8주차 : 탐색

커리큘럼

1. 변수, 입출력
2. 조건문, 반복문
3. 리스트, 튜플, 세트, 딕셔너리
4. 함수
5. class
6. 알고리즘 입문, 그리디 알고리즘
7. 재귀함수
8. **탐색**
9. DP(Dynamic Programming)

탐색(Search)

- 배열(리스트), 그래프 등에서 해당하는 자료를 찾는 것
- 선형 탐색, 이분 탐색
- 브루트 포스, 백트래킹, DFS, BFS

선형 탐색(Linear Search)

- 앞에서부터 순차적으로 해당하는 자료를 찾는 직관적으로 많이 사용하는 방법
- 평균 시간 복잡도 O(n)으로 자료가 아주 크면 적합하지 않음
- 조건의 제약에는 자유로움

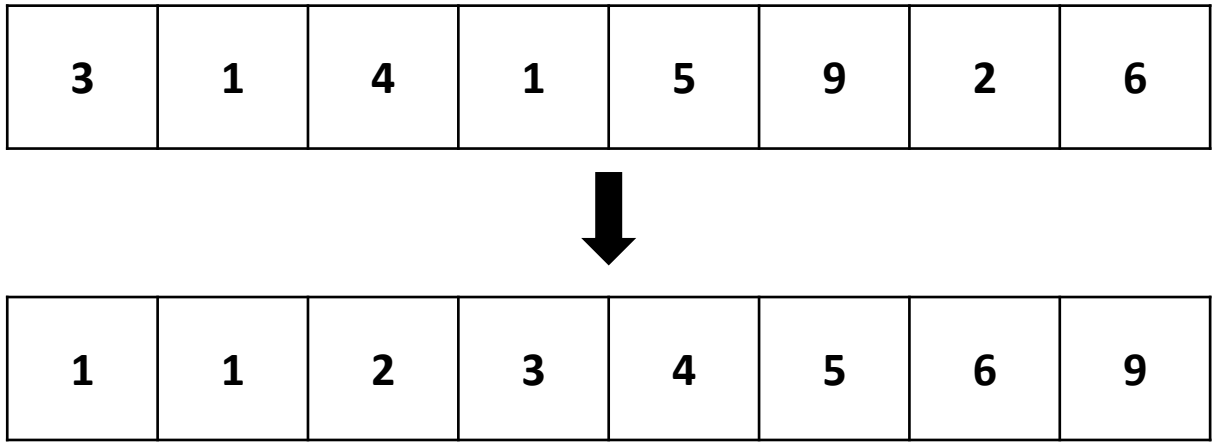
```
def LinearSearch(a_list, key):
    start = 0
    end = len(a_list)-1
    while start <= end:
        if a_list[start] == key:
            return True
        else:
            start += 1
    return False
```

3	1	4	1	5	9	2	6

이분 탐색(Binary Search)

- 정렬된 리스트에 대해서 매 과정마다 절반으로 비교 단위를 줄여가며 자료를 찾는 방법
- 평균 시간 복잡도 O(log n)으로 자료가 아주 클 때 유용
- 단, 최초 주어지는 리스트는 정렬이 되어 있어야 한다

```
def BinarySearch(a_list, key):
    start = 0
    end = len(a_list)-1
    while start <= end:
        mid = (start+end)//2
        if a_list[mid] == key:
            return True
        elif a_list[mid] > key:
            end = mid-1
        else:
            start = mid+1
    return False
```



예제 1

정수가 각각 하나씩 적혀 있는 숫자 카드 N개를 가지고 있다. 정수 M개가 주어졌을 때, 이 수가 적혀 있는 숫자 카드가 몇 개 있는지 구하는 프로그램을 만들어 보기 바랍니다.

```
10
6 3 2 10 10 10 -10 -10 7 3
8
10 9 -5 2 3 4 5 -10
```

첫째 줄: 숫자 카드의 개수 N
둘째 줄: 숫자 카드에 적혀 있는 정수 N
셋째 줄: M
넷째 줄: M개의 정수

```
[3, 0, 0, 1, 2, 0, 0, 2]
```

브루트 포스(Brute Force)

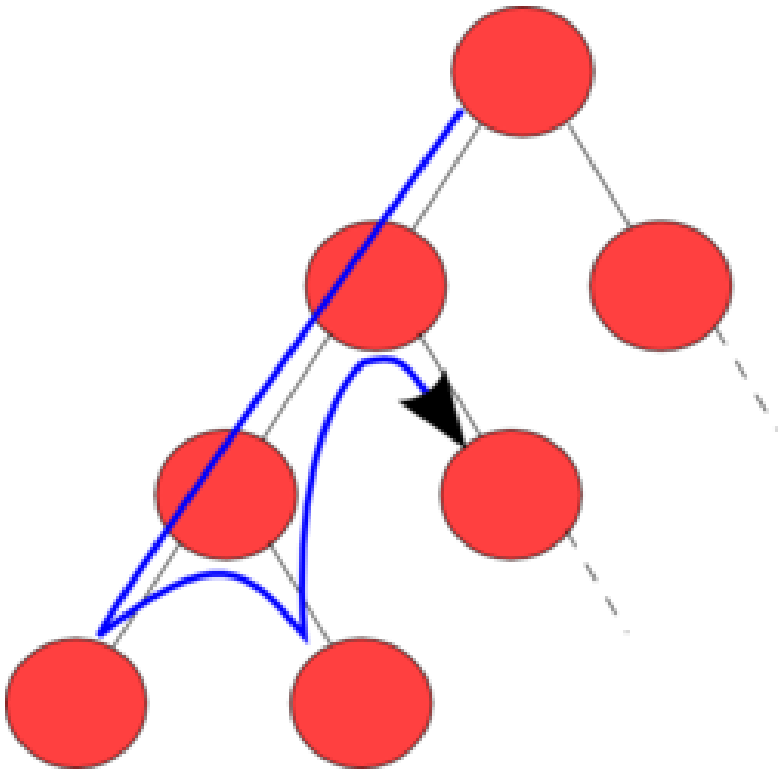
- Brute: 무식한, Force: 힘
- 모든 경우를 전체 탐색하는 알고리즘
- 정답을 확실하게 찾을 수 있는 장점이 있지만, 비효율적, 오래 걸리는 알고리즘

```
def sum_of_divisor(n):
    s = 0
    for i in range(1,n+1):
        if n%i == 0:
            s = s+i
    return s
```

n의 약수들의 합

1	2	3	4	5	6	7	8

깊이 우선 탐색(DFS, Depth-first search)



- 완전 탐색 방법 중 하나
- 탐색 트리의 수직방향으로 점차 깊은 곳까지 목표 노드를 참자 탐색해 나가는 기법(backtracking 과정 존재)
- 저장공간의 수요가 비교적 적다.
- 해가 없는 경로에 빠질 가능성이 있다.

백트래킹(Backtracking)

- 해를 찾아가는 도중, 지금의 경로가 해가 될 것 같지 않으면 더이상 가지 않고 되돌아 가는 기법
- 모든 가능한 경우의 수 중에서 특정한 조건을 만족하는 경우만 살펴보는 것
- 주로 DFS 등으로 모든 경우를 탐색하면서, 조건문 등을 걸어 답이 나오지 않는 상황을 정의하고, 그러한 상황일 경우에 탐색을 중지시킨 뒤 그 이전으로 돌아가서 다시 다른 경우를 탐색
- 유망하다(promising): 해가 될 가능성이 있다
- 가지치기(pruning): 유망하지 않은 노드에 가지 않는 것

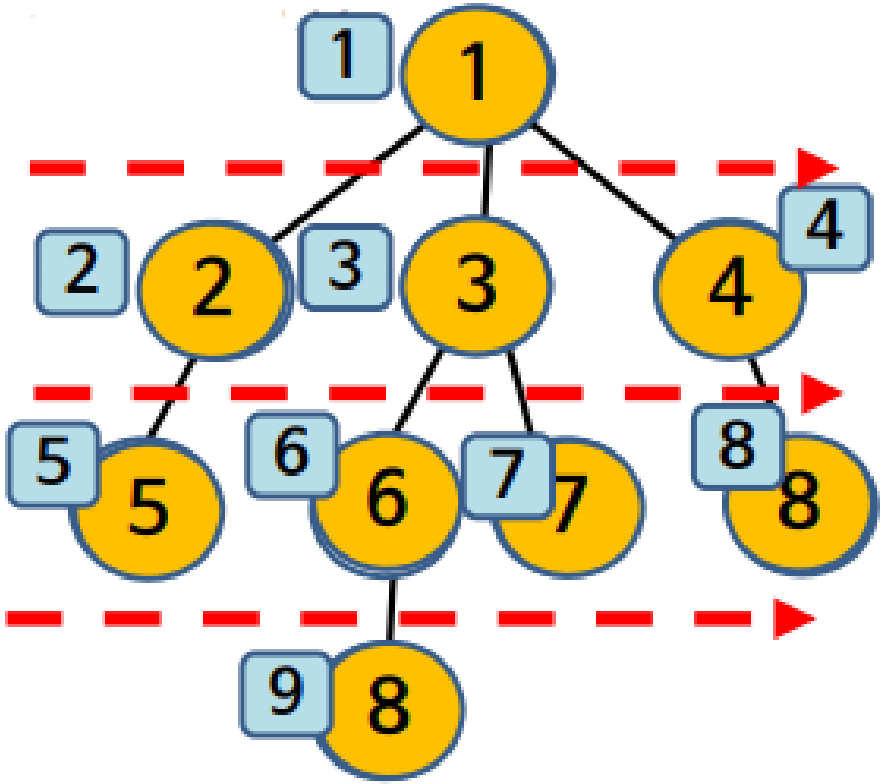
예제 2

자연수 N, M이 주어졌을 때, 1부터 N까지 자연수 중에서 중복 없이 M개를 고른 수열을 모두 구하는 프로그램을 작성하시기 바랍니다. 단, 수열은 오름차순으로 합니다.

```
4 2
```

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

너비 우선 탐색(BFS, Breadth-first search)



- 완전 탐색 방법 중 하나
- 탐색 트리의 루트 노트부터 목표 노드를 만날 때까지 단계별로 횡방향으로 탐색을 진행해 나가는 방식
- 최적해를 보장한다.
- 경로가 길면 저장공간의 수요가 크다.


과제

8-1

N의 제곱근을 구해보기 바랍니다. 단, 제곱근이 정수가 아니라면 -1을 출력합니다.

입력 예시	출력 예시
100	10
10	-1

8-2(N-Queen)

N-Queen 문제는 크기가 N × N인 체스 판 위에 퀸 N개를 서로 공격할 수 없게 놓는 문제입니다.

N이 주어졌을 때, 퀸을 놓는 방법의 수를 구하는 프로그램을 작성하기 바랍니다.

입력 예시	출력 예시
8	92

8-3

나무 N개가 있을 때, 나무 M미터를 얻기 위한 절단기의 높이의 최댓값을 구하는 프로그램을 만들어 보시기 바랍니다.

예를 들어 절단기의 높이가 15이고 나무의 높이가 각각 20, 15, 10, 17이라 두면, 나무를 자른 뒤 높이는 15, 15, 10, 15가 되고 남은 길이는 5, 0, 0, 2로 총 7미터를 얻을 수 있습니다.

입력 예시

```
4 7
20 15 10 17
```

출력 예시

```
15
```

8-4

스도쿠를 푸는 프로그램을 만들어 보기 바랍니다. 빈칸은 0 으로 제시됩니다.

```
0 3 5 4 6 9 2 7 8
7 8 2 1 0 5 6 0 9
0 6 0 2 7 8 1 3 5
3 2 1 0 4 6 8 9 7
8 0 4 9 1 3 5 0 6
5 9 6 8 2 0 4 1 3
9 1 7 6 5 2 0 8 0
6 0 3 7 0 1 9 5 2
2 5 8 3 9 4 7 6 0
```

입력 예시

```
1 3 5 4 6 9 2 7 8
7 8 2 1 3 5 6 4 9
4 6 9 2 7 8 1 3 5
3 2 1 5 4 6 8 9 7
8 7 4 9 1 3 5 2 6
5 9 6 8 2 7 4 1 3
9 1 7 6 5 2 3 8 4
6 4 3 7 8 1 9 5 2
2 5 8 3 9 4 7 6 1
```

출력 예시