SW 역량 테스트 대비반 – 3회차

2020년 07월 22일

지난 수업 정리

- **algorithm 헤더**
  - sort
  - count
  - find
  - copy
  - swap
  - transform
  - min/max
- **cmath**
  - abs
- **cctype**
  - toupper, tolower

문제 풀이 6

http://tech.kakao.com/2017/11/14/kakao-blind-recruitment-round-3/

문제2. 압축

신입사원 어피치는 카카오톡으로 전송되는 메시지를 압축하여 전송 효율을 높이는 업무를 맡게 되었다. 메시지를 압축하더라도 전달되는 정보가 바뀌어서는 안 되므로, 압축 전의 정보를 완벽하게 복원 가능한 무손실 압축 알고리즘을 구현하기로 했다.

어피치는 여러 압축 알고리즘 중에서 성능이 좋고 구현이 간단한 **LZW**(Lempel–Ziv–Welch) 압축을 구현하기로 했다. LZW 압축은 1983년 발표된 알고리즘으로, 이미지 파일 포맷인 GIF 등 다양한 응용에서 사용되었다.

LZW 압축은 다음 과정을 거친다.

1. 길이가 1인 모든 단어를 포함하도록 사전을 초기화한다.
2. 사전에서 현재 입력과 일치하는 가장 긴 문자열 `w` 를 찾는다.
3. `w` 에 해당하는 사전의 색인 번호를 출력하고, 입력에서 `w` 를 제거한다.
4. 입력에서 처리되지 않은 다음 글자가 남아있다면(`c`), `w+c` 에 해당하는 단어를 사전에 등록한다.
5. 단계 2로 돌아간다.

압축 알고리즘이 영문 대문자만 처리한다고 할 때, 사전은 다음과 같이 초기화된다. 사전의 색인 번호는 정수값으로 주어지며, 1부터 시작한다고 하자.

문제 풀이 6

아이디어

1. 사전을 이용하라는 말에 바로 map이라는 자료구조를 이용 (dict)
2. 영문 대문자 색인 초기화 (1 ~ 26)
3. 가장 긴 문자열 w를 찾기 위해서는 글자를 붙여가며 사전에 있는지 찾아보다가 사전에 존재하지 않는다면 그 전의 문자열이 가장 긴 w이기 때문에 해당 문자열의 색인번호를 answer에 추가해주고 해당하는 새로운 문자열은 색인에 추가
4. 가장 긴 문자열을 찾으려할 때 만약 msg의 가장 마지막 문자열을 붙였다면 그 문자열까지가 가장 긴 문자열 w에 해당하므로 색인 번호를 answer에 추가

문제풀이 6

답안

https://onlinegdb.com/H1i-74Z2E

문제풀이 6

```
int main() {
    string msg, str_in;
    vector<int> answer;
    map<string, int> dictionary;
    int dic_index = 27;
    cin >> msg;
    int len = msg.length();

    for (int i = 0; i < len; i++) {
        str_in += msg[i];
        if (i == len - 1) {
            // 마지막 글자의 경우
            if (str_in.length() == 1) answer.push_back(str_in[0] - 'A' + 1);
            else answer.push_back(dictionary[str_in]);
        }
        else if (dictionary[str_in + msg[i + 1]] == 0) {
            // 현재 입력까지의 단어 저장
            if (str_in.length() == 1) answer.push_back(str_in[0] - 'A' + 1);
            else answer.push_back(dictionary[str_in]);
            // 다음 글자 포함한 단어 사전 추가
            dictionary[str_in + msg[i + 1]] = dic_index++;
            str_in.clear();
```

오늘 할 내용

구분	내용
1주차 (07/08)	수업 방향 설명 및 C++ 기초 문법 강의 및 실습
2주차 (07/15)	기초 자료구조 강의 및 실습
3주차 (07/22)	재귀함수/탐색 알고리즘 강의 및 실습 1
4주차 (07/29)	
5주차 (08/05)	
6주차 (08/12)	
7주차 (08/19)	
8주차 (08/26)	
(옵션) 9주차 ~	

재귀함수 기본

탐색알고리즘

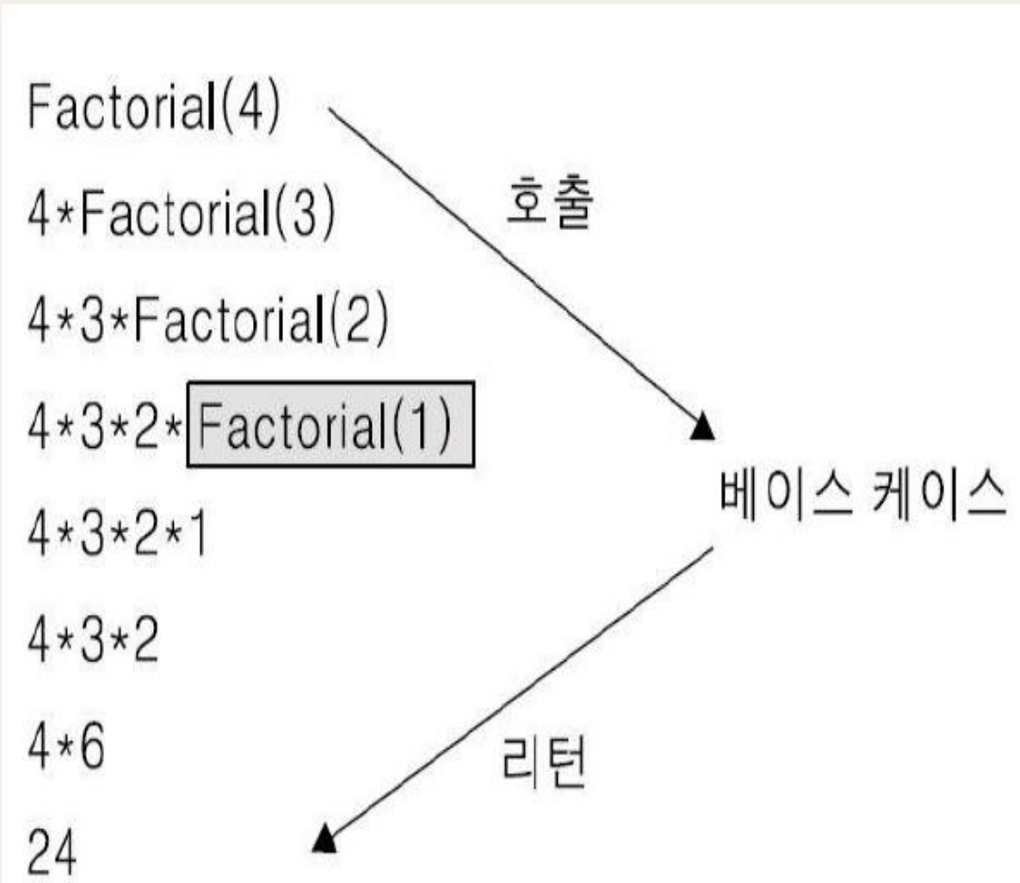
(선형탐색(순차/이진탐색),

비선형탐색(DFS, BFS))

관련 문제 풀이

재귀호출

```
int Factorial(int n)
{ if (n = = 1)
      return 1;
   else
     return(n * Factorial(n-1));
  }
```



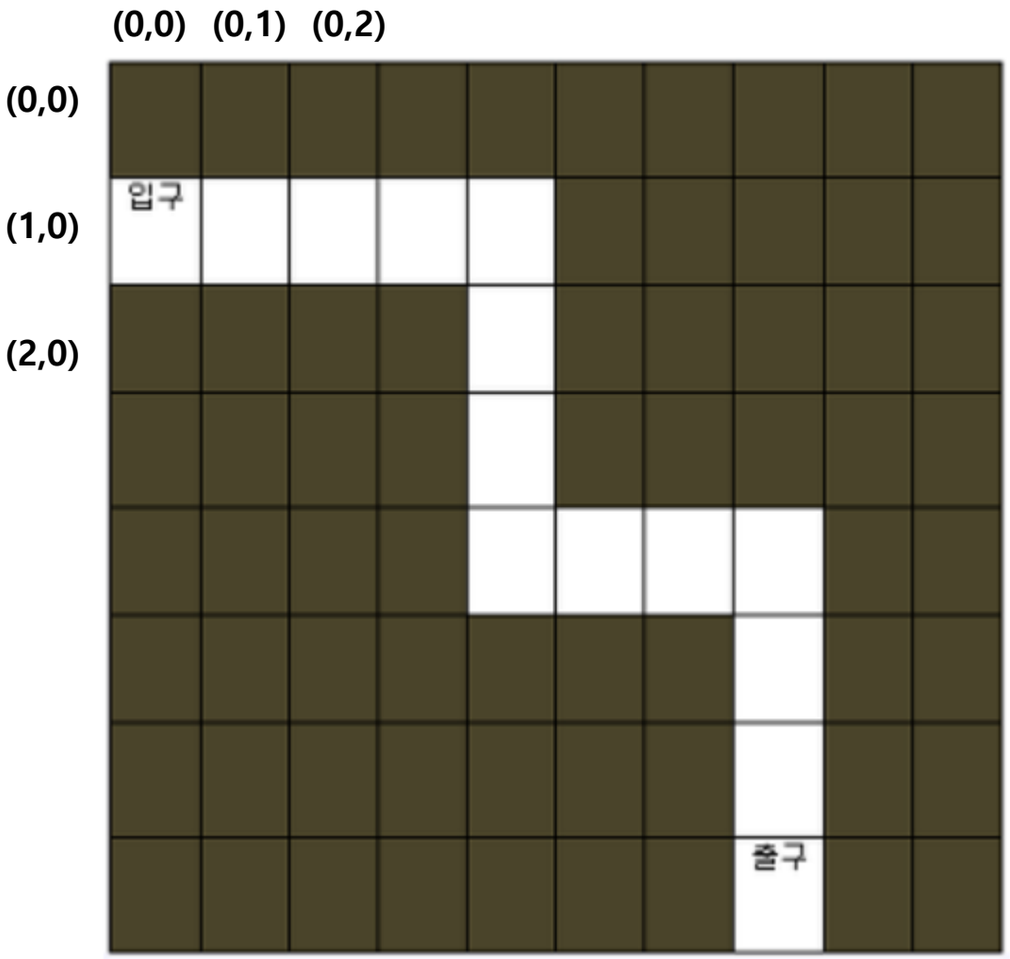
[그림 4-3] 재귀호출의 들어가기와 나오기

```
int Factorial(int n)
{ int product = 1;
  for (int i = 1; i <= n; i++)
      product *= i;
  return product;
}
```

팩토리얼
곱셈의 결과 값을 초기화
1부터 n까지
계속 곱해서 저장
결과를 리턴

<반복문으로의 변환>

미로찾기 (8*10) – 재귀함수



<미로의 배열 표현>

```
1  int ar[8][10] =
2  {
3  1,1,1,1,1,1,1,1,1,1,
4  0,0,0,0,0,1,1,1,1,1,
5  1,1,1,1,0,1,1,1,1,1,
6  1,1,1,1,0,1,1,1,1,1,
7  1,1,1,1,0,0,0,0,1,1,
8  1,1,1,1,1,1,1,0,1,1,
9  1,1,1,1,1,1,1,0,1,1,
1  1,1,1,1,1,1,1,0,1,1
0  };
```

<결과>

```
1, 0
1, 1
1, 2
1, 3
1, 4
2, 4
3, 4
4, 4
4, 5
4, 6
4, 7
5, 7
6, 7
7, 7
Hit any key to close this window...
```

미로찾기 (8*10) – 재귀함수

답안

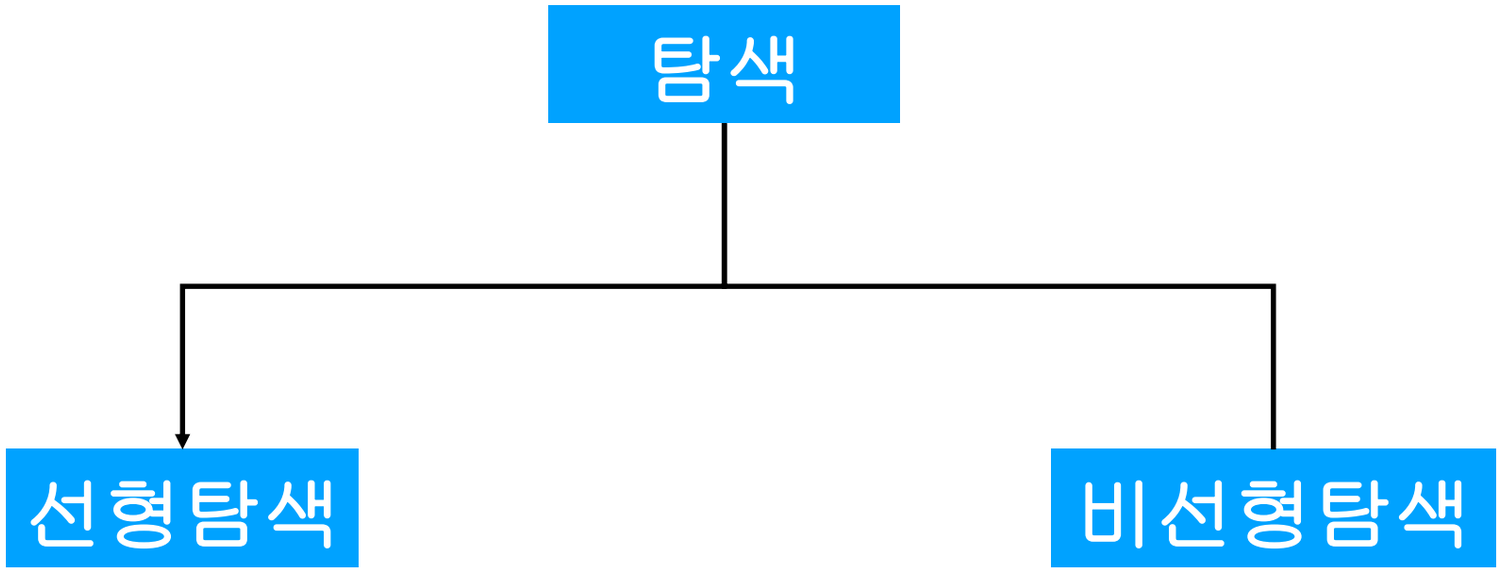
```
#include <stdio.h>

int ar[8][10] =
{
1,1,1,1,1,1,1,1,1,1,
0,0,0,0,0,1,1,1,1,1,
1,1,1,1,0,1,1,1,1,1,
1,1,1,1,0,1,1,1,1,1,
1,1,1,1,0,0,0,0,1,1,
1,1,1,1,1,1,1,0,1,1,
1,1,1,1,1,1,1,0,1,1,
1,1,1,1,1,1,1,0,1,1
};
int f[8][10];

void find(int i, int j) {
        f[i][j]=1;
        printf("%d, %d₩n",i,j);
if(j+1<10&&ar[i][j+1]!=1&&f[i][j+1]==0) find(i ,j+1);
if(j-1>=0&&ar[i][j-1]!=1&&f[i][j-1]==0) find(i, j-1);
if(i+1<8&&ar[i+1][j]!=1&&f[i+1][j]==0) find(i+1, j);
if(i-1>=0&&ar[i-1][j]!=1&&f[i-1][j]==0) find(i-1, j);
}

int main() {
      find(1,0);
}
```

탐색 알고리즘



순차탐색
이진탐색

깊이우선탐색(DFS)
너비우선탐색(BFS)

순차탐색

1 순차 탐색

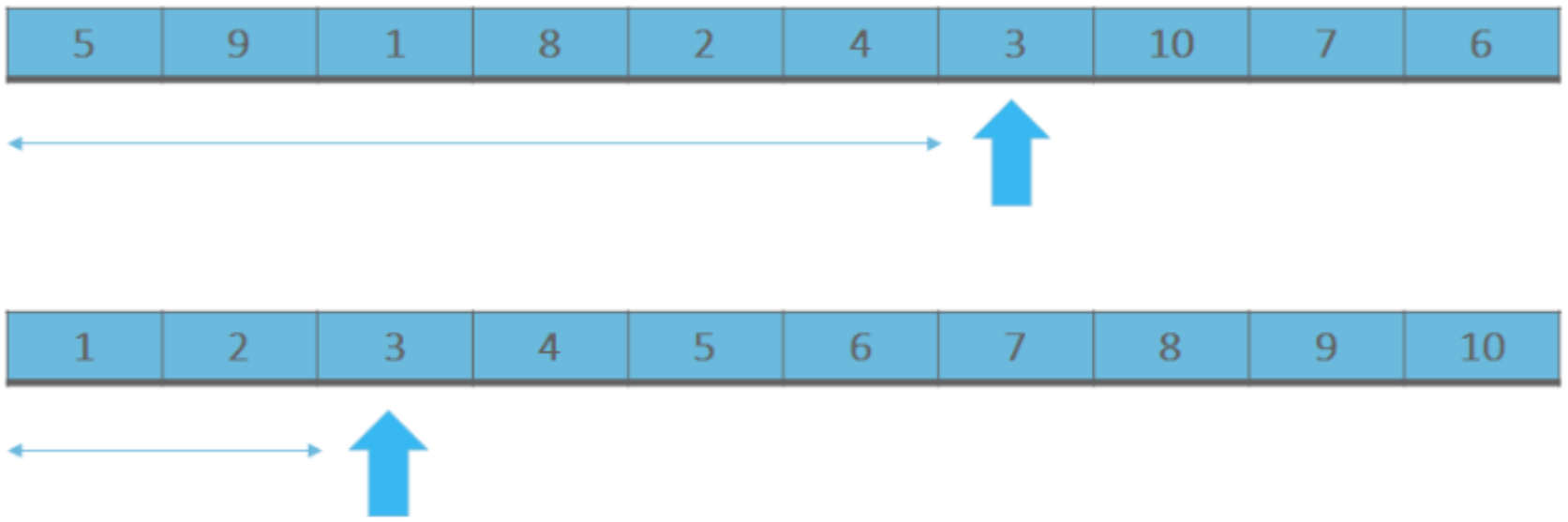
처음부터 원하는 데이터가 나올 때까지

순서대로 찾는 방식

순차 탐색

데이터의 정렬 여부와 상관없이 사용가능

e.g. 3을 찾을 때



시간복잡도 : O(n)

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 29 38 12 57 74 40 85 61	85 8

출처: 한국정보올림피아드(2007 지역본선 초등부)

순차탐색

답안

1. 입력 값이 크지 않으므로 자료구조로 배열을 이용 (index 이용에 용이)
2. 처음부터 비교해가며 최대값을 찾음
3. 끝까지 비교 후, 최대값과 해당 배열 값의 index를 출력

https://onlinegdb.com/r1w5CCun4

이진탐색

2. 이진탐색

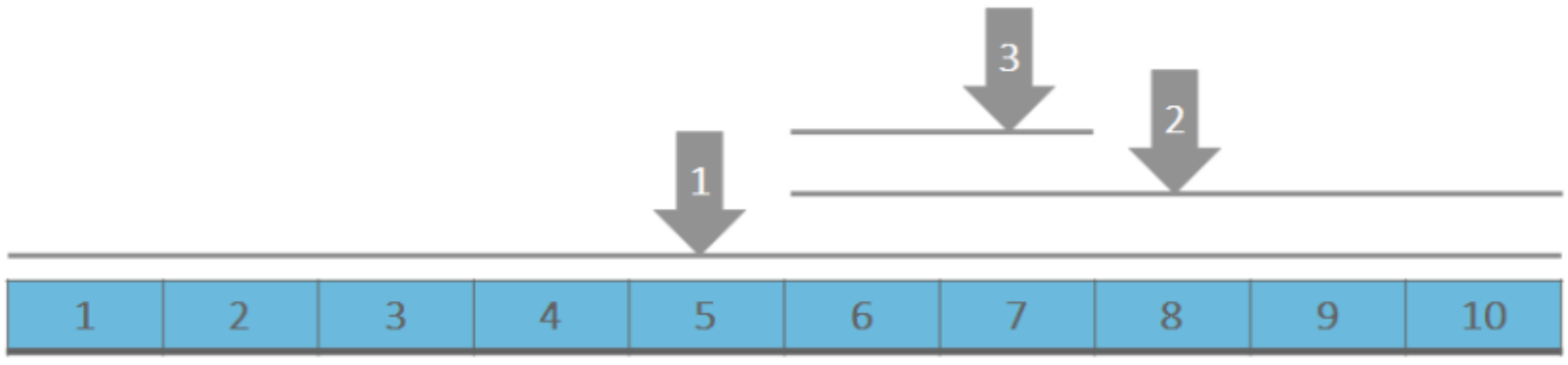
업앤다운 게임처럼

중간에 위치한 값으로 비교해나가는 방식

이진 탐색

이미 정렬된 데이터를 탐색할 때만 사용 가능하다.

e.g. 7을 찾을 때



시간복잡도 : $O(\log_2 n)$

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> 해더 파일안에 있는 **binary_search** 함수를 이용하면 됩니다.

binary_search 의 기본형은 이러합니다.

```
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

n개로 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 n이 입력된다.
둘째 줄에 n개의 정수가 공백으로 구분되어 입력된다.
셋째 줄에는 찾고자 하는 값 k가 입력된다.

(단, 2 <= n <= 1,000,000 , 각 원소의 크기는 100,000,000을 넘지 않는다.)

출력

찾고자 하는 원소의 위치를 출력한다. 만약 모든 원소가 k보다 작으면 n+1을 출력한다. **(위치는 1부터 시작)**

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

이진탐색

답안

1. 입력 값이 크지 않으므로 자료구조로 배열을 이용 (index 이용에 용이)
2. 배열의 index를 기준으로 이진탐색을 수행 (start, end)
3. 일치하는 값의 index의 +1을 반환하거나 일치하는 값이 없다면 찾는 값(k)보다 작은 값 +1을 출력

https://onlinegdb.com/rJu-byK3N

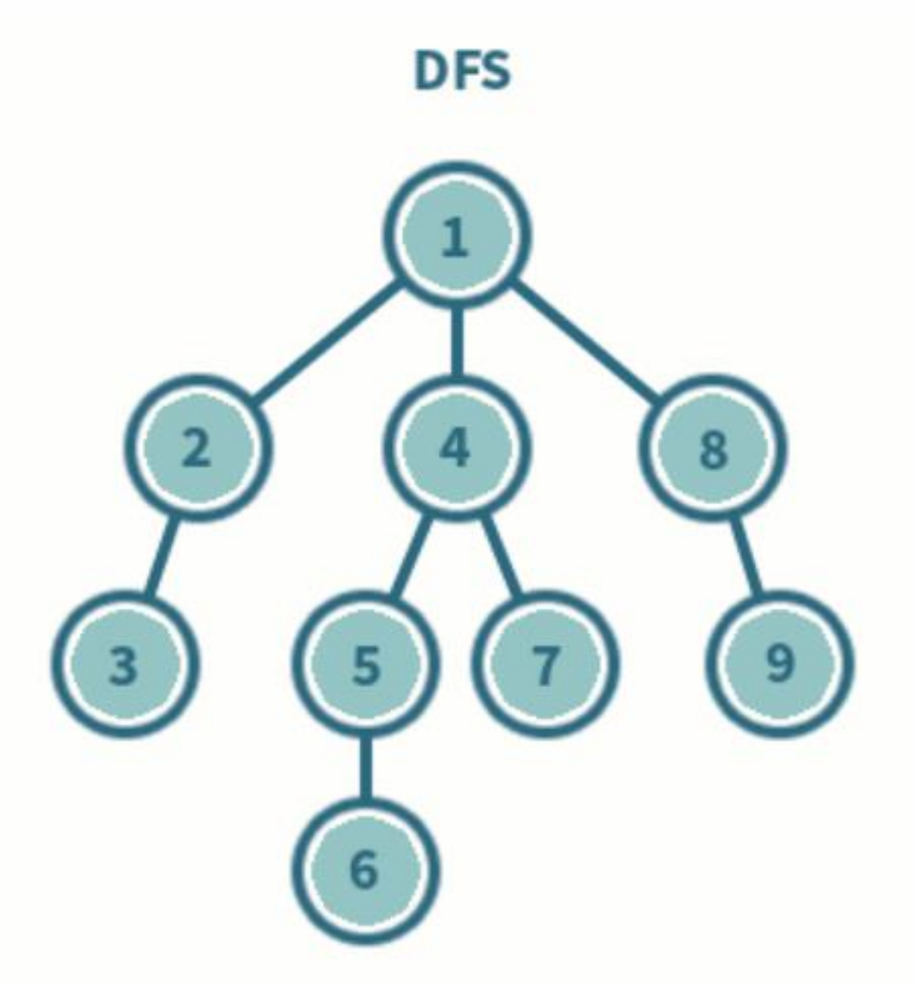
순차탐색 vs 이진탐색 성능비교

O(N) vs O(log_2N)

https://onlinegdb.com/HkGB-yKnV

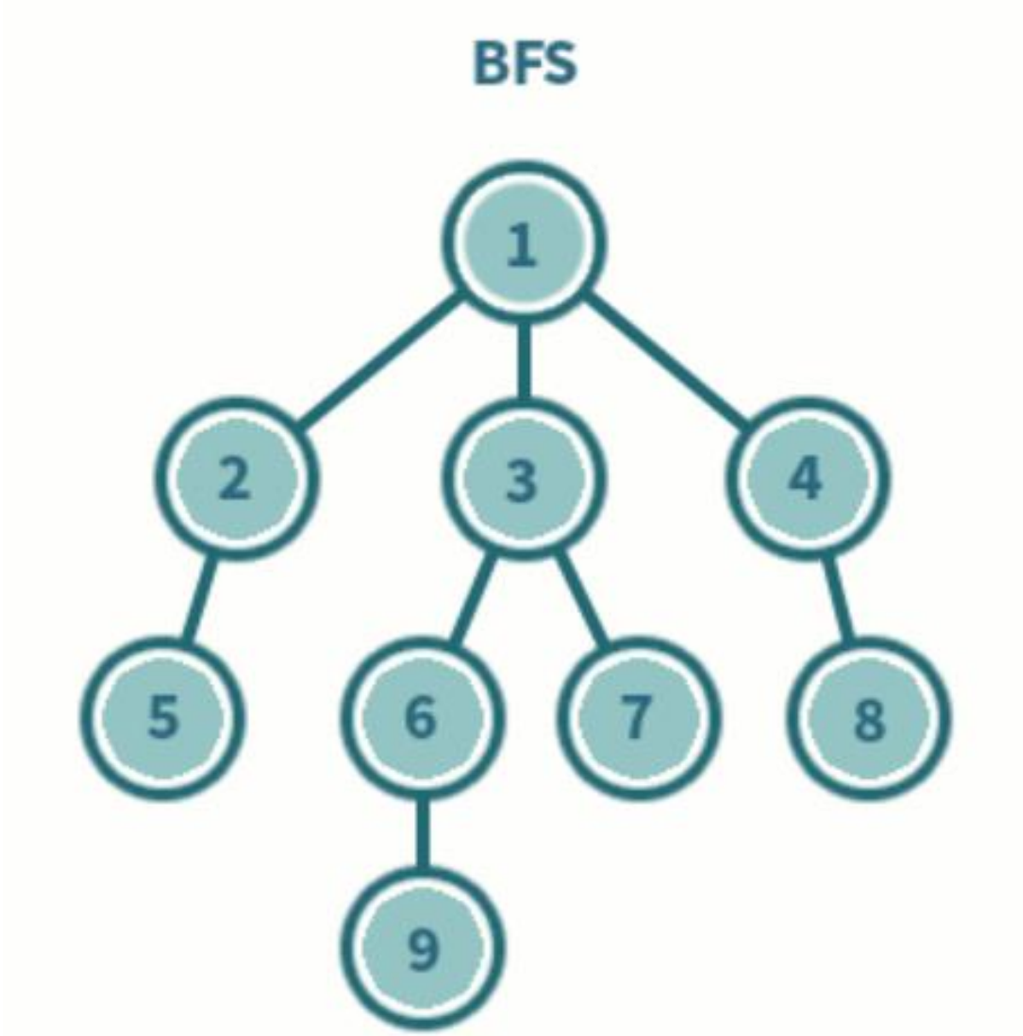
DFS (Depth First Search)

- 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 ≤ N ≤ 1,000), 간선의 개수 M(1 ≤ M ≤ 10,000), 탐색을 시작할 정점의 번호 V가 주어진다. 다음 M개의 줄에는 간선이 연결하는 두 정점의 번호가 주어진다. 어떤 두 정점 사이에 여러 개의 간선이 있을 수 있다. 입력으로 주어지는 간선은 양방향이다.

출력

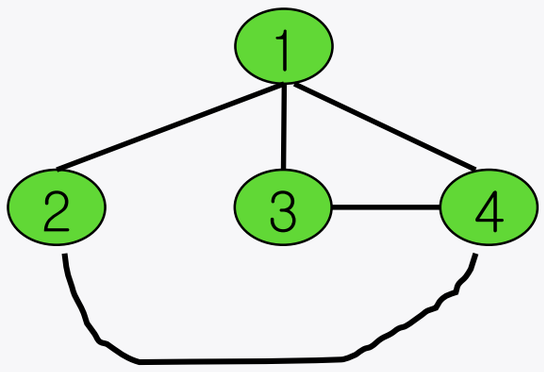
첫째 줄에 DFS를 수행한 결과를, 그 다음 줄에는 BFS를 수행한 결과를 출력한다. V부터 방문된 점을 순서대로 출력하면 된다.

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
```

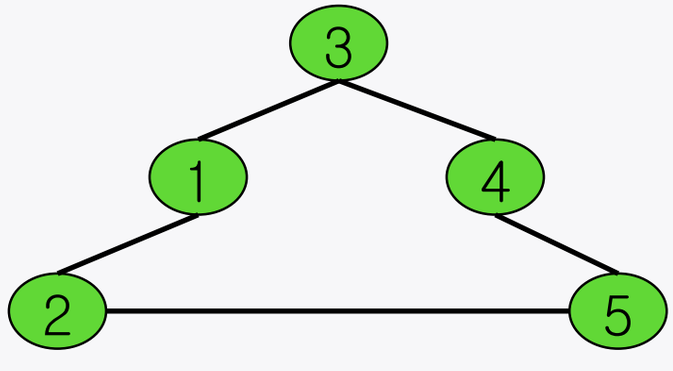


예제 출력 1 복사

```
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

1. **그래프 저장을 위해 자료구조로 2차원 벡터를 사용**
2. **두번째 정점 번호가 작은 것부터 방문하기 위해 sorting을 수행**
3. **시작 정점부터 재귀/스택을 이용한 DFS 구현 (노드의 방문 여부를 체크)**
4. **시작 정점부터 큐를 활용한 BFS 구현 (노드의 방문 여부를 체크)**

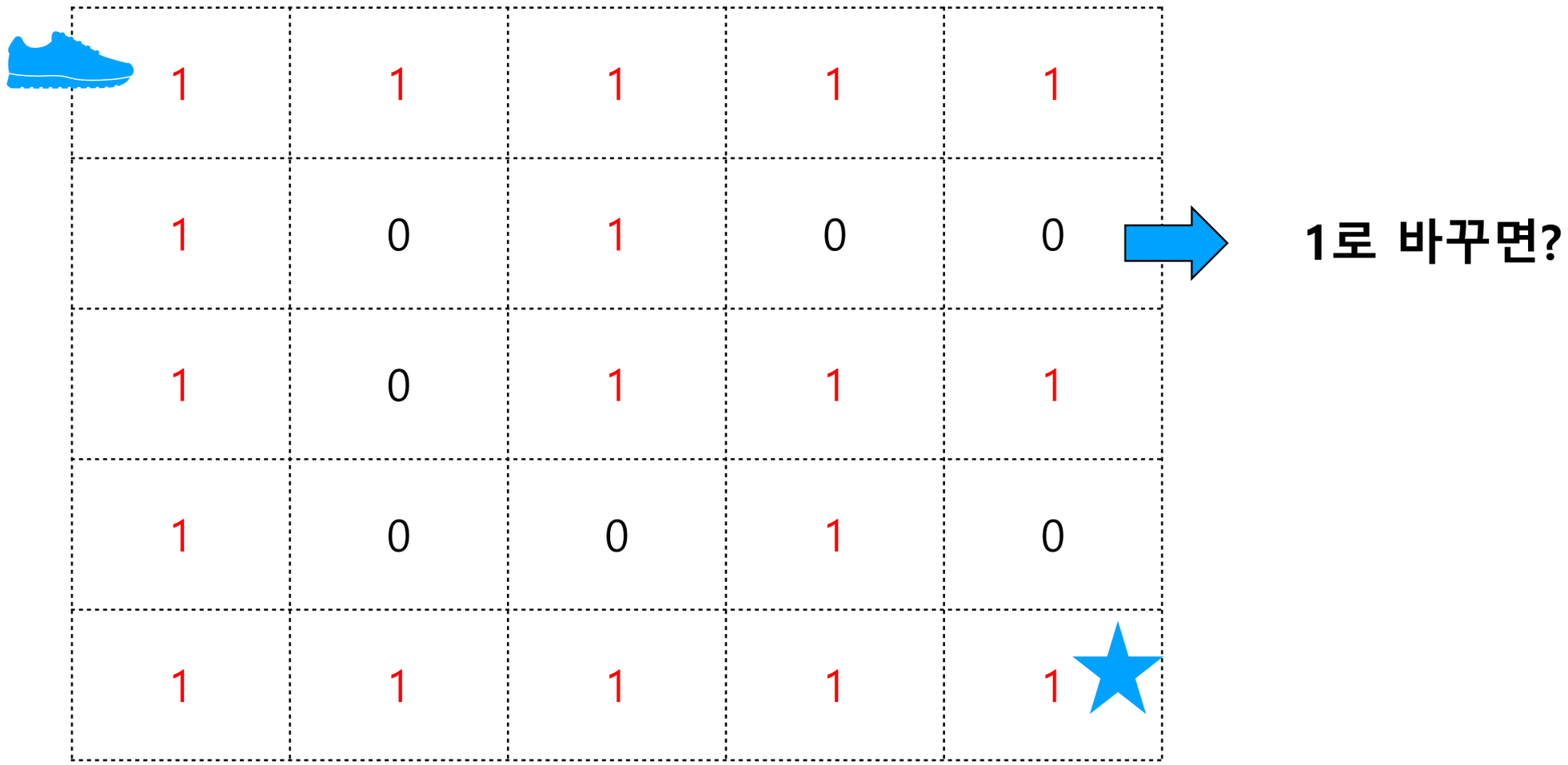
DFS-recursion 활용 : https://onlinegdb.com/Sy2f7g_5r

DFS-stack 활용 : https://onlinegdb.com/HJxmDguqB

BFS-queue활용 : https://onlinegdb.com/rJbHwxd9S

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?



미로찾기

아이디어

1. 2차원 배열을 선언 및 초기화
2. DFS 구현을 위해 재귀함수로 경로 탐색
3. 방문한 경로는 0으로 설정해주고 백트래킹을 하는 경우에는 다시 1로 원복 시켜줌
4. 최종 정정에 왔을 경우 cnt를 증가 시키고 최종 결과를 출력

미로찾기

풀이

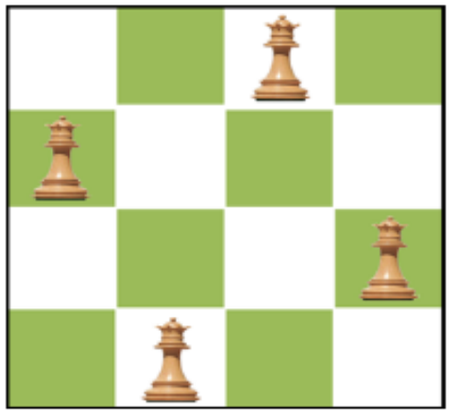
https://onlinegdb.com/B1z924pFI

Backtracking

n-queen

전산학에서 백트래킹 문제로 n-queen problem이 유명하다.
이 문제는 n*n체스 보드판에 n개의 queen을 서로 공격하지 못하도록 배치하는 방법을 찾아내는 문제이다.

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



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

입력

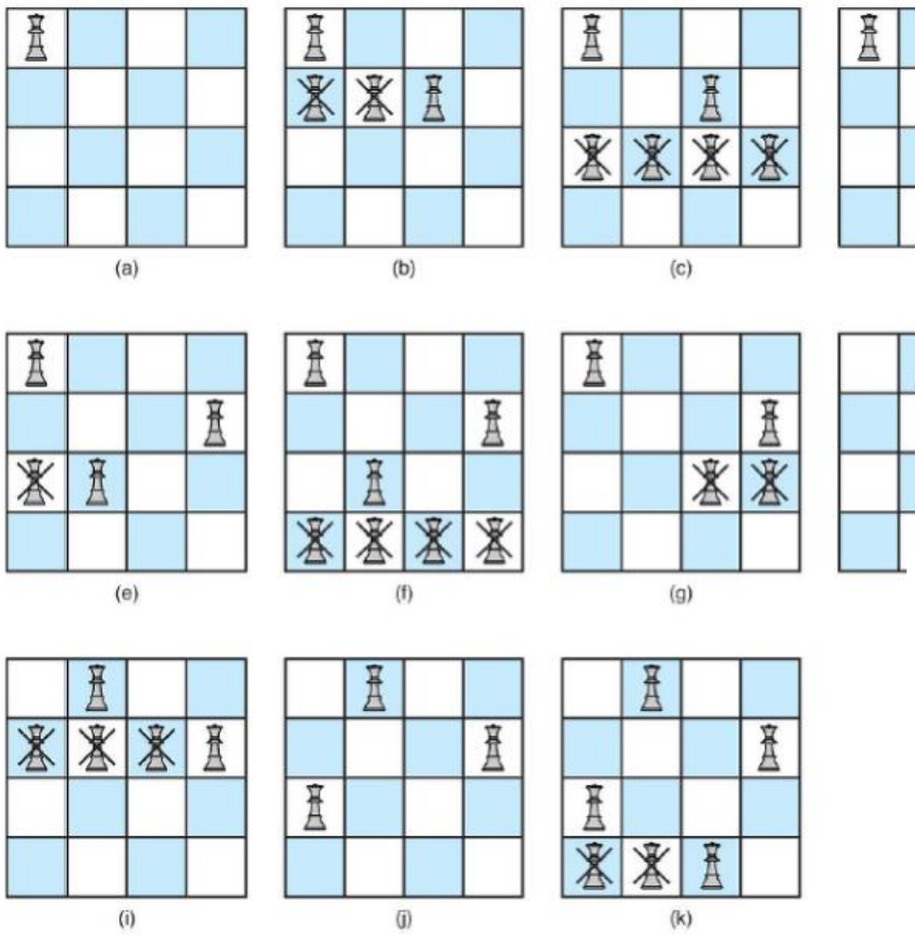
정수 n이 입력으로 들어온다.(3 <= n <= 9)

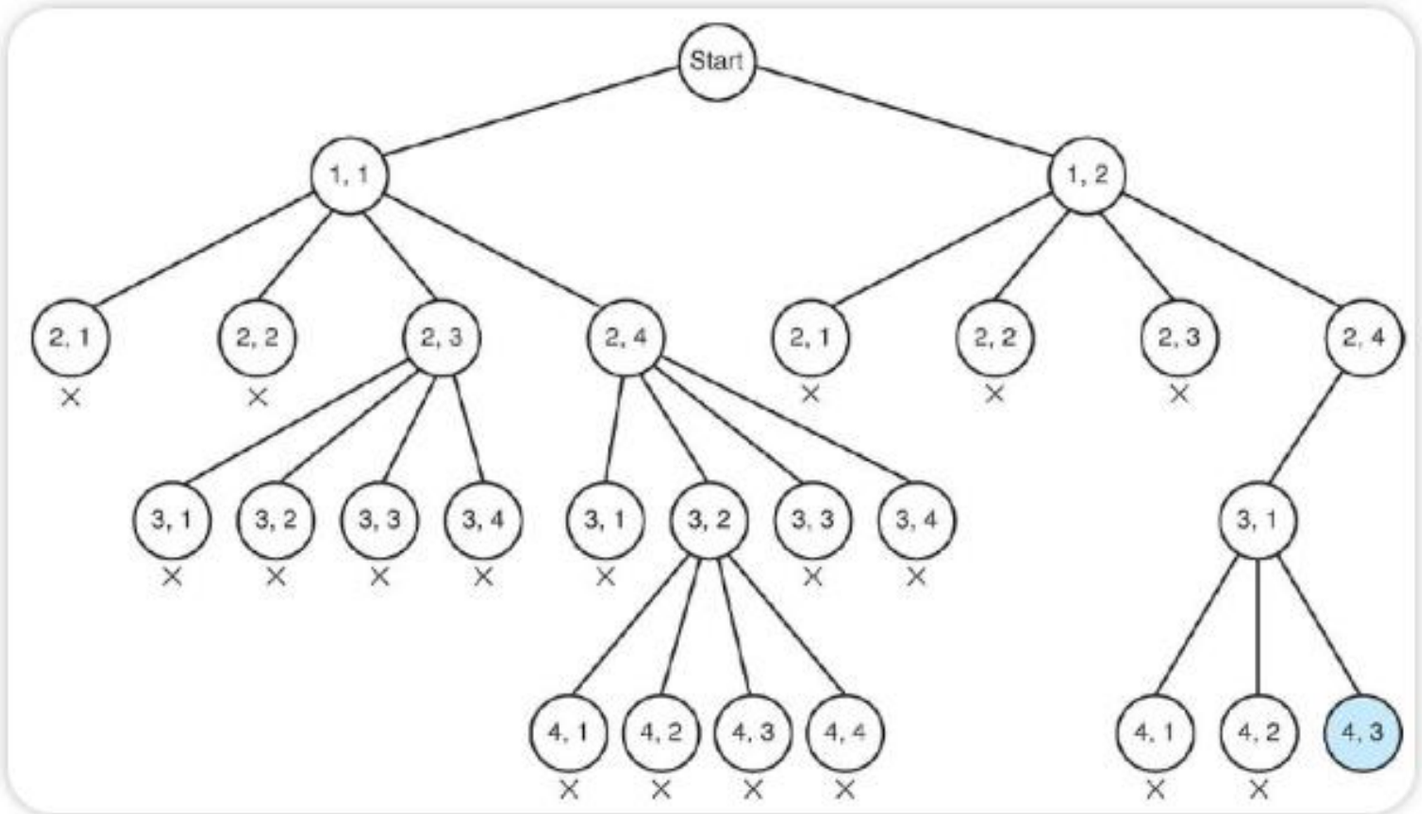
출력

서로 다른 총 경우의 수를 출력한다.

입력 예	출력 예
4	2
8	**92**

Backtracking





Backtracking

아이디어

1. 첫 번째 행, 첫 번째 열에 퀸을 놓는다.
2. 다음 행에서 가능한 가장 왼쪽 열에 퀸을 놓는다.
3. n번째 열에 더 이상 퀸을 놓을 수 없다면 백트래킹 한다.
4. 마지막 행에 퀸을 놓으면 하나의 해를 구한 것이다.
5. 모든 경우를 조사할 때까지 백트래킹 해가며 해들을 구한다.

Backtracking

답안

https://onlinegdb.com/S16FFedqH

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

아이디어

1. 1차원 배열로 소수를 구해주고(에라토스테네스의 체) 다른 1차원 배열로 해당 숫자까지 가는 경로의 수를 넣어준다.
2. 원하는 숫자로 도달하는데 최소 몇 단계가 필요한지 찾을 것이니, 현재 상태에서 숫자 하나만 바꿔서 만들 수 있는
3. 1) 소수이고, 2) 1000 이상의 값이며, 3) 이전에 검사하지 않은 값을 모두 큐에 넣어두며, 여기까지 도달하는데 걸리는 비용은 “이전 비용 + 1”이라고 저장해둔다.
4. ex) 1033(check를 위해 비용을 1부터 시작) -> 숫자 한 자리만 바꿔서 만들 수 있는 1) 소수이고, 2) 1000 이상의 값이며, 3) 이전에 검사하지 않은 숫자가 총 8개며, 모두 큐에 넣었다고 가정. (1433, 1733, 1933, 1013, 1063, 1093, 1031, 1039)
5. 큐의 size는 현재 8이며, 이 숫자에 도달하는데 걸리는 비용은 2.
6. 현재 큐에 들어있는 8개의 숫자를 다시 숫자 한 자리만 바꾸고 1) ~ 3) 까지의 조건을 만족하는 숫자를 큐에 넣으며 여기까지 걸리는 비용을 +1 한다.
7. 8개의 숫자를 모두 검사하고 난 뒤 큐에 새로 저장한 검사 값들은, 1033에서 이 값까지 도달하는데 비용이 3인 상태.
8. 이런식으로 진행하며 원하는 숫자에 도달 시 비용(실제는 -1한 값)을 출력.
9. 위와 같은 시퀀스로 프로그래밍 할 것이기 때문에 Queue와 BFS 개념을 사용할 것이다.

BFS 문제 풀이

답안

https://onlinegdb.com/SyqjKBpK8