알고리즘의 효율성 분석

이충기

명지대학교
컴퓨터공학과

문제 1: 입력 크기와 기본 연산

```
maximum = A[0]
for (i = 1; i < N; i++)
    if (A[i] > maximum) maximum = A[i]
```

총 i가 1~N 까지 변할 때 N번 비교

기본 연산 모두 될수 있음. A[i]를 끄집어내는데 시간이 걸릴 수 있기 때문

N-1 번 비교됨

답: N

A라는 배열 크기

답: A라는 배열에서 최대값

Q: 위 알고리즘의 구하는 값, 입력 크기, 기본 연산과 시간복잡도를 적어라.

답: A[i] > maximum

비교하는 연산이 가장 많음

답: O(n)

기본 연산 반복이 n-1 이니까

문제 2: 이진 탐색

다음의 숫자들에 대해 50를 이진탐색으로 찾는 과정을 보여라.

오름차순

총 11개 정수

10, 20, 25, 35, 45, 55, 60, 75, 80, 90, 95

0 1 2 3 4 5 6 7 8 9 10

high

① low

② low

mid (25<50)

mid (55>50)

high

③ low/mid(35<50) high

④ low/high/mid

low==high "찾을 수 없다"

mid 결정 규칙

mid = low + (high-low)/2

① 같다? 찾음

② 해당 값보다 크다? 오른쪽 (low = mid+1)

③ 해당 값보다 작다? 왼쪽 (high = mid-1)

규칙에 따라 low와 high이동

① 배열 Score, Rank 선언
② 등수 변수 count 선언
③ Score의 i(0~N-1)에 대해 Score[i] j(0~N-1)의 j를 증가시켜 가며 Score[i]와 Score[j]를 비교
④ Score[j]가 더 크거나 Score[i]와 Score[j]가 같으면 count 증가
⑤ Score[i]의 비교가 끝났다면 count를 Rank[i]에 저장하고 count를 0으로 초기화,
⑥ i가 N-1 까지 다 돈 후에는 종료

문제 3: 등수 매기기

- 점수들을 값이 큰 순서대로 등수를 매겨 보자. 예를 들면 5명의 점수가 82, 75, 98, 63, 40이면 등수는 2등, 3등, 1등, 4등, 5등이다. 점수들은 N개(> 0)이다. 점수들

```java
public final class App {
    public static void main(String[] args){
        int count = 0;
        int Score[] = {82, 75, 98, 63, 40};
        int Rank[] = {1,2,3,4,5};

        for(int i = 0; i < 5; i++){
            for(int j = 0; j < 5; j++){
                if(Score[i] <= Score[j]){
                    count++;
                }
            }
            Rank[i] = count;
            count = 0;
        }

        for(int i = 0; i < 5; i++){
            System.out.println(Rank[i] + " ");
        }
    }
}
```