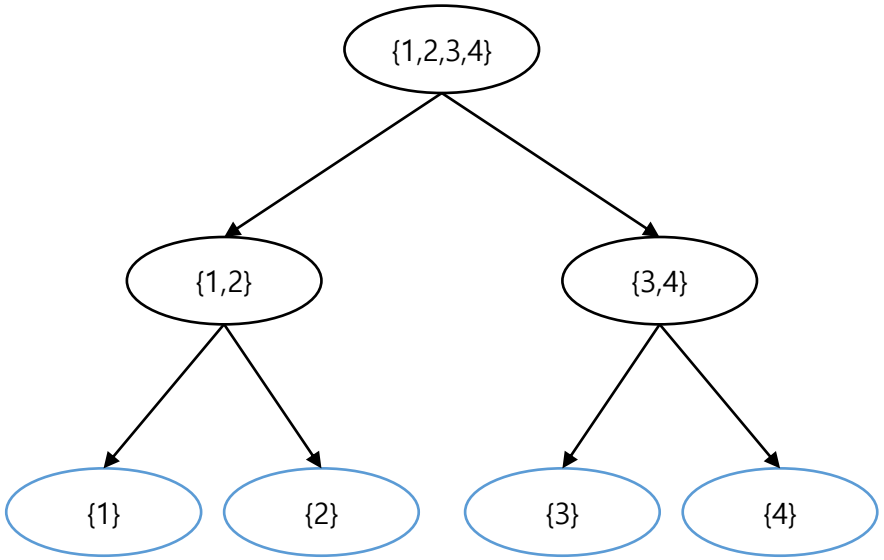


재귀함수 – 예시 1

- 배열 A의 모든 원소의 합을 구하는 작업
  - f(l, r) : A[l] + A[l+1] + … + A[r]을 구하는 함수

  - 일을 혼자 하는 건 어려우니까 직원 2명을 고용해서
  - 배열을 반으로 잘라서 나눠주면?
  - f(l, r) = f(l, (l+r)/2) + f((l+r)/2+1, r)

  - 재귀 호출 과정을 살펴보면…
    - 이렇게 하지 마세요
  - 함수 f가 올바른 결과를 반환한다고 “믿고”
  - f(l, r)에서는 f(l, (l+r)/2) + f((l+r)/2+1, r)을 반환하면 됨
    - 수학적 귀납법



재귀함수 – 예시 2

- 배열 A를 정렬하는 작업
  - f(l, r) = A[l..r]을 정렬된 상태로 만드는 함수
  - l = r이면 이미 정렬된 상태 (종료 조건)

  - f(l, m)과 f(m+1, r)을 호출
    - m = (l+r)/2

  - A[l..m]과 A[m+1..r]이 정렬되어 있다고 "믿고"
  - 정렬된 두 배열을 잘 합치면 됨

재귀함수

- 재귀호출
  - 자신과 동일한 일을 하는 부하 직원에게 작업을 요청하고 그 결과물을 돌려받는 과정
  - 부하 직원의 결과물이 항상 올바르다고 믿고 작업을 수행하면 됨
  - = 수학적 귀납법

  - 학교에서는 이런 거 안 알려줌 ㅎㅎ;;

재귀함수

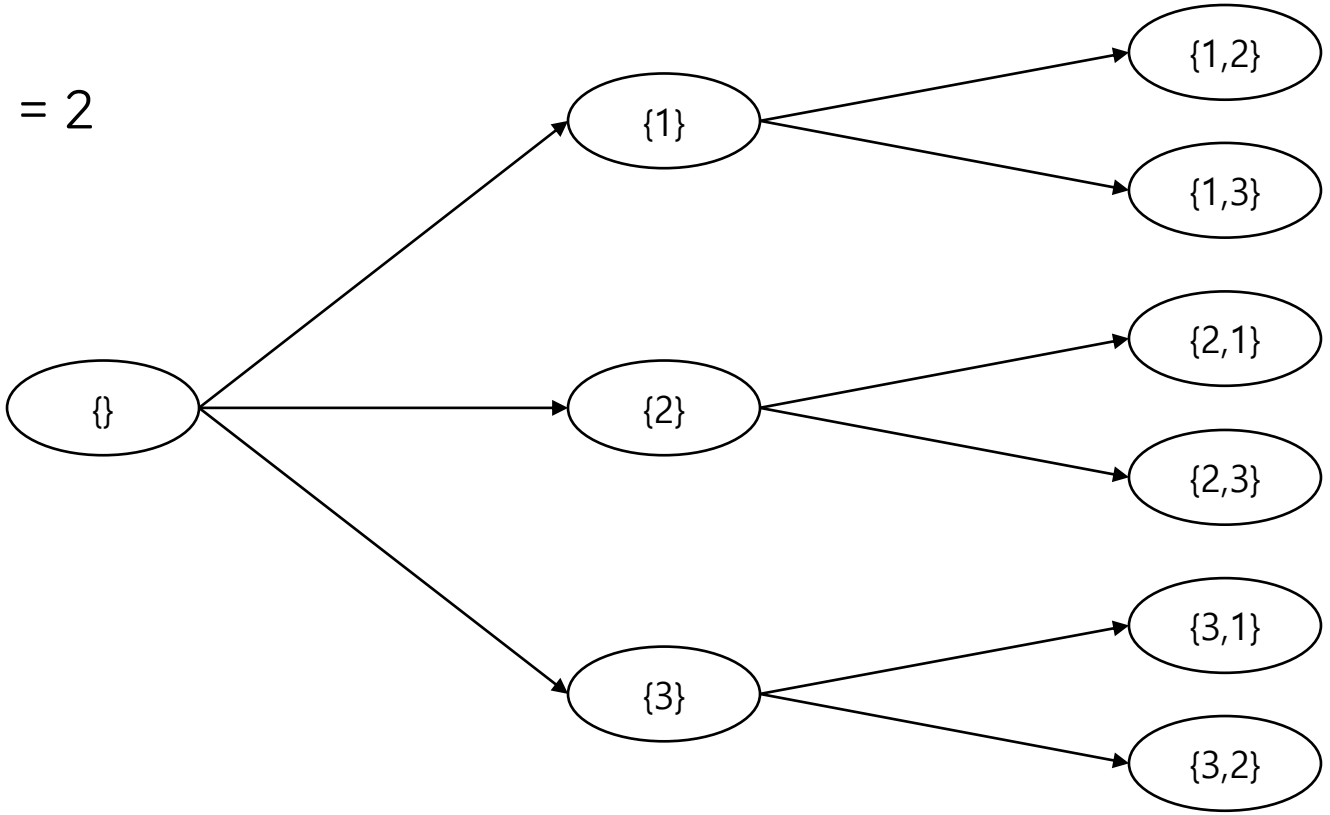
- 주의사항
  - 재귀 호출을 하지 않는 케이스(종료 조건)이 있어야 함
    - 없으면 프로그램이 종료되지 않음
  - 재귀 호출 과정에 사이클이 없어야 함
    - f(a) → f(b) → f(c) → … → f(a)
    - 프로그램이 종료되지 않음
  - 종료 조건에 가까워지는 방향으로 설계
    - 앞에서 본 예시는 l = r이 종료 조건, r – l이 작아지는 방향으로 재귀 호출

질문?

재귀함수 – 예제 3

- BOJ 15649 N과 M (1)
  - 1부터 N까지 자연수 중에서 중복 없이 M개를 고른 수열을 모두 출력
  - 수열은 사전 순으로 증가하는 순서로 출력

  - ex. N = 3, M = 2



재귀함수 – 예제 3

- f(지금까지 선택한 수의 개수, 지금까지 만든 배열)
  - f(choose, arr)
  - choose = M이면 배열 출력하고 종료
  - arr에 없는 수를 배열의 맨 뒤에 추가하고 재귀 호출
    - arr[choose] = num;
    - f(choose+1, arr)

```c
#include <stdio.h>

void f(int n, int m, int choose, int arr[]){
    if(choose == m){ // 종료 조건
        for(int i=0; i<m; i++){
            printf("%d ", arr[i]);
        }
        printf("\n");
        return;
    }

    for(int i=1; i<=n; i++){ // n개의 수를 한 번씩 시도
        int exist = 0; // 이미 만든 배열에 i가 있으면 exist = 1
        for(int j=0; j<choose; j++){
            if(arr[j] == i) exist = 1;
        }
        if(exist == 0){
            arr[choose] = i; // i 선택
            f(n, m, choose+1, arr); // 선택한 수의 개수 1 증가
            arr[choose] = 0; // i 넣은 거 원상 복구
        }
    }
}

int main(){
    int N, M, A[8];
    scanf("%d %d", &N, &M);
    f(N, M, 0, A);
}
```

질문?

재귀함수 – 예제 3

- exist 변수의 값을 매번 계산하는 것을 비효율적
  - use[num] = num을 사용했으면 1, 안 했으면 0
    - arr[choose] = num; use[num] = 1;
    - f(choose+1, arr, use);
    - arr[choose] = 0; use[num] = 0;

```c
#include <stdio.h>

void f(int n, int m, int choose, int arr[], int use[]){
    if(choose == m){ // 종료 조건
        for(int i=0; i<m; i++){
            printf("%d ", arr[i]);
        }
        printf("\n");
        return;
    }

    for(int i=1; i<=n; i++){ // n개의 수를 한 번씩 시도
        if(use[i] == 0){ // 아직 i를 사용하지 않았다면
            arr[choose] = i; // i 선택
            use[i] = 1;
            f(n, m, choose+1, arr, use); // 선택한 수의 개수 1 증가
            arr[choose] = 0; // i 넣은 거 원상 복구
            use[i] = 0;
        }
    }
}

int main(){
    int N, M, A[8], U[9] = {0}; // U는 0으로 초기화
    scanf("%d %d", &N, &M);
    f(N, M, 0, A, U);
}
```

재귀함수 – 예제 3

- 항상 5개의 인자를 모두 넘겨줘야 할까?
  - n, m은 변하지 않음
  - arr, use도 포인터를 넘겨주기 때문에 변하지 않음
  - 항상 동일한 값은 전역 변수로 빼도 됨

```c
#include <stdio.h>

int N, M, A[8], U[9]; // 전역 변수는 0으로 초기화

void f(int choose){
    if(choose == M){
        for(int i=0; i<M; i++) printf("%d ", A[i]);
        printf("\n");
        return;
    }

    for(int i=1; i<=N; i++){
        if(!U[i]){
            A[choose] = i; U[i] = 1;
            f(choose+1);
            A[choose] = 0; U[i] = 0;
        }
    }
}

int main(){
    scanf("%d %d", &N, &M);
    f(0);
}
```

재귀함수 – 예제 4

- BOJ 15650 N과 M (2)
  - 1부터 N까지 자연수 중에서 중복 없이 M개를 오름차순으로 고른 수열을 모두 출력
  - 수열은 사전 순으로 증가하는 순서로 출력

  - 현재 배열에 맨 뒤에 있는 수보다 큰 수를 선택해야 함

```
int N, M, A[8];

void f(int choose){
    if(choose == M){
        for(int i=0; i<M; i++) printf("%d ", A[i]);
        printf("\n");
        return;
    }

    for(int i=1; i<=N; i++){
        if(choose == 0 || A[choose-1] < i){
            A[choose] = i;
            f(choose+1);
        }
    }
}
```

질문?

재귀함수 – 예제 5

- BOJ 11729 하노이 탑 이동 순서
  - 3개의 기둥이 있고 첫 번째 기둥에 서로 다른 크기의 원판 N개 있음
  - 어떤 원판 위에는 자신보다 작은 원판만 올라올 수 있음
  - 원판을 최소한으로 이동해서 첫 번째 기둥에 있는 모든 원판을 세 번째 기둥으로 옮겨야 함

재귀함수 – 예제 5

- 최적 전략에 대해 생각해보자
  - 첫 번째 기둥에 있는 N개의 원판을 세 번째 기둥으로 옮겨야 함
  - 가장 큰 원판부터 차례대로 세 번째 기둥으로 옮김

  - 가장 큰 원판을 어떻게 꺼내지?
  - 위에 있는 N-1개를 잠시 두 번째 기둥으로 옮기면 됨

  - N-1개를 첫 번째 기둥에서 두 번째 기둥으로 옮기고
  - N번째 원판을 첫 번째 기둥에서 세 번째 기둥으로 옮기고
  - N-1개를 두 번째 기둥에서 세 번째 기둥으로 옮기면 됨

재귀함수 – 예제 5

- f(n, a, b, c) : n개의 원판을 a에서 c로 옮기는 과정을 구하는 함수
  - n-1개를 a에서 b로 옮김 : f(n-1, a, c, b);
  - n번째 원판을 a에서 c로 옮김 : f(1, a, b, c);
  - n-1개를 b에서 c로 옮김 : f(n-1, b, a, c);

  - 종료 조건 : n = 1, a c 출력하고 종료

  - 이동 횟수
    - T(1) = 1
    - T(n) = 2*T(n-1) + 1
    - T(n) = 2^n-1

```c
#include <stdio.h>

void f(int n, int a, int b, int c){
    if(n == 1){
        printf("%d %d\n", a, c);
        return;
    }
    f(n-1, a, c, b);
    f(1, a, b, c);
    f(n-1, b, a, c);
}

int main(){
    int N;
    scanf("%d", &N);
    printf("%d\n", (1<<N) - 1);
    f(N, 1, 2, 3);
}
```

질문?

순열

순열

- 순열: N개의 원소를 중복없이 나열하는 것
  - 1 2 3 / 1 3 2 / 2 1 3 / 2 3 1 / 3 1 2 / 3 2 1
  - 1 1 2 2 / 1 2 1 2 / 1 2 2 1 / 2 1 1 2 / 2 1 2 1 / 2 2 1 1

  - 중복 원소가 없으면 N!가지
    - 1 2 3을 나열하는 경우의 수 = 3! = 6
  - 중복 원소가 있으면 N! / (1개수)! / (2개수)! / ...
    - 1 1 2 2를 나열하는 경우의 수 = 4! / 2! / 2! = 6

  - 재귀 함수를 이용해 모든 순열을 확인할 수 있음
    - BOJ 15649 N과 M (1)

순열

- std::next_permutation(begin, end)
  - 사전 순으로 바로 다음 순열을 구하는 함수
  - 다음 순열이 없으면 false, 있으면 true 반환

  - 예시
    - int arr[5] = {0, 1, 2, 3, 4}
    - std::next_permutation(arr, arr+5)
    - arr = {0, 1, 2, 4, 3}으로 바뀜

    - int arr[4] = {1, 1, 2, 2}
    - std::next_permutation(arr, arr+4)
    - arr = {1, 2, 1, 2}로 바뀜
    - 중복 원소 있어도 상관 없음

순열 – 예시 1

- BOJ 10819 차이를 최대로
  - N개의 정수로 구성된 배열 A가 주어짐
  - 원소의 순서를 적당히 바꿔서 아래 식의 값을 최대화하는 문제
    - | A[0] – A[1] | + | A[1] – A[2] | + … + | A[N-2] – A[N-1] |
  - 모든 순열을 확인해보면 됨
    - 주의) 처음에 A를 정렬해야 모든 순열을 확인할 수 있음

```cpp
#include <stdio.h>
#include <algorithm>
using namespace std;

int N, A[8];

int main(){
    scanf("%d", &N);
    for(int i=0; i<N; i++) scanf("%d", &A[i]);
    sort(A, A+N);

    int mx = 0;
    do{
        int now = 0;
        for(int i=1; i<N; i++){
            if(A[i-1] > A[i]) now += A[i-1] - A[i];
            else now += A[i] - A[i-1];
        }
        if(now > mx) mx = now;
    }while(next_permutation(A, A+N));
    printf("%d", mx);
}
```

순열 – 예시 2

- BOJ 17359 전구 길만 걷자
  - N개의 이진 문자열이 주어짐
  - N개를 잘 배열해서 “01” 또는 “10”이 등장하는 횟수를 최소화해야 함

  - 원소들을 배열하는 가능한 모든 순서를 확인하면 되므로
  - 모든 순열을 시도하면 됨

질문?

비트마스크

부분 집합

- 배열 A의 모든 부분 집합을 구하는 방법
  - 재귀함수
  - 비트마스크

- 비트 연산
  - and: 0011 and 1010 = 0010
  - or : 0011 or 1010 = 1011
  - xor : 0011 xor 1010 = 1001

  - shl : 101 << 2 = 10100
  - shr : 101 >> 1 = 10

비트마스크

- 부분 집합
  - (각 원소가 들어가는 경우 / 들어가지 않는 경우)^{n가지} → 2^n가지
  - 각 원소의 사용 여부를 비트로 표현
    - A[i]를 사용하면 2^i를 나타내는 비트를 1로 표시

  - ex. A = {0, 1, 2, 3}
  - {} : 0000
  - {0} : 0001
  - {1, 3} : 1010
  - {0, 1, 2, 3} : 1111

  - 크기가 n인 집합의 부분 집합은 0 이상 2^n 미만의 정수와 일대일 대응

비트마스크 – 예시

- BOJ 1182 부분수열의 합
  - 길이가 N인 수열 A와 정수 S가 주어짐
  - A의 부분 수열 중 원소의 합이 S가 되는 경우의 수

- 공집합을 제외한 모든 부분 집합을 보면 됨
  - 공집합 → 0

```c
#include <stdio.h>

int N, S, A[20];

int main(){
    scanf("%d %d", &N, &S);
    for(int i=0; i<N; i++) scanf("%d", &A[i]);

    int cnt = 0;
    for(int bit=1; bit<(1<<N); bit++){
        int sum = 0;
        for(int i=0; i<N; i++){
            if(bit & (1 << i)) sum += A[i];
        }
        if(sum == S) cnt++;
    }
    printf("%d", cnt);
}
```

질문?

과제

- 필수 과제
  - N과 M (1), N과 M (2)
  - 별 찍기 - 10
  - 하노이탑 이동 순서
  - 차이를 최대로
  - 전구 길만 걷자
  - 부분 수열의 합
- 심화 과제
  - Z
  - 쿼드 트리
  - 물약 구매
  - 랭퍼든 수열쟁이야!!