알고리즘의 설계와 분석

2020년도 2학기 최 희석

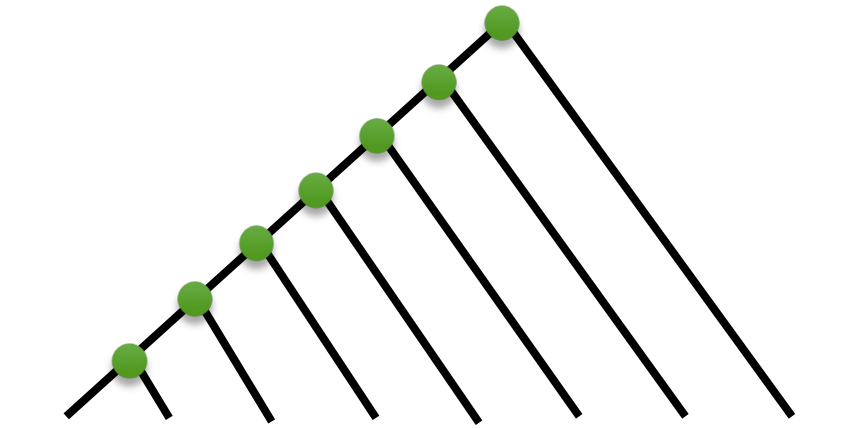
목차

- 알고리즘의 설계
  - 최소값 찾기
  - (정렬 VS 미 정렬) 된 카드에서 원하는 카드 찾기
  - 알고리즘 설계 기법
- 알고리즘의 분석
  - 알고리즘의 정확성, 효율성 분석
  - 시간 복잡도
  - 점근 성능

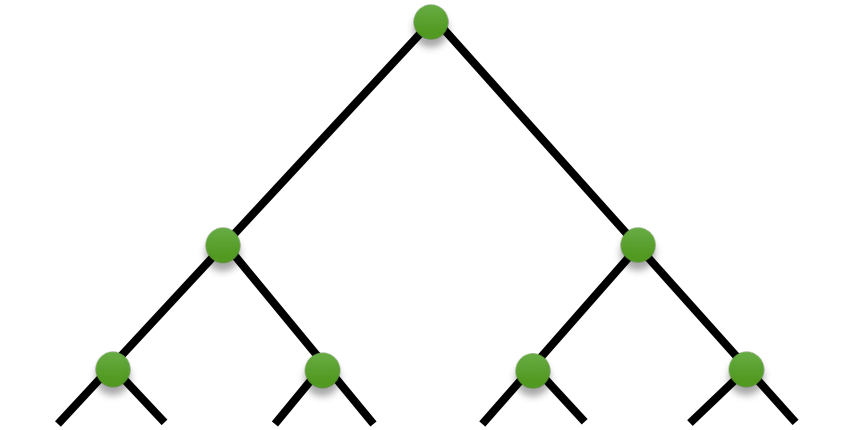
알고리즘의 설계

최소 값 찾기

효율적 알고리즘이란



알고리즘1



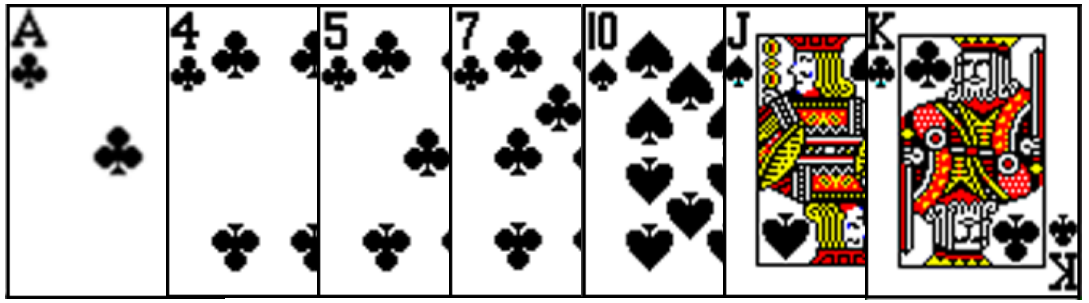
알고리즘2

최소값 찾기에서 알고리즘1과 알고리즘2 중에서 어떤 것이 더 효율적인가?

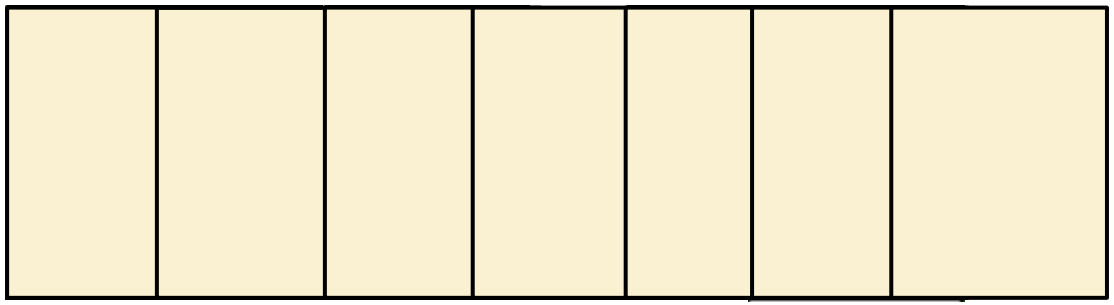
(정렬 VS 미 정렬) 된 카드에서 원하는 카드 찾기

효율적 알고리즘이란

미 정렬된 카드 중에서 원하는 카드 찾기



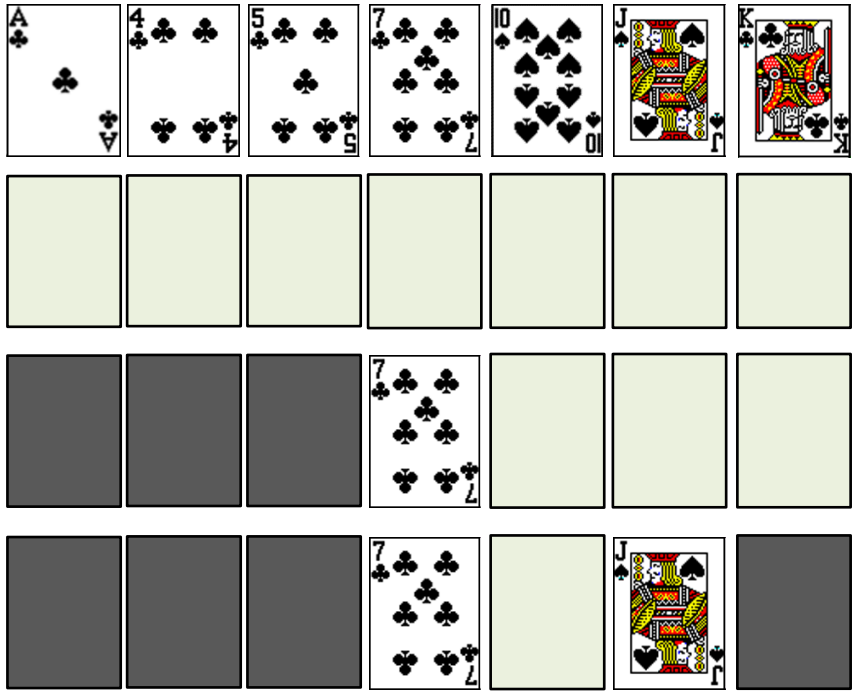
카드를 섞어서 뒤집어 놓았다. 이 카드 중에서 K를 찾아라!



순차 탐색

효율적 알고리즘이란

- 순서대로 정렬된 카드에서 원하는 카드 찾기



이진 탐색

알고리즘 설계 기법

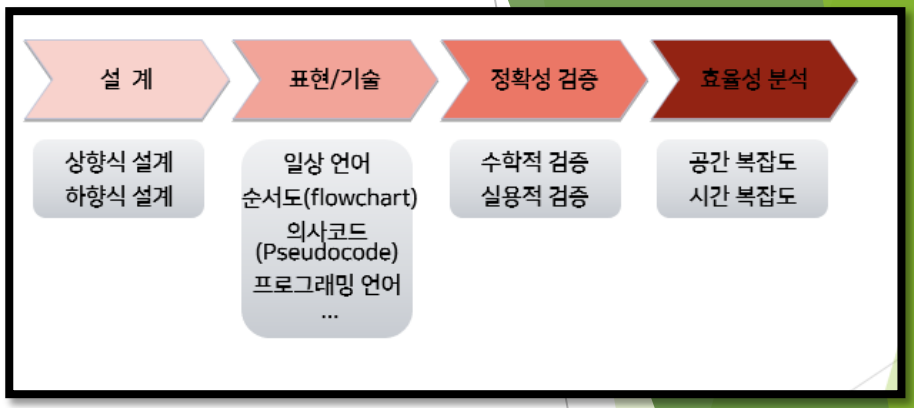
설계 기법

- 순차탐색 (미 정렬) vs 이진탐색 (정렬)
- 문제와 제반 조건이 매우 다양
  - 일반적인 기법은 없음
- 대표적인 설계 기법
  - 분할정복 방법
  - 욕심쟁이 방법
  - 동적 프로그래밍 방법

알고리즘의 분석

알고리즘의
정확성, 효율성 분석

정확성 vs 효율성



- 정확성 분석
  - 유효한 입력, 유한 시간 → 정확한 결과 생성
    - 다양한 수학적 기법을 사용한 이론적 증명이 필요
    - 다양한 입력 상황을 가정한 테스트 데이터를 통한 실용적 방법 → 제한적
- 효율성 분석
  - 알고리즘 수행에 필요한 컴퓨터 자원의 양을 측정
  - 공간 복잡도 (Space Complexity) → 메모리 양
  - 시간 복잡도 (Time Complexity) → 수행 시간

시간 복잡도

시간 복잡도

- 알고리즘의 수행시간
  - 실제 수행되는 시간을 측정하지 않음 → 컴퓨터 및 프로그래밍 언어 등에 의존
  - 알고리즘에 사용된 단위 연산들의 수행 횟수의 합
- 입력의 크기
  - 단순히 단위 연산의 개수가 아닌 입력 크기의 함수로 표현
- 입력 데이터의 상태에 따라 달라짐
  - 평균 수행 시간, 최선 수행 시간 , 최악 수행 시간

시간 복잡도

```
SumAverage(A[0..n-1], n)
{
1   i = 0;
2   sum = 0;
3   while ( i < n ) {
4     sum = sum + A[i];
5     i = i + 1;
    }
6   mean = sum / n;
}
```

1
1
n+1
n
n
1

f(n) = 3n+4 ⋯▸ O(n)

점근 성능

점근 성능

- 입력 데이터의 크기 n이 충분히 커짐에 따라 결정되는 성능

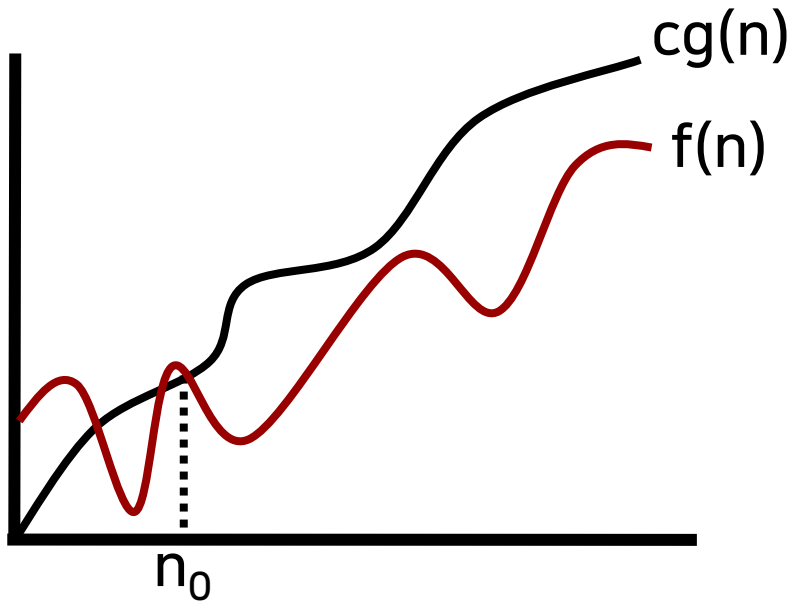
	$f_1(n)=10n+9$	$f_2(n)=n^2/2+3n$
n=5	59	27.5
n=10	109	80
n=15	159	157.5
n=16	169	176
n=20	209	260

- 다항식의 수행 시간에서 가장 큰 영향을 미치는 것은? → 최고차항

점근 성능의 표기법

정의 1 ‘Big-oh’ 점근적 상한

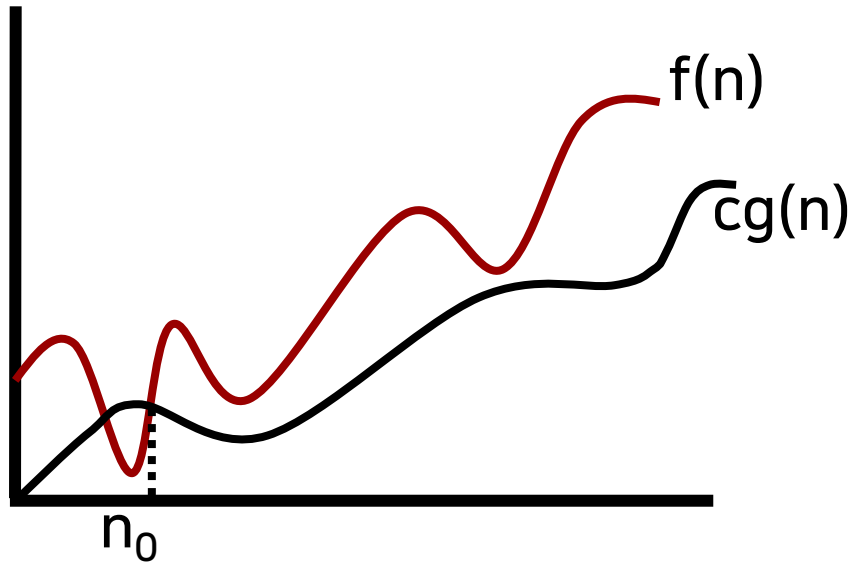
- 어떤 양의 상수 c와 n_0이 존재하여 모든 $n \geq n_0$에 대하여 $f(n) \leq c \cdot g(n)$이면 $f(n)=O(g(n))$이다.



점근 성능의 표기법

정의 2 ‘Big-omega’ 점근적 하한

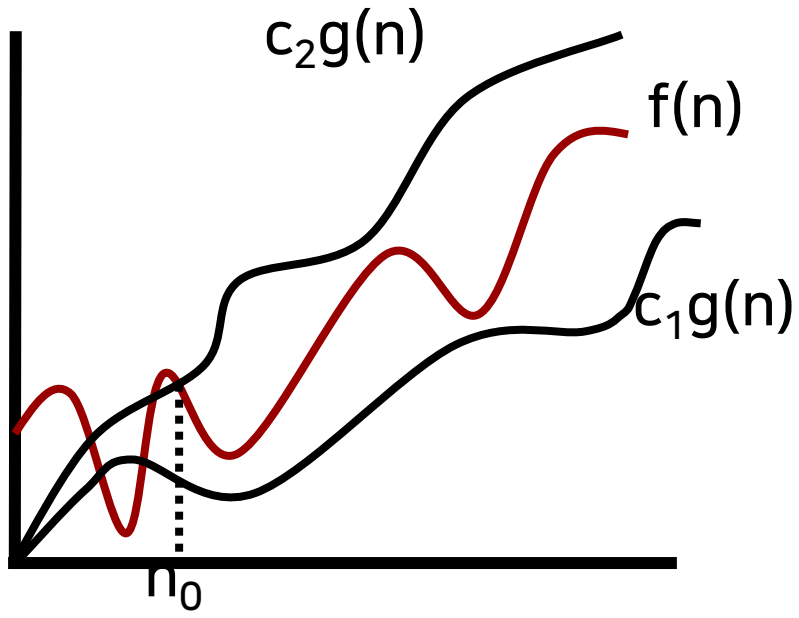
- 어떤 양의 상수 c와 n_0이 존재하여 모든 $n \geq n_0$에 대하여 $f(n) \geq c \cdot g(n)$이면 $f(n)=\Omega(g(n))$이다.



점근 성능의 표기법

정의 3 ‘Big-theta’ 점근적 상하한

- 어떤 양의 상수 c와 n_0이 존재하여 모든 $n \geq n_0$에 대하여 $c1 \cdot g(n) \leq f(n) \leq c2 \cdot g(n)$이면 $f(n)=\Theta(g(n))$이다.



점근 성능의 표기법

정의 1 'Big-oh' 점근적 상한

어떤 양의 상수 c와 n_0이 존재하여 모든 $n \geq n_0$에 대하여 $f(n) \leq c \cdot g(n)$이면 $f(n)=O(g(n))$이다.

- $f(n)=3n+3$, $g(n)=n$
  - $n \geq 2$에 대해서 $3n+3 \leq 5 \cdot n \rightarrow f(n)=O(g(n))=O(n)$
  - $n \geq 1$에 대해서 $3n+3 \geq 3 \cdot n \rightarrow f(n)=\Omega(g(n))=\Omega(n)$
  - $f(n)=O(n)$ and $f(n)=\Omega(n) \rightarrow f(n)=\Theta(n)$
- $f(n)= 2n^3+3n^2-n+10$ $\rightarrow O(n^3)$

점근 성능의 표기법

- 주요 함수 간의 연산 시간의 크기 관계

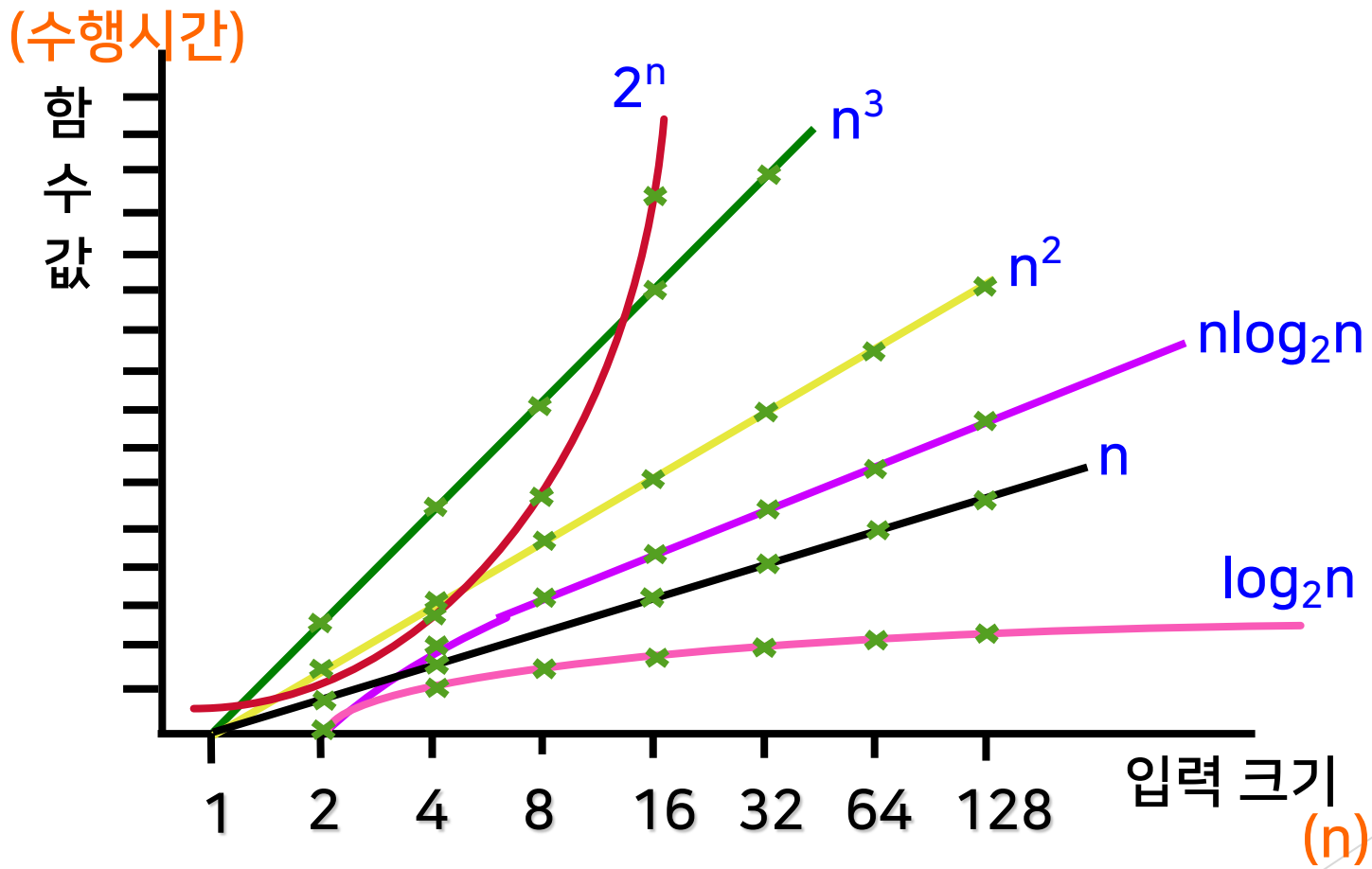
$O(1) < O(\log n) < O(n) < O(n\log n) < O(n^2) < O(n^3) < \cdots < O(2^n)$

효율적 ⇦ ⇨ 비 효율적

$10n+9$ → $O(n)$

$n^2/2+3n$ → $O(n^2)$

입력 크기에 따른 연산 시간의 증가 추세



시간 복잡도 사용 예

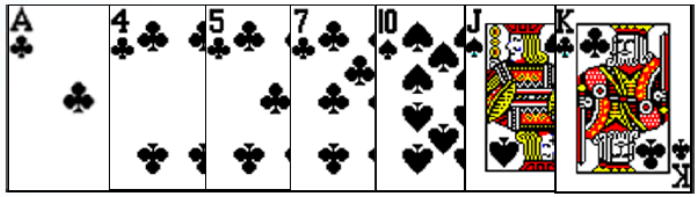
W, Y and want to compute $M = WY^{-1}$. There is another way to do the same computation, which changes its time complexity from $O(n^{\omega})$ to $O(n^{\omega_1})$. Indeed, we consider $M^{-1} = YW^{-1}$. If W is chosen smartly, then the product YW^{-1} can be computed in $O(n^2)$ time instead of $O(n^{\omega})$ time. Indeed, this happens if W is diagonal, or more generally, if W is the product of at most $O(n)$ elementary matrices (see [45] for a definition). Indeed, if this is the case, then W^{-1} is also a product of at most $O(n)$ elementary matrices, and taking a product with an elementary matrix only takes $O(n)$ time. In this case, we want to invert the matrix $M^{-1} = YW^{-1}$, but it is positive-definite. Thus we can invert it using $O(n^{\omega_1})$ operations instead of $O(n^{\omega})$ operations. We show below that W is the product of at most $O(n)$ elementary matrices.

과제 안내

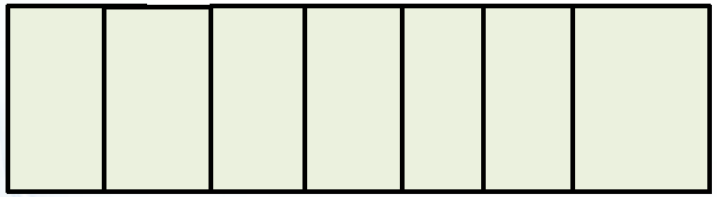
순차 탐색

```
SequentialSearch(a,n,key)
Begin
   for i = 0 to n-1 by 1 do
      if a[i] == key then
         return i;    //returning index of the array
      end if
   end for
   return -1;  //key not found
End
```

◆ 뒤섞인 카드에서 원하는 카드 찾기

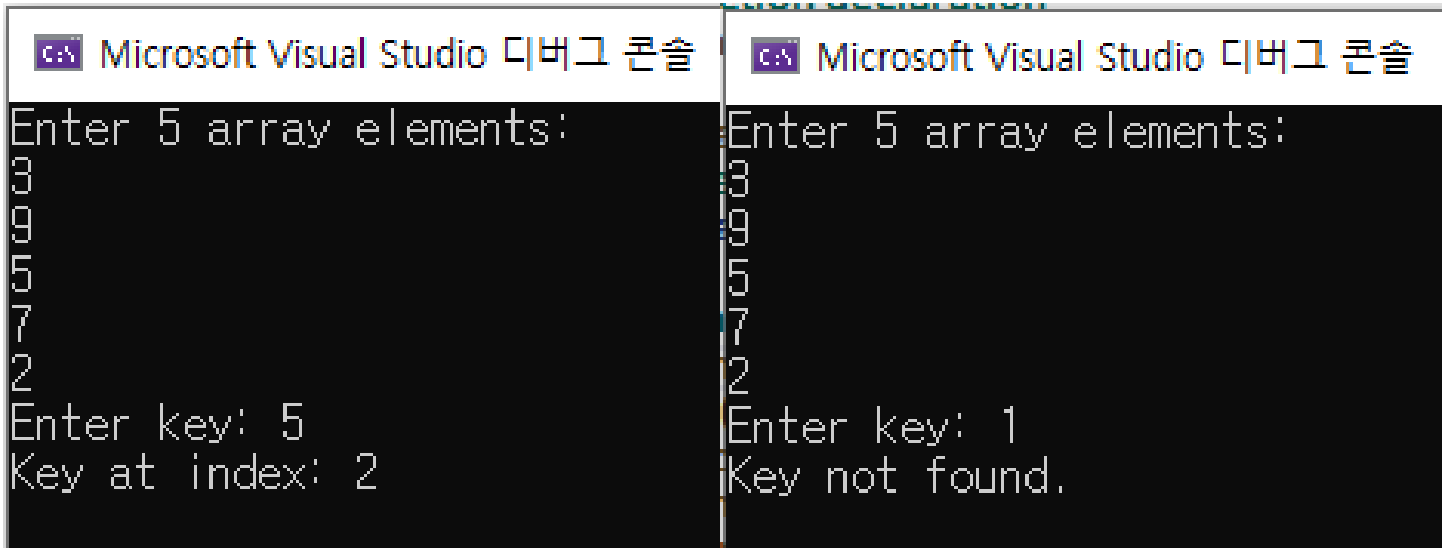


카드를 섞어서 뒤집어 놓았다. 이 카드 중에서 K를 찾아라!



순차 탐색

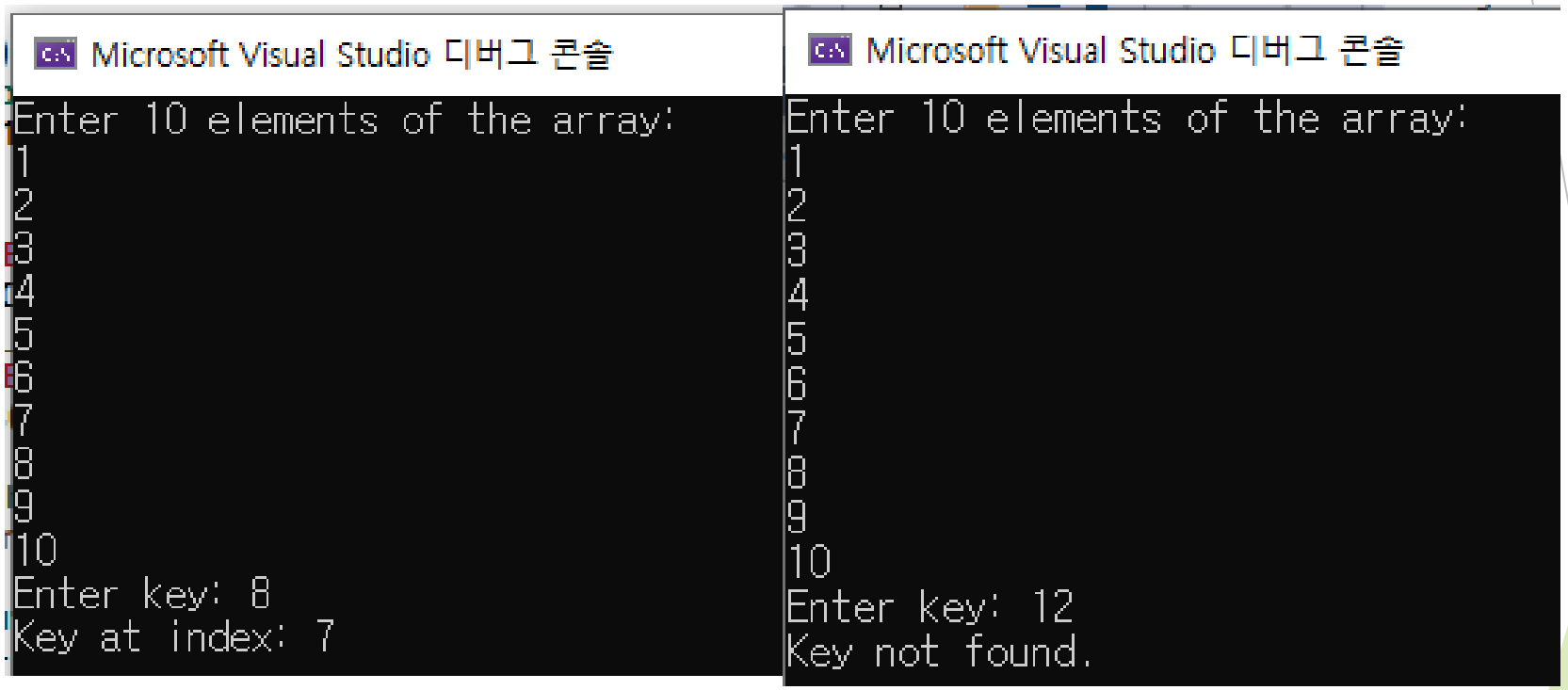
순차 탐색


Microsoft Visual Studio 디버그 콘솔
Enter 5 array elements:
3
9
5
7
2
Enter key: 5
Key at index: 2
Microsoft Visual Studio 디버그 콘솔
Enter 5 array elements:
3
9
5
7
2
Enter key: 1
Key not found.

이진 탐색

```
BinarySearch(a, n, key)
Begin
    Set start = 0, end = n-1, mid = (start + end)/2;
    while( start <= end && a[mid] != key) do
        if (key < a[mid]) then
            Set end = mid – 1;    //search in the left half
        else
            Set start = mid + 1;    //search in the right half
        end if
        Set mid = (start + end)/2;
    end while
    if(start > end)
        return -1;        //key not found
    return mid;        //returning key index
End
```

이진 탐색


Microsoft Visual Studio 디버그 콘솔
Enter 10 elements of the array:
1
2
3
4
5
6
7
8
9
10
Enter key: 8
Key at index: 7
Microsoft Visual Studio 디버그 콘솔
Enter 10 elements of the array:
1
2
3
4
5
6
7
8
9
10
Enter key: 12
Key not found.

2주차 과제

- 순차탐색, 이진탐색 구현하기
  - e-Class 업로드
- 양식 (한글, 워드,PDF -> 자유)
- 파일명 (이름_학번_전공)
  - 예) 최희석_2014182009_게임공학

- 질의 응답은 e-Class 질의응답 게시판에 남겨 주시길 바랍니다.