PULSE 2023 Fall

완전탐색 – brute force / DFS / BFS

목차

- Problem Solving / Competitive Programming
  - ❖ 문제 이해

- Brute Force
  - ❖ Brute Force 기법 – 단순 brute force, 순열, 재귀
  - ❖ What is Graph?
  - ❖ Depth-First Search
  - ❖ Breadth-Frist Search

PS / CP

PS / CP

- Problem Solving
  - 주어진 문제를 해결하는 과정
  - 문제를 푼다는 과정 그 자체에 포커스를 두는 것
- Competitive Programming
  - 경쟁적 프로그래밍
  - 정해진 시간 내 문제를 빠르고 정확하게 해결하는 것

- 고려할 점
  - 사용 언어, 시간 제한(시간 복잡도), 메모리 제한(공간 복잡도)
  - 문제 해결 능력, 구현 능력, 배경 지식(알고리즘, 자료구조)

문제 이해

- 문제를 접하게 되면, 가장 우선적으로 수행되어야 함!

  - 제한 시간
  - 메모리 제한
  - 사용할 알고리즘
  - 사용할 자료구조

문제 이해

- 문제 이해는 매우 중요!
  - 문제를 제대로 파악하지 못한 경우
    - 이해하지 않고 진행 시 막힐 때마다 고민하는 시간이 증가함(더 큰 손해 발생)
    - 본인이 잘못된 방향으로 코딩하는 것을 인지하기 어려움
    - 작성한 코드를 포기해야 하는 상황 발생 가능성 ↑
  - 이러한 상황을 피하려면?

문제 이해

- 문제 이해 방법
  - 이 문제가 원하는 바는 무엇인가?
  - 어떤 제약 사항이 있는가?
  - 제공되는 입력 데이터는 어떤 형식인가?
  - 어떤 결과를 출력해야 하는가?
  - 예제에 대한 추가적인 설명이 있는가?

Brute Force

Brute Force

- 무식하게 해결한다 == Brute Force
- 모든 경우의 수를 나열하면서 답을 찾음.(**완전 탐색**)

- ex) 학생 10명을 한 줄로 세우려고 한다. 사이가 안 좋은 학생들이 있을 때, 떨어뜨려서 세우는 방법은 몇 가지인가?
  - ➢ 줄 세우는 모든 경우의 수를 나열하면서 확인
  - ➢ 10명이면? 10!, 360만 정도의 경우의 수

Brute Force 기법 – 단순 brute force

- 반복 / 조건문을 통해 가능한 모든 방법을 단순히 찾는 경우

- 각 테스트 케이스마다 a, b, c의 범위를 차례대로 모두 대입하여 진행하면 됨.

단순한 문제 (Small)

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
1 초	1024 MB	2260	1785	1560	80.495%

문제

세 양의 정수 a, b, c가 주어질 때, 다음 조건을 만족하는 정수 쌍 (x, y, z)의 개수를 구하시오.

- $1 \le x \le a$
- $1 \le y \le b$
- $1 \le z \le c$
- $(x \bmod y) = (y \bmod z) = (z \bmod x)$

$(A \bmod B)$는 A를 B로 나눈 나머지를 의미한다.

입력

첫째 줄에 테스트 케이스의 수 T가 주어진다. $(1 \le T \le 100)$

다음 T개의 각 줄에는 세 정수 a, b, c가 공백으로 구분되어 주어진다. $(1 \le a, b, c \le 60)$

출력

한 줄에 하나씩 정답을 출력한다.

예제 입력 1 복사

```
2
1 2 3
3 2 4
```

예제 출력 1 복사

```
1
2
```

첫 번째 예시에서 조건을 만족하는 쌍은 $(1, 1, 1)$이다.

두 번째 예시에서 조건을 만족하는 쌍은 $(1, 1, 1)$과 $(2, 2, 2)$이다.

Brute Force 기법 – 순열

- 순열
  - 임의의 수열이 있을 때, 다른 순서로 연산하는 모든 경우의 수
  - **Python, C++**의 경우 관련 라이브러리 존재 (Java는 직접 구현 필요)
  - 순열의 시간 복잡도는 N개의 배열에서 O(N!)이므로 시간 제한을 고려해야 함

Brute Force 기법 – 순열

Ex) {1, 2, 3}의 모든 순열을 출력하는 코드 (라이브러리 활용)

Python

```python
import itertools

arr = [ 1, 2, 3 ]
nPr = itertools.permutations(arr, 3)
print(list(nPr))
```

C++

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

int main() {
    vector<int> v{ 1, 2, 3};
    sort(v.begin(), v.end()); // sort 반드시 필요!!
    do {
        for (auto it = v.begin(); it != v.end(); ++it)
            cout << *it << ' ';
        cout << endl;
    } while (next_permutation(v.begin(), v.end()));
}
```

Brute Force 기법 – 재귀

- 자신이 수행할 작업을 유사한 형태의 여러 조각으로 쪼갠 뒤 그 중 한 조각을 수행하고, 나머지를 자기 자신이 호출해 실행하는 함수

- For/while – 같은 코드를 반복하여 실행하는 명령어
- 재귀 함수 – 자기자신을 반복하여 호출하는 함수

Brute Force 기법 – 재귀

- 1부터 정수 n까지의 합을 구하는 함수 (단, 0 < n < 2^31)

단순 반복문

```
int sum(int n){
    int ret = 0;
    for(int i = 1; i <= n; i++){
        ret += i;
    }
    return ret;
}
```

재귀 함수

```
int recursiveSum(int n){
    if(n == 1) return 1;
    return n + recursiveSum(n-1);
}
```

Brute Force 기법 – 재귀

- 재귀 호출을 이용한 완전 탐색에서 **모든 입력이 기저 사례를 이용해야함**!

```
int recursiveSum(int n){
    if(n == 1) return 1;
    return n + recursiveSum(n-1);
}
```

- 기저 사례(=base case)
  - 더 이상 쪼개지지 않는 가장 작은 작업

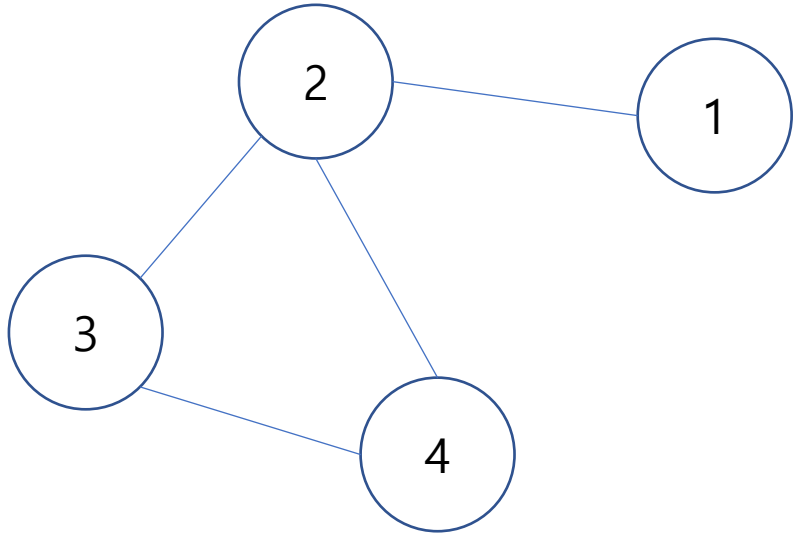
Brute Force

- 가능한 모든 경우의 수를 탐색
  - 요구조건에 충족되는 결과만을 가져옴
  - 예외 없이 100%로 정답만을 출력
  - **괴장히 느리게 동작할 수 있음 -> 문제 파악이 중요한 이유!!!**

1) 크기의 입력을 가정하고 답의 개수를 계산하여 제한 시간 안에 해결 가능한지 판단
2) 가능한 모든 답의 후보를 만드는 과정을 여러 개의 선택으로 분할
3) 그 중 하나의 조각을 선택해 답의 일부를 만들고, 나머지 답을 재귀 호출을 통해 완성
4) 조각이 하나밖에 남지 않은 경우, 이것을 기저 사례로 선택해 처리

Graph

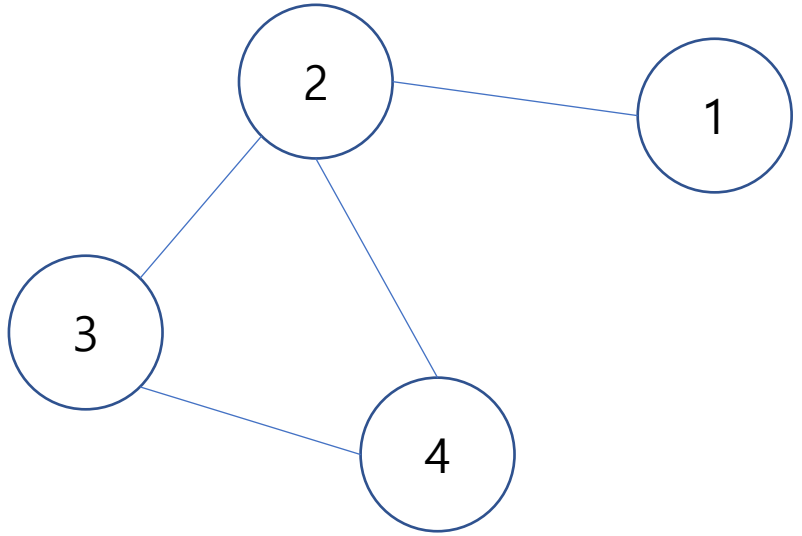
- 정점 집합과 두 정점을 연결하는 간선 집합으로 구성된 구조
  - G = (V, E)
  - V: 정점들의 집합
  - E ⊆ { {x, y} | x, y ∈ V }: 간선들의 집합



- 그래프 종류
  - 무향 그래프 (Undirected Graph): 간선에 방향이 없는 그래프
  - 유향 그래프 (Directed Graph): 간선에 방향이 있는 그래프
  - 가중치 그래프 (Weighted Graph): 간선에 가중치가 있는 그래프
  - 연결 그래프 (Connected Graph): 모든 정점이 직/간접적으로 연결되어 있는 그래프

Graph 표현 방식

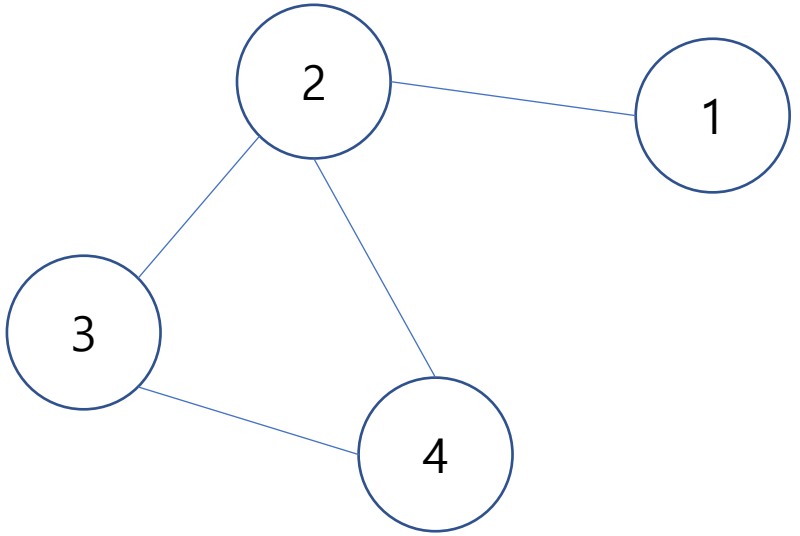
- 인접 행렬
  - |V| * |V| 크기의 2차원 배열 G
  - u에서 v로 가는 간선이 있으면 G[u][v] = 1
    - 무향 그래프? G[u][v] = G[v][u]
    - 가중치 그래프? G[u][v] = (가중치)
  - 공간 복잡도: O(|V|^2)

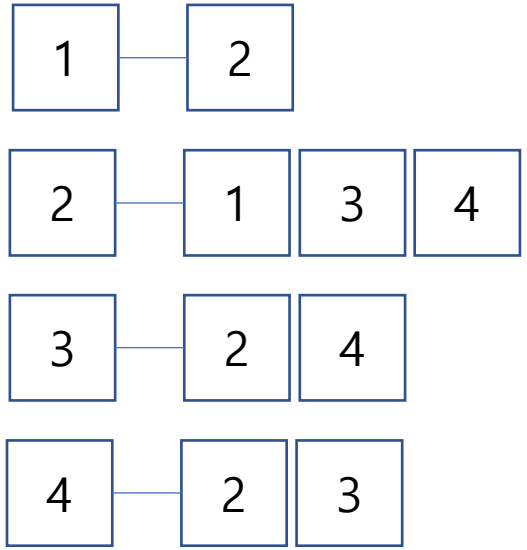


0	1	0	0
1	0	1	1
0	1	0	1
0	1	1	0

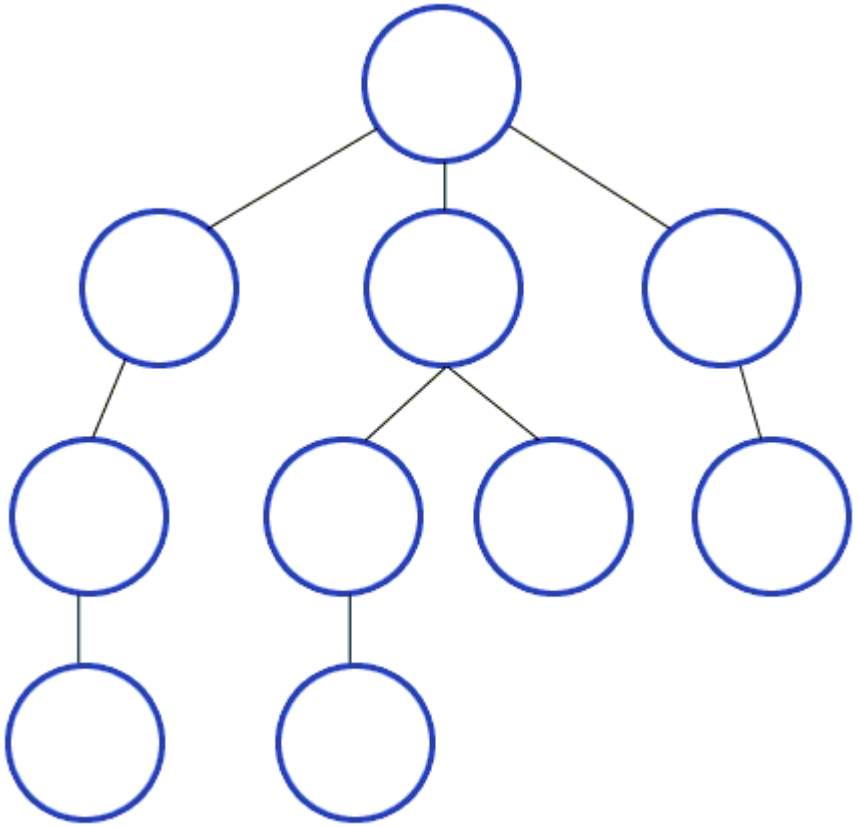
Graph 표현 방식

- 인접 리스트
  - 연결 리스트 |V|개 관리
  - U번째 리스트는 u에서 나가는 간선을 관리함
  - 공간 복잡도: O(|V| + |E|)
    - O(|V| + sum deg(v)) 인데 sum deg(v) = 2E



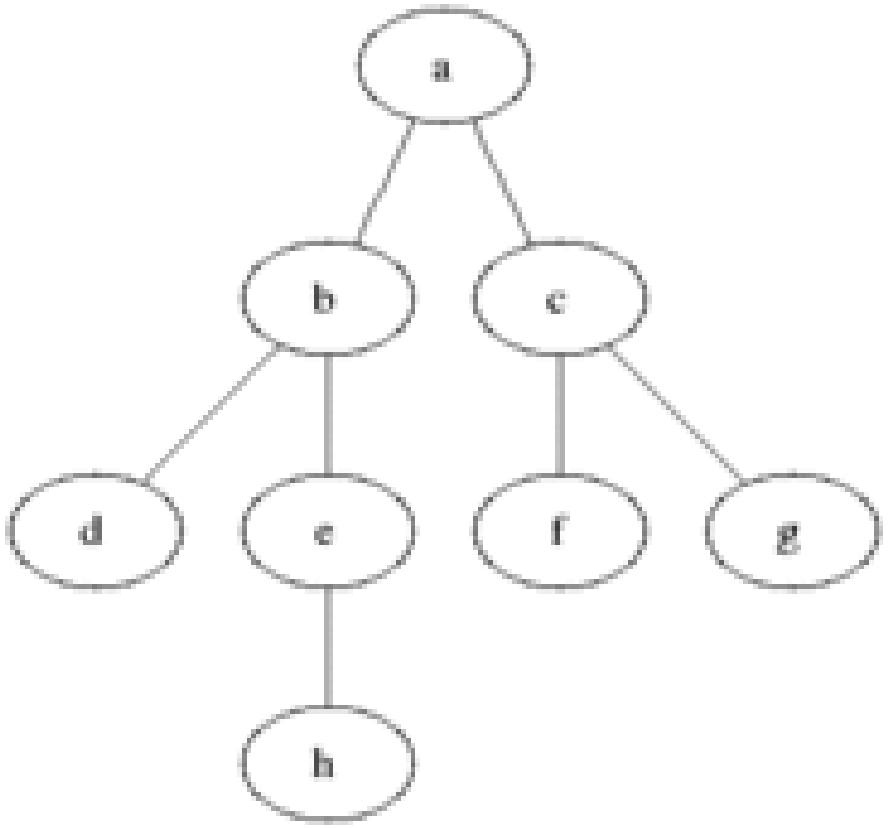


Depth-First Search(DFS)



https://ko.wikipedia.org/wiki/%EA%B9%8A%EC%9D%B4_%EC%9A%B0%EC%84%A0_%ED%83%90%EC%83%89

Breadth-First Search(BFS)



https://ko.wikipedia.org/wiki/%EB%84%88%EB%B9%84_%EC%9A%B0%EC%84%A0_%ED%83%90%EC%83%89