



SW 역량 테스트 대비반 - 4회차

2019년 05월 16일

지난 수업 정리

- **algorithm 헤더**

- sort
- count
- find
- copy
- swap
- transform
- min/max

- **cmath**

- abs

- **cctype**

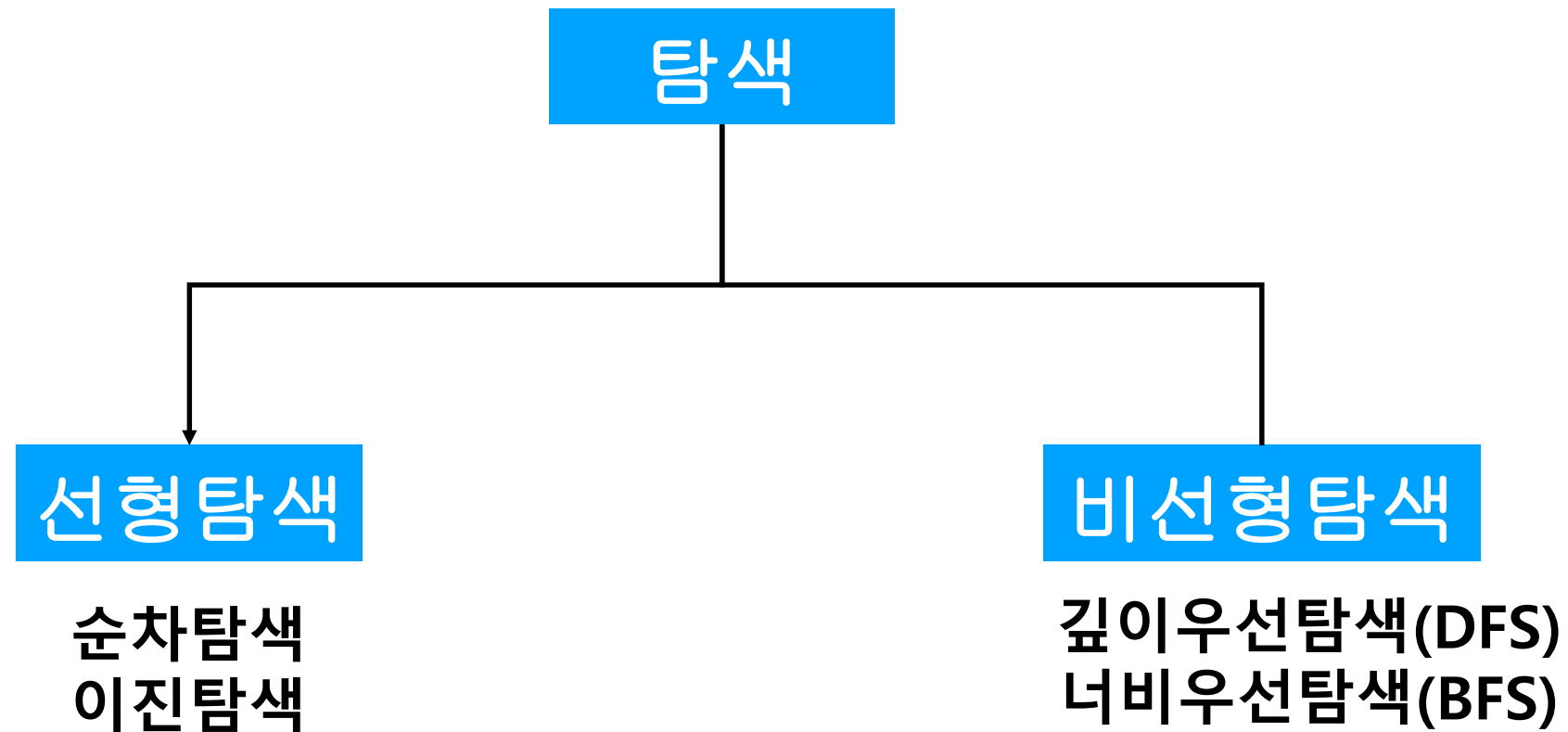
- toupper, tolower

Homework1 : <https://onlinegdb.com/rJCl8N-34>

오늘 할 내용

구분	상세 내용	교육 날짜
1주차	운영계획 및 SW 역량 레벨 테스트	04/04
2주차	C/C++ 기초 문법 강의 및 실습 1	04/11
3주차	C/C++ 기초 문법 강의 및 실습 2	04/18
4주차	기초 자료구조 강의 및 실습	04/25
5주차	정렬 알고리즘 강의 및 실습	05/09
6주차	탐색 알고리즘 강의 및 실습 1	05/16
7주차	<div>탐색알고리즘</div> <div>(선형탐색(순차/이진탐색),</div> <div>비선형탐색(DFS, BFS))</div> <div>관련 문제 풀이</div> <div>(레벨에 따라 2~3개 분반 필요할 수 있음)</div>	05/23
8주차		05/30
9주차		06/07
10주차		06/13
11주차		06/20
12주차		06/27
13주차 ~		7월 부터는 주 2회 (화/목)

탐색 알고리즘



순차탐색

1 순차 탐색

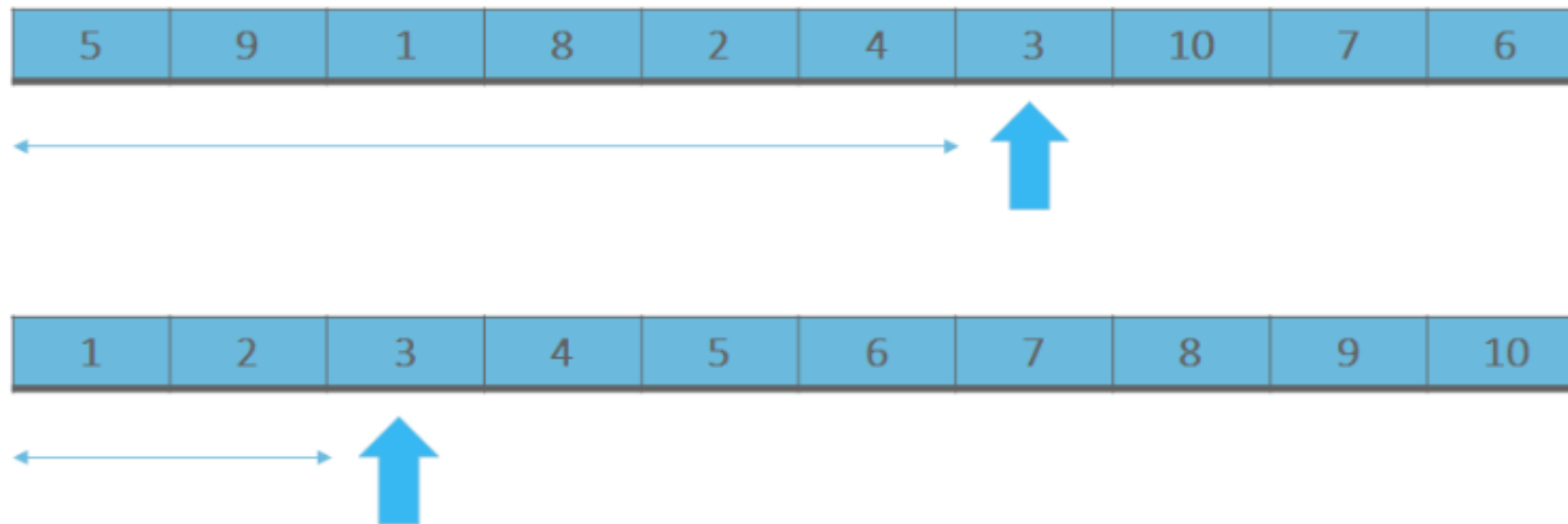
처음부터 원하는 데이터가 나올 때까지

순서대로 찾는 방식

순차 탐색

데이터의 정렬 여부와 상관없이 사용가능

e.g. 3을 찾을 때



순차탐색

문제

최댓값(S)

9개의 서로 다른 자연수가 주어질 때, 이들 중 최댓값을 찾고 그 값이 몇 번째 수인지를 구하는 프로그램을 작성하시오.

예를 들어, 서로 다른 9개의 자연수가 각각

3, 29, 38, 12, 57, 74, 40, 85, 61

라면, 이 중 최댓값은 85이고, 이 값은 8번째 수이다.

입력

첫째 줄부터 아홉째 줄까지 한 줄에 하나의 자연수가 주어진다. 주어지는 자연수는 100보다 작다.

출력

첫째 줄에 최댓값을 출력하고, 둘째 줄에 최댓값이 몇 번째 수인지를 출력한다.

입력 예	출력 예
3	85
29	8
38	
12	
57	
74	
40	
85	
61	

순차탐색



<https://onlinegdb.com/r1w5CCun4>

이진 탐색

2. 이진 탐색

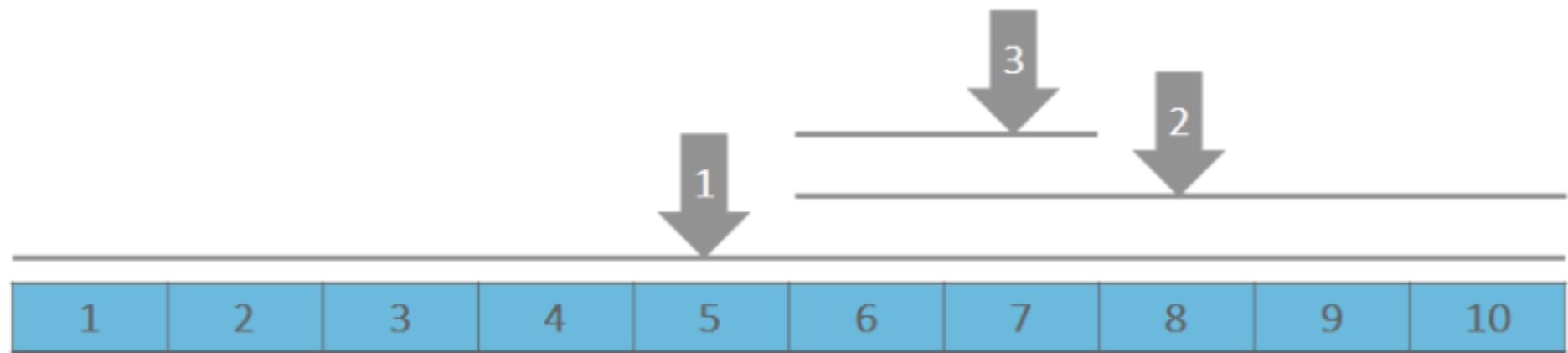
업앤다운 게임처럼

중간에 위치한 값으로 비교해나가는 방식

이진 탐색

이미 정렬된 데이터를 탐색할 때만 사용 가능하다.

e.g. 7을 찾을 때



이진탐색 - 구현방법

직접 구현 - 반복문을 이용한 이진 탐색

```
1 //반복문을 이용한 이진탐색을 이용하여 탐색
2 bool BinarySearch(int *arr, int len, int key){
3     int start = 0;
4     int end = len-1;
5     int mid;
6
7     while(end - start >= 0) {
8         mid = (start + end) / 2;    //중앙 값
9
10        if (arr[mid] == key) {    //key값을 찾았을때
11            return true;
12
13        } else if (arr[mid] > key) {    //key값이 mid의 값보다 작을때 (왼쪽으로)
14            end = mid - 1;
15
16        } else {    //key값이 mid의 값보다 클때 (오른쪽으로)
17            start = mid + 1;
18
19        }
20    }
21    return false;
22 }
```

이진탐색 - 구현방법

직접 구현 - 재귀를 이용한 이진 탐색

```
1 //재귀를 이용한 이진탐색을 이용하여 탐색
2 bool BinarySearch(int *arr, int start, int end, int key) {
3
4     if (start > end) return false; //key 값이 없을 때
5
6     int mid = (start + end) / 2;
7
8     if (arr[mid] == key) { //key 값을 찾았을 때
9         return true;
10
11     } else if (arr[mid] > key) { //key 값이 mid 의 값보다 작을때(왼쪽으로)
12         return BinarySearch(arr, start, mid - 1, key);
13
14     } else { //key 값이 mid 의 값보다 작을때(왼쪽으로)
15         return BinarySearch(arr, mid + 1, end, key);
16
17     }
18 }
```

이진탐색 - 구현방법

STL이용 - `binary_search` 함수를 이용한 이진 탐색

`<algorithm>` 헤더 파일안에 있는 `binary_search` 함수를 이용하면 됩니다.

`binary_search` 의 기본형은 이렇습니다.

```
1 template <class ForwardIterator, class T>
2     bool binary_search (ForwardIterator first, ForwardIterator last, const T& val)
```

```
1  #include<iostream>
2  #include<algorithm>
3  using namespace std;
4
5  //STL를 이용한 이진탐색을 이용하여 탐색
6  int main(void){
7      int n= 100;
8      int arr[n];
9
10     for(int i=0; i<n; i++){
11         arr[i] = i;
12     }
13
14     cout << "exist : " << binary_search(arr, arr+n, 70) << endl;
15
16     return 0;
17 }
```

이진탐색

문제

lower bound

n 개로 이루어진 정수 집합에서 원하는 수 k 이상인 수가 처음으로 등장하는 위치를 찾으시오.

단, 입력되는 집합은 오름차순으로 정렬되어 있으며, 같은 수가 여러 개 존재할 수 있다.

입력

첫째 줄에 한 정수 n 이 입력된다.

둘째 줄에 n 개의 정수가 공백으로 구분되어 입력된다.

셋째 줄에는 찾고자 하는 값 k 가 입력된다.

(단, $2 \leq n \leq 1,000,000$, 각 원소의 크기는 $100,000,000$ 을 넘지 않는다.)

출력

찾고자 하는 원소의 위치를 출력한다. 만약 모든 원소가 k 보다 작으면 $n+1$ 을 출력한다.

입력 예	출력 예
5 1 3 5 7 7 7	4
8 1 2 3 5 7 9 11 15 6	5
5 1 2 3 4 5 7	6
5 2 2 2 2 2 1	1

이진 탐색

답안

<https://onlinegdb.com/rJu-byK3N>

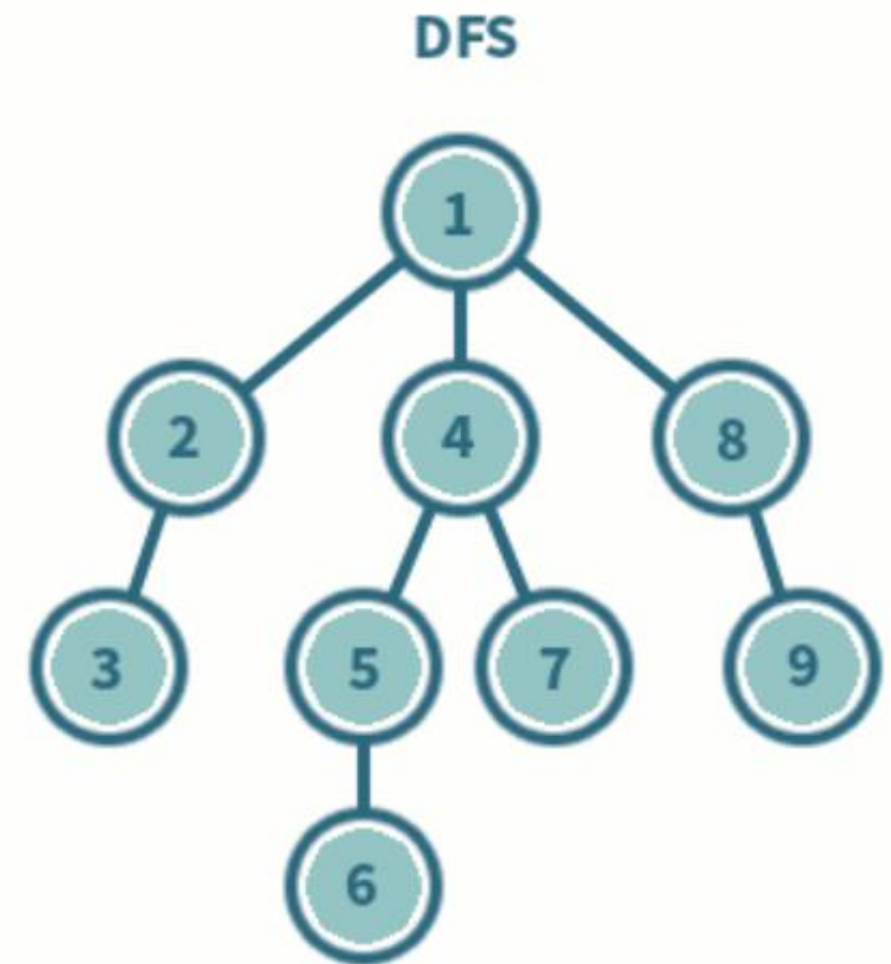
순차탐색 vs 이진탐색 성능비교

$O(N)$ vs $O(\log N)$

<https://onlinegdb.com/HkGB-yKnV>

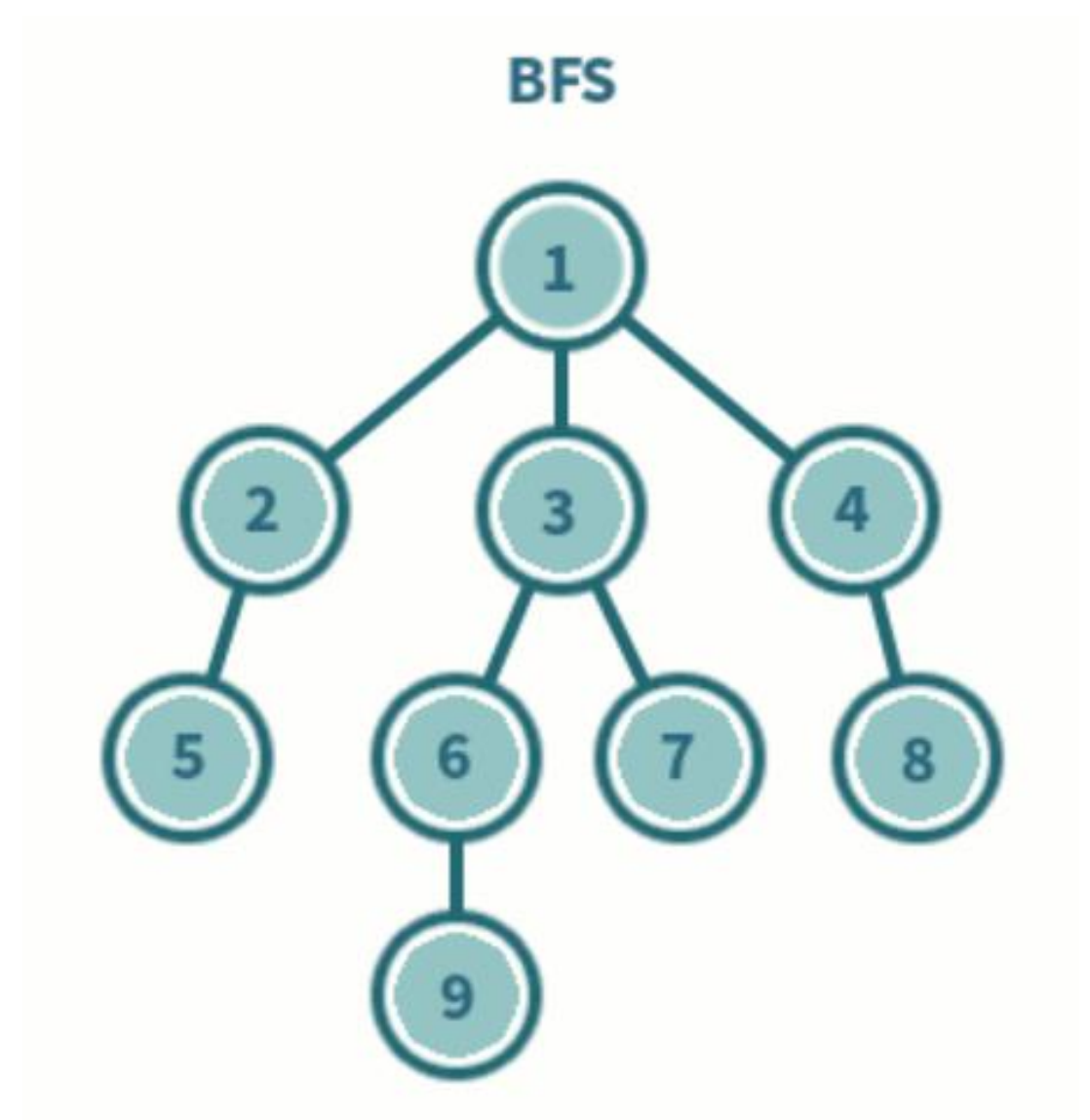
DFS (Depth First Search)

- 현재 정점에서 갈 수 있는 점들까지 들어가면서 탐색
- 구현 : 스택 또는 재귀함수로 구현



BFS(Breadth First Search)

- 현재 정점에 연결된 가까운 점들부터 탐색
- 구현 : 큐를 이용해서 구현



DFS/BFS 기본 구현 문제 풀이

<https://www.acmicpc.net/problem/1260>

문제

그래프를 DFS로 탐색한 결과와 BFS로 탐색한 결과를 출력하는 프로그램을 작성하시오. 단, 방문할 수 있는 정점이 여러 개인 경우에는 정점 번호가 작은 것을 먼저 방문하고, 더 이상 방문할 수 있는 점이 없는 경우 종료한다. 정점 번호는 1번부터 N번까지이다.

입력

첫째 줄에 정점의 개수 N ($1 \leq N \leq 1,000$), 간선의 개수 M ($1 \leq M \leq 10,000$), 탐색을 시작할 정점의 번호 V 가 주어진다. 다음 M 개의 줄에는 간선이 연결하는 두 정점의 번호가 주어진다. 어떤 두 정점 사이에 여러 개의 간선이 있을 수 있다. 입력으로 주어지는 간선은 양방향이다.

출력

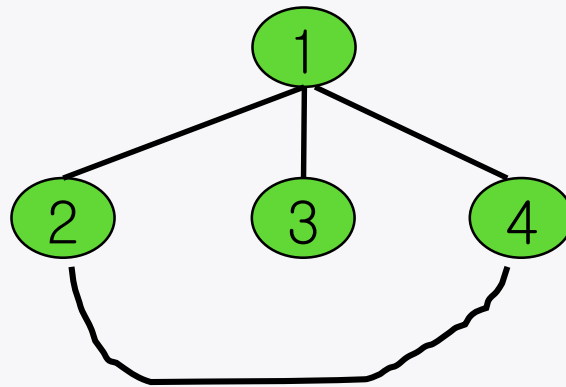
첫째 줄에 DFS를 수행한 결과를, 그 다음 줄에는 BFS를 수행한 결과를 출력한다. V 부터 방문된 점을 순서대로 출력하면 된다.

DFS/BFS 기본 구현 문제 풀이

첫째 줄에 DFS를 수행한 결과를, 그 다음 줄에는 BFS를 수행한 결과를 출력한다. V부터 방문된 점을 순서대로 출력하면 된다.

예제 입력 1 복사

```
4 5 1
1 2
1 3
1 4
2 4
3 4
```

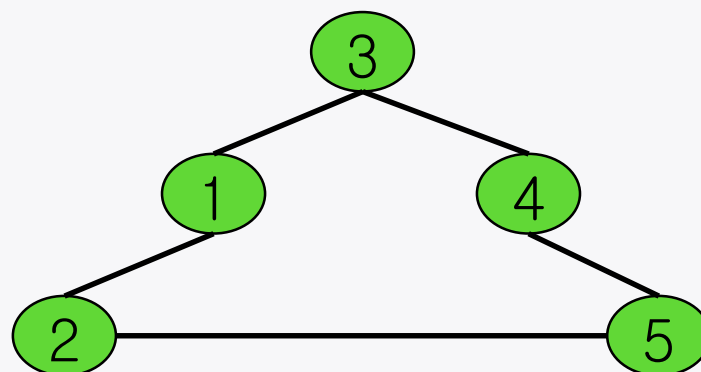


예제 출력 1 복사

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

예제 입력 2 복사

```
5 5 3
5 4
5 2
1 2
3 4
3 1
```



예제 출력 2 복사

```
3 1 2 5 4
3 1 4 2 5
```

DFS/BFS 기본 구현 문제 풀이

답안



DFS-recursion 활용 : <https://onlinegdb.com/rkUbSQFhV>

DFS-stack 활용 : <https://onlinegdb.com/BJqykVF2E>

BFS-queue 활용 : <https://onlinegdb.com/HJBvxEK2N>

미로찾기

- 시작점(0,0) 에서 4개 방향(왼쪽, 오른쪽, 위, 아래)으로 이동하여 출구 (4,4) 까지 가기
- 출구로 갈 수 있는 경로는 몇 개인가?

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

미로찾기

풀이

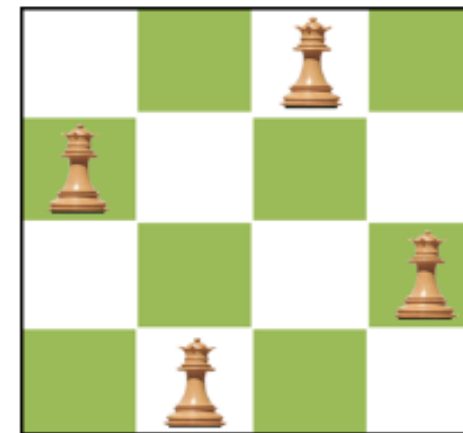
<https://onlinegdb.com/r1YRWVthV>

Backtracking

n-queen

전산학에서 백트래킹 문제로 n-queen problem이 유명하다.
이 문제는 $n \times n$ 체스 보드판에 n 개의 queen을 서로 공격하지 못하도록 배치하는 방법을 찾아내는 문제이다.

아래 그림은 n 이 4일 경우 queen을 서로 공격하지 못하게 배치한 한 예를 나타낸다.



체스판 크기 및 queen의 수를 나타내는 n 을 입력받아서 서로 공격하지 못하도록 배치하는 총 방법의 수를 구하는 프로그램을 작성하시오.

입력

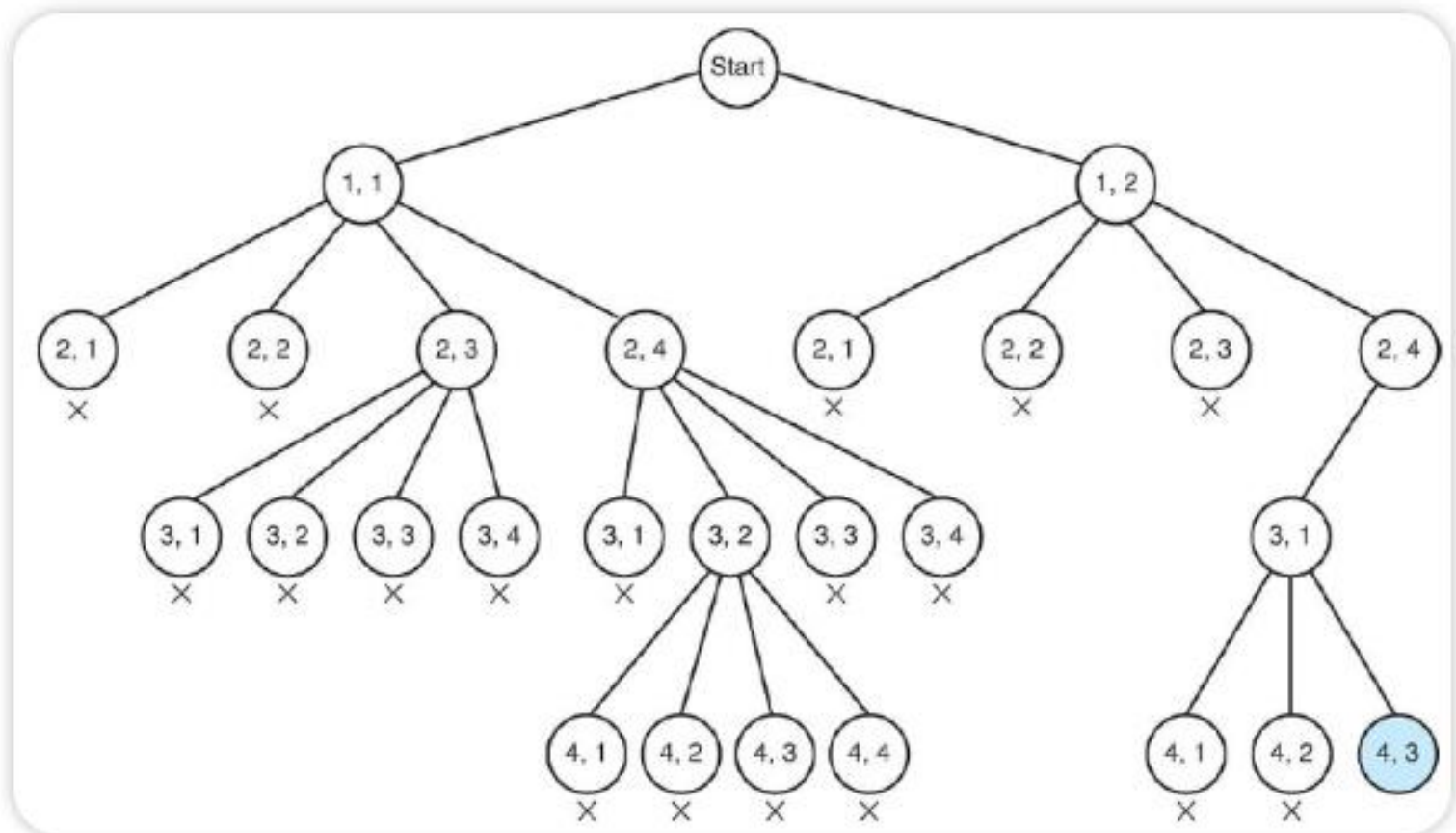
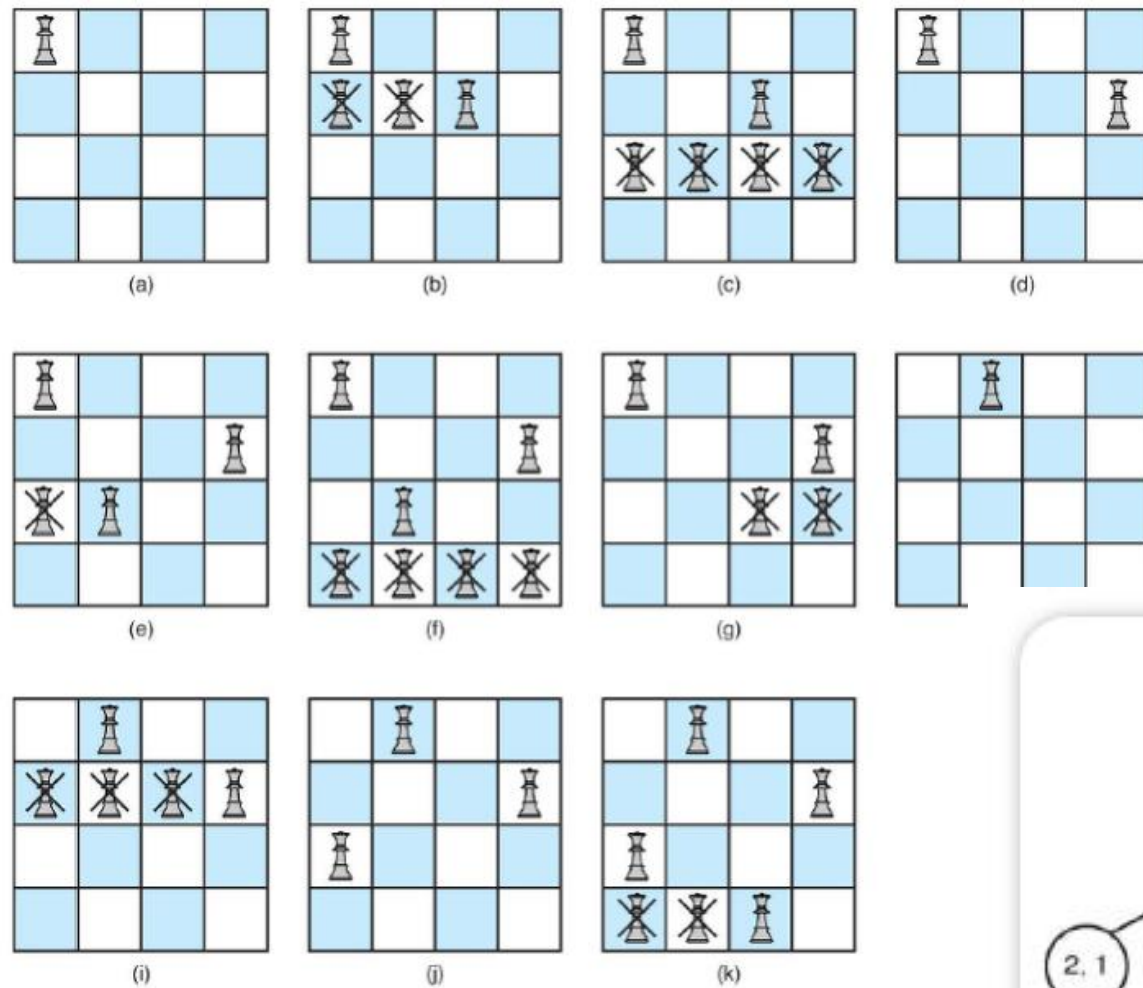
정수 n 이 입력으로 들어온다. ($3 \leq n \leq 9$)

출력

서로 다른 총 경우의 수를 출력한다.

입력 예	출력 예
4	2

Backtracking



Backtracking



<https://onlinegdb.com/HyuKgBK34>

BFS 문제 풀이

<https://www.acmicpc.net/problem/1963>

소수를 유난히도 좋아하는 창영이는 게임 아이디 비밀번호를 4자리 '소수'로 정해놓았다. 어느 날 창영이는 친한 친구와 대화를 나누었는데:

- “이제 슬슬 비번 바꿀 때도 됐잖아”
- “응 지금은 1033으로 해놨는데... 다음 소수를 무엇으로 할지 고민중이야”
- “그럼 8179로 해”
- “흠... 생각 좀 해볼게. 이 게임은 좀 이상해서 비밀번호를 한 번에 한 자리 밖에 못 바꾼단 말이야. 예를 들어 내가 첫 자리만 바꾸면 8033이 되니까 소수가 아니잖아. 여러 단계를 거쳐야 만들 수 있을 것 같은데... 예를 들면... 1033 1733 3733 3739 3779 8779 8179처럼 말이야.”
- “흠... 역시 소수에 미쳤군. 그럼 아예 프로그램을 짜지 그래. 네 자리 소수 두 개를 입력받아서 바꾸는데 몇 단계나 필요한지 계산하게 말야.”
- “귀찮아”

그렇다. 그래서 여러분이 이 문제를 풀게 되었다. 입력은 항상 네 자리 소수만(1000 이상) 주어진다고 가정하자. 주어진 두 소수 A에서 B로 바꾸는 과정에서도 항상 네 자리 소수임을 유지해야 하고, '네 자리 수'라 하였기 때문에 0039 와 같은 1000 미만의 비밀번호는 허용되지 않는다.

입력

첫 줄에 test case의 수 T가 주어진다. 다음 T줄에 걸쳐 각 줄에 1쌍씩 네 자리 소수가 주어진다.

출력

각 test case에 대해 두 소수 사이의 변환에 필요한 최소 회수를 출력한다. 불가능한 경우 Impossible을 출력한다.

BFS 문제 풀이

<https://www.acmicpc.net/problem/1963>

예제 입력 1 [복사](#)

```
3
1033 8179
1373 8017
1033 1033
```

예제 출력 1 [복사](#)

```
6
7
0
```

<참고> 에라토스테네스의 체

https://ko.wikipedia.org/wiki/%EC%97%90%EB%9D%BC%ED%86%A0%EC%8A%A4%ED%85%8C%EB%84%A4%EC%8A%A4%EC%9D%98_%EC%B2%B4

BFS 문제 풀이

답안

<https://onlinegdb.com/r1A1GDK3N>