4주차 : Quick Sort

알 고 리 즘

2015. 09. 24.

충남대학교 컴퓨터공학과 임베디드 시스템 연구실
TA 권진세

Overview

▶ Quick Sort

- Quick Sort란?
- Quick Sort의 내부 정렬 과정
- 그 밖의 Partition 알고리즘

▶ 실습 / 과제

1) **기본적인 Partition 알고리즘을** 이용한 Quick Sort 구현

2) **임의의 3개 원소의 중간값을** 이용한 Quick Sort 구현

Quick Sort

▶ Quick Sort란?

Divide-and-conquer를 기반으로 하는 **내부 정렬** 알고리즘.
분할, 정복, 결합을 수행하는 2개의 프로시저가 필요하다.

▷ 퀵 정렬이 구현된 재귀 프로시저

```
QUICKSORT(A, p, r)
  if p < r
    then q ← PARTITION(A, p, r)
      QUICKSORT(A, p, q–1)
      QUICKSORT(A, q+1, r)
```

Quick Sort

▶ Quick Sort의 내부 정렬 과정

내부 정렬을 담당하는 **PARTITION 프로시저**는
다음과 같은 순서로 동작한다. (교재에 기술된 알고리즘)

1) input parameter = A[p .. r], p, r

2) X = A[r]

3) 정렬 : **{Value ≤ X} {X} {Value > X}**

4) 반환 : **{X}의 인덱스**

Partition

- **Quick Sort의 내부 정렬 과정 (Partition)**

  PARTITION 프로시저의 초기 설정은 다음과 같다.

Partition

- **Quick Sort의 내부 정렬 과정 (Partition)**

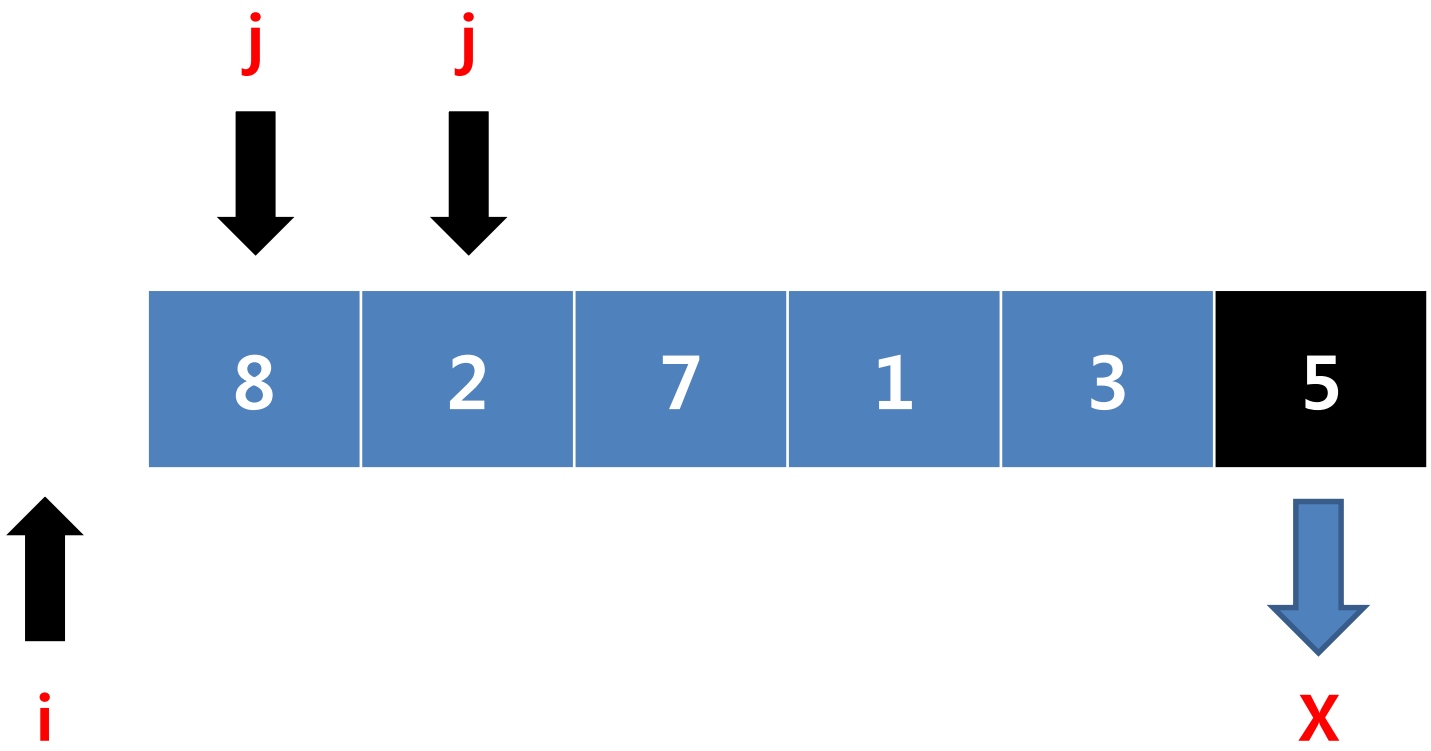
  반복문 안에는 하나의 조건문 **if A[j] ≤ X**만 존재한다.

for j = p to r-1

Partition

- **Quick Sort의 내부 정렬 과정 (Partition)**

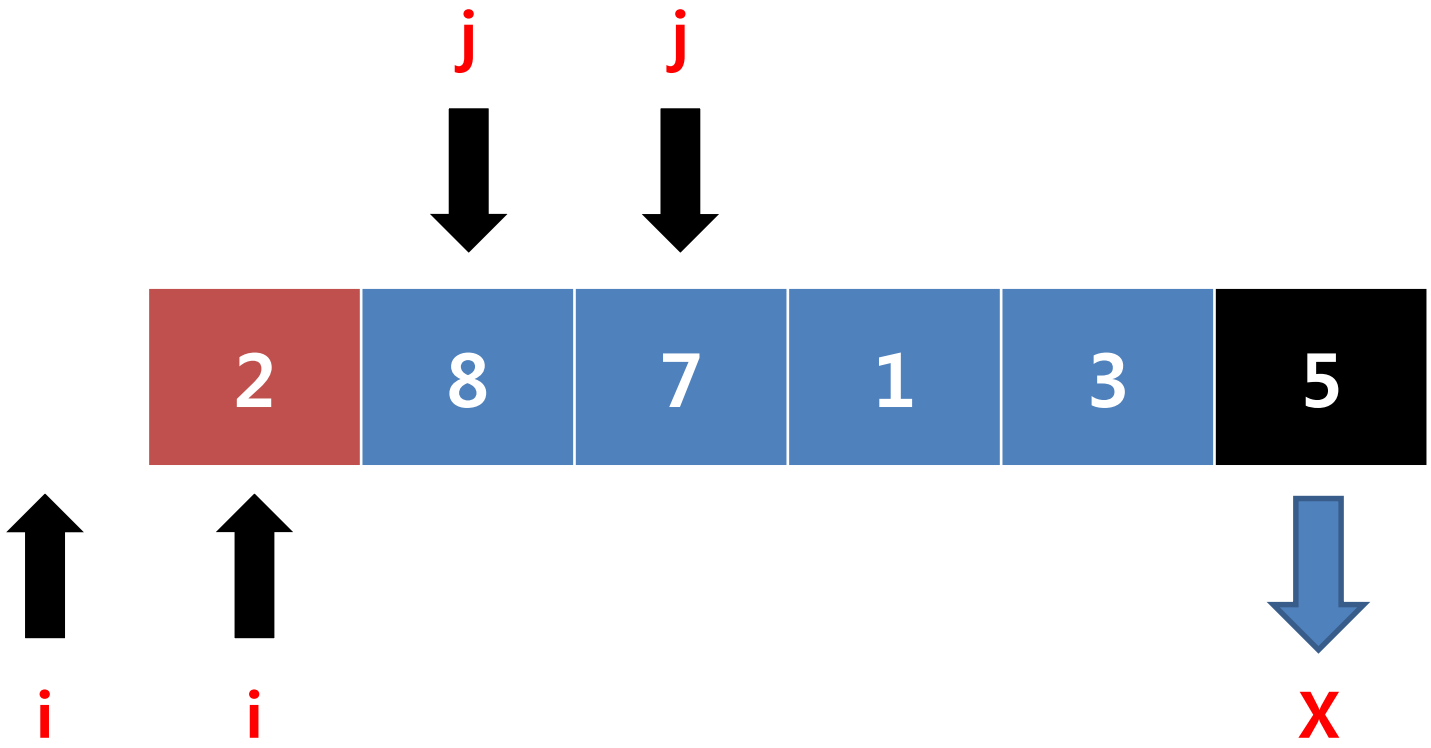
  조건문이 참일 때만 **i를 +1** 시키고, **A[i] ↔ A[j]**를 수행한다.



Partition

- **Quick Sort의 내부 정렬 과정 (Partition)**

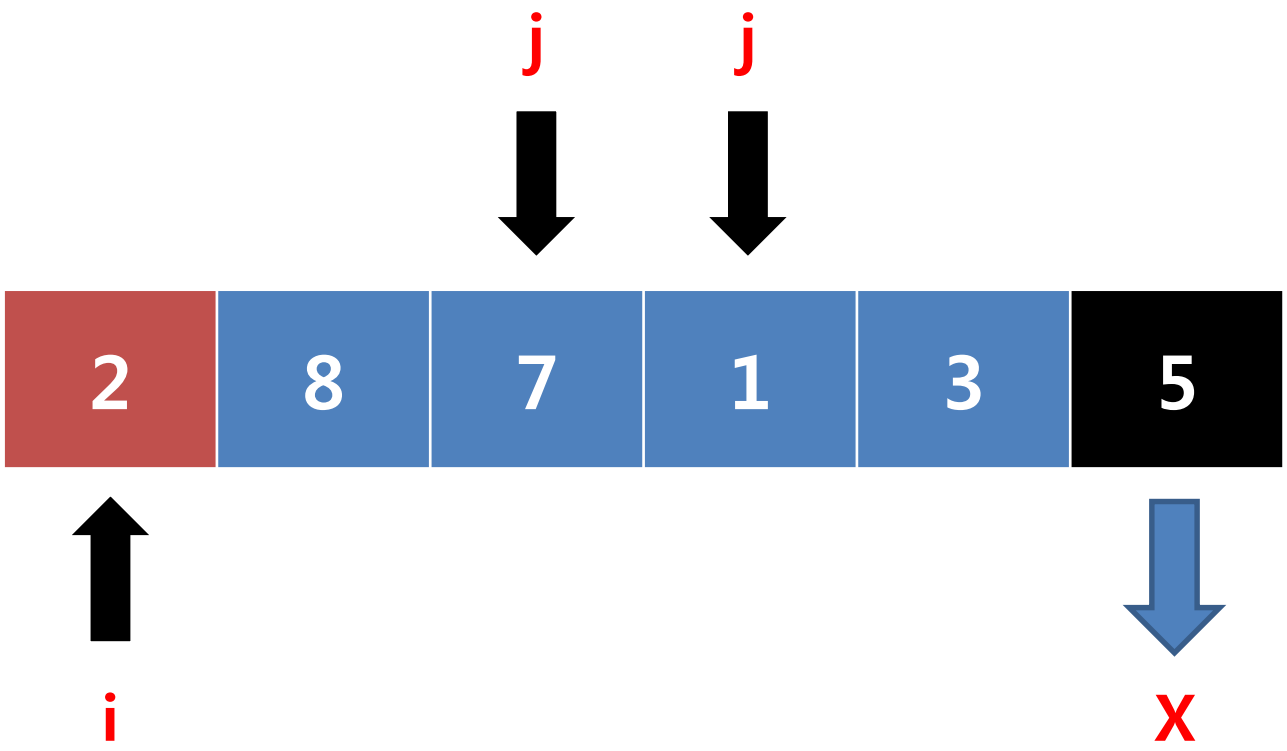
  조건문이 참일 때만 **i를 +1** 시키고, **A[i] ↔ A[j]**를 수행한다.



Partition

- **Quick Sort의 내부 정렬 과정 (Partition)**

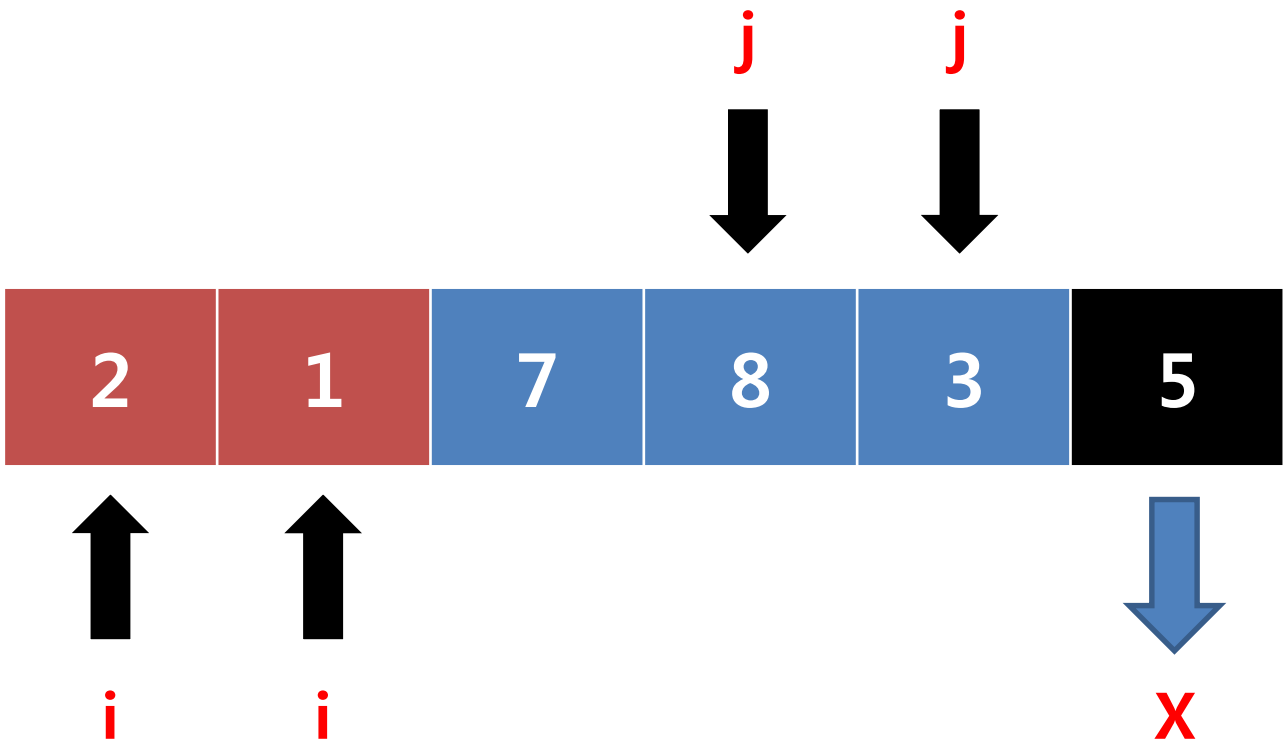
  조건문이 참일 때만 **i를 +1** 시키고, **A[i] ↔ A[j]**를 수행한다.



Partition

- **Quick Sort의 내부 정렬 과정 (Partition)**

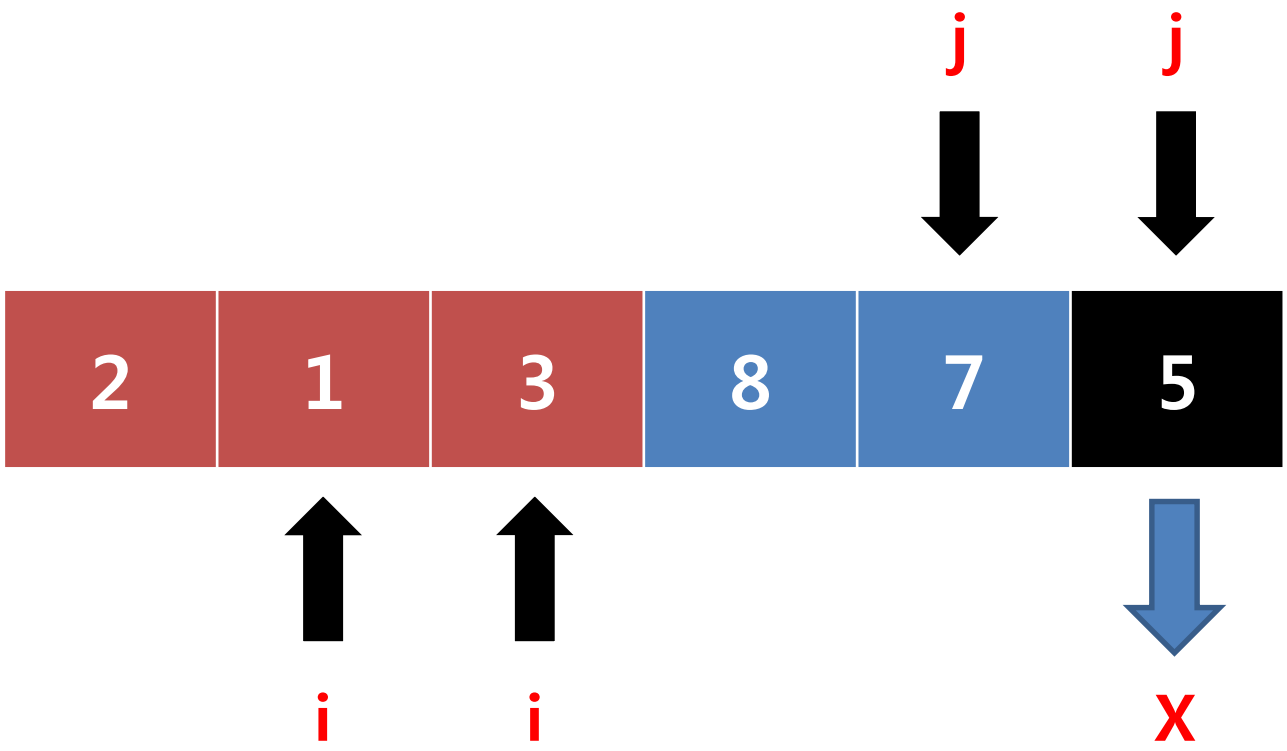
  조건문이 참일 때만 **i를 +1** 시키고, **A[i] ↔ A[j]**를 수행한다.



Partition

- **Quick Sort의 내부 정렬 과정 (Partition)**

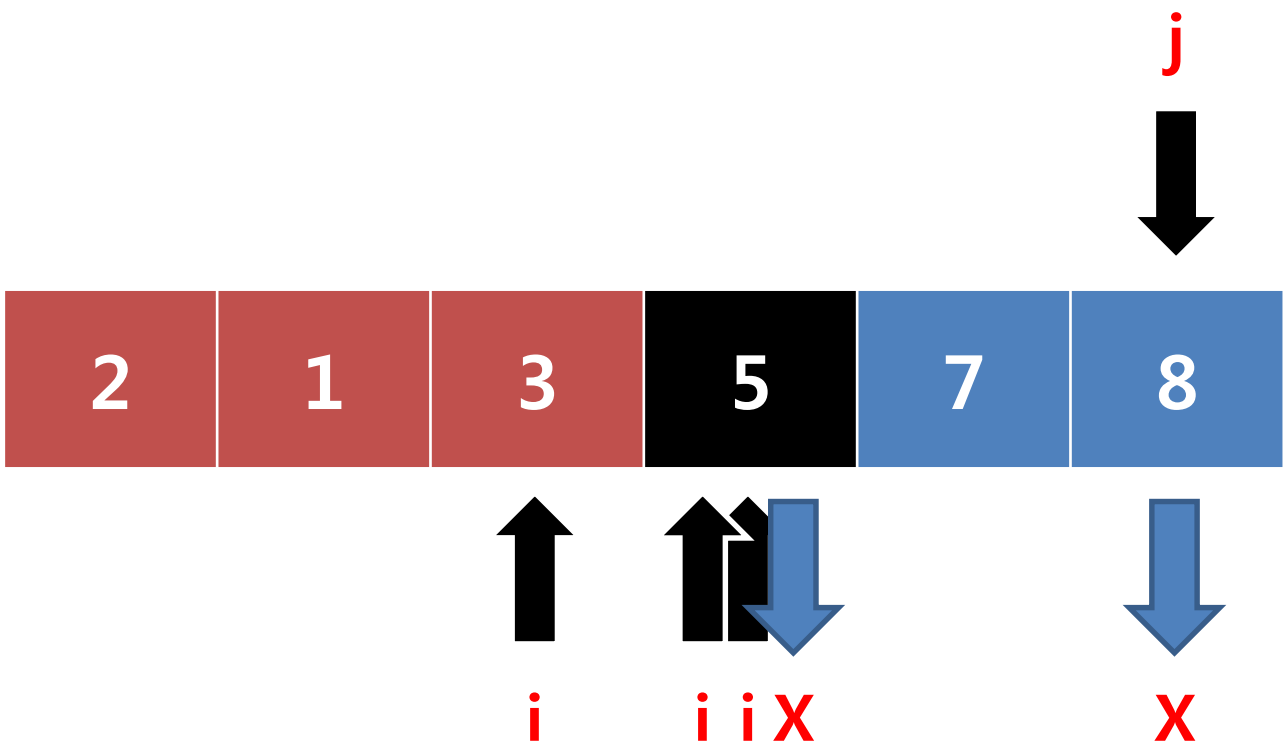
  조건문이 참일 때만 **i를 +1** 시키고, **A[i] ↔ A[j]**를 수행한다.



Partition

- **Quick Sort의 내부 정렬 과정 (Partition)**

  반복문이 종료되면 **i를 +1** 시키고, **A[i] ↔ A[j]**를 수행한다.



Partition

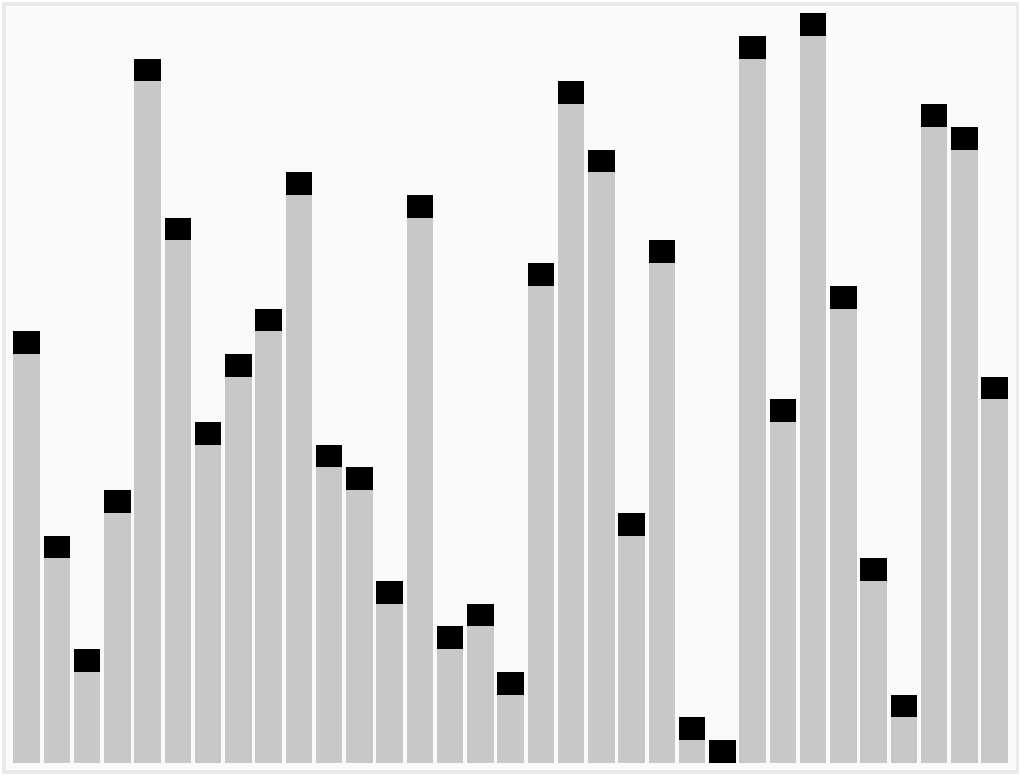
- **Quick Sort의 내부 정렬 과정 (Partition)**

  최종적으로 X가 위치한 곳의 인덱스 **i를 반환**한다.

  QUICKSORT 프로시저는 이를 반환 받은 후
  A[p .. i-1], A[i+1 .. r]로 각각 **재귀 호출**한다.

Partition

- Quick Sort 과정 GIF



Partition

```
PARTITION (A, p, r)
    x ← A[ r ]
    i ← p – 1
    for j ← p to r – 1
        do if A[ j ] ≤ x
            then i ← i + 1
                 A[ i ] ↔ A[ j ]
    i ← i + 1
    A[ i ] ↔ A[ r ]
    return i
```

```
RANDOMIZED-PARTITION (A, p, r)
    i = RANDOM( p , r )
    exchange A[r] with A[i]
    return PARTITION (A, p, r)
```

Practice / Homework

1. Quick Sort(A, p, r) 구현

2. Partition(A, p, r) 구현

3. Randomized-Partition(A, p, r) 구현

4. 보고서에 포함될 내용
   Normal Partition과 Randomized-Partition에서 QuickSort 성능 비교

과제 예시)

Input Data : 10, 9, 8, 7, 6, 5, 4, 3, 2, 1
Output Data : 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Practice / Homework

※ 그 외 실습 과제 수행 중 유의 사항

- 포함내용 : 코드와 보고서만 제출 (압축하지 마세요)
  ※ 매주 실습 보고서도 함께 제출할 것
  ※ 201500000_04.c 파일 함께 압축 후 전송

- 제출이름 :
  메일 : [알고리즘00반]_201500000_홍길동_4주차
  파일 : 201500000_04.c / 보고서_201500000_홍길동_04주차.hwp

- 제출기한 : 2015-10-01 18:00까지

- 메일주소 : kwonse@cnu.ac.kr

APPENDIX 1. File I/O

1. 파일 입출력 방법

```
FILE* fp;          //fp : input file pointer
FILE* fop;         //fop : input file pointer

//파일 이름은 "00_201500000_insertion.txt"
//입출력 파일은 *.c 소스파일과 같은 폴더에 있어야 한다.
fp = fopen(FILENAME,"rt");                  //입력 파일 열기
fop = fopen(FILENAME2,"wt");                //출력 파일 열기

if (fp == NULL) {
        printf("**** Input File open error ****₩n");
        exit(1);
} //파일 없을 경우 예외처리로 프로그램 종료
```

APPENDIX 1. File I/O (계속)

```
while(!feof(fp)){
} //파일의 끝날때 까지 반복

fscanf(fp,"%d, ", &변수);        // ex) 123, 456,
                                 // int 값만 추출함 ', '는 제외됨

fprintf(fop, "%d", 출력할 값);  // 파일 출력 시 사용

fclose(fp);
```

APPENDIX 2. 배열 넘기기

1. Main 함수의 배열을 주소로 넘겨서 다룰 때 !! 이중포인터 사용
   - 장점 : 메모리 절약, 리턴 불필요.

```
#include <stdio.h>
#include <stdlib.h>

void user_malloc(int** num);

void main(void){
    int *ptr;
    user_malloc(&ptr);      //포인터 변수 ptr의 주소를 인자로 보냄.
    printf("%d₩n", *ptr);  //출력 값은 10 이다.
    return 0;
}

void user_malloc(int** num){
    *num = (int*)malloc(sizeof(int));
    (*num)[0] = 10;
}
```

APPENDIX 3. 동적 할당 메모리 크기

Q & A.

포인터로 받은 배열의 크기를 구하는 방법? (있다)

- Malloc 함수의 선언을 보면 void* malloc(size_t size)
- Size_t는 많은경우 unsinged long int로 되어있으므로
- 메모리를 할당 할 때 이 크기만큼 더 할당해서 할당 영역의 처음 부분에 길이의 값을 저장해 두고 있음.

- *(ptr - sizeof(size_t))
- **다음과 같은 함수로 만들어 사용 가능**

```
int sizeof_ar(int* S){
    int size;
    size = *(S - sizeof(int));
    return size;
}
```

APPENDIX 4. 중간 값 찾기

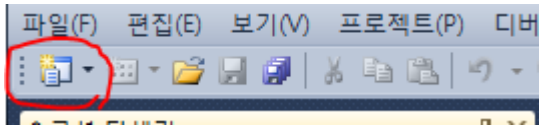
3개의 원소중에 중간 값을 찾는 방법

```
int iPivot;
int ptrCenter = (ptrLeft + ptrRight) / 2;
if(!(ptrLeft < ptrCenter ^ ptrCenter < ptrRight))
    iPivot = ptrCenter;
else if(!(ptrCenter < ptrLeft ^ ptrLeft < ptrRight))
    iPivot = ptrLeft;
else
    iPivot = ptrRight;
```

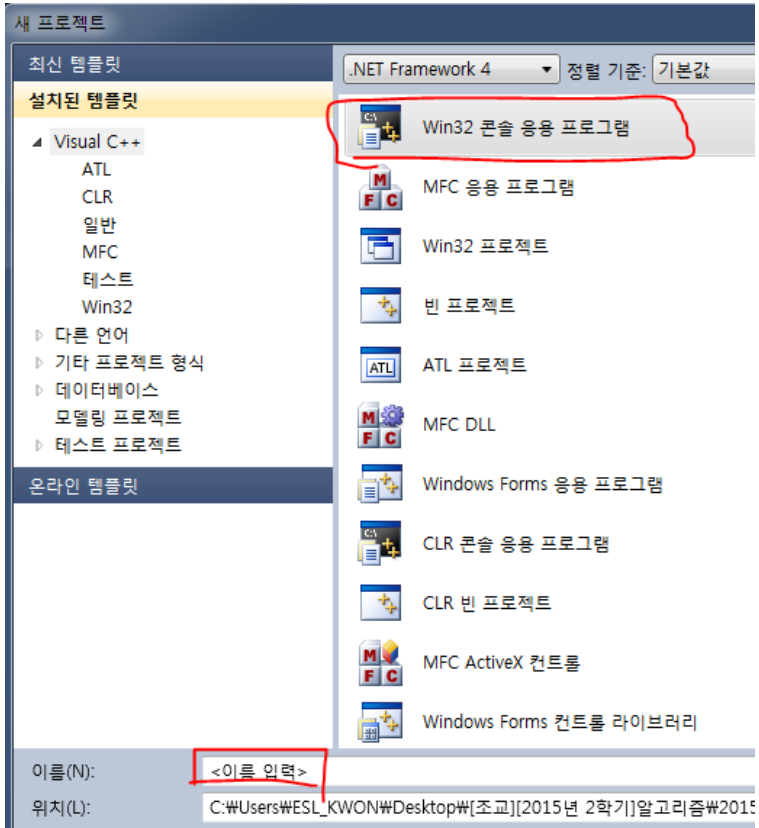
Trouble Shooting

Visual Studio 2010 프로젝트 생성

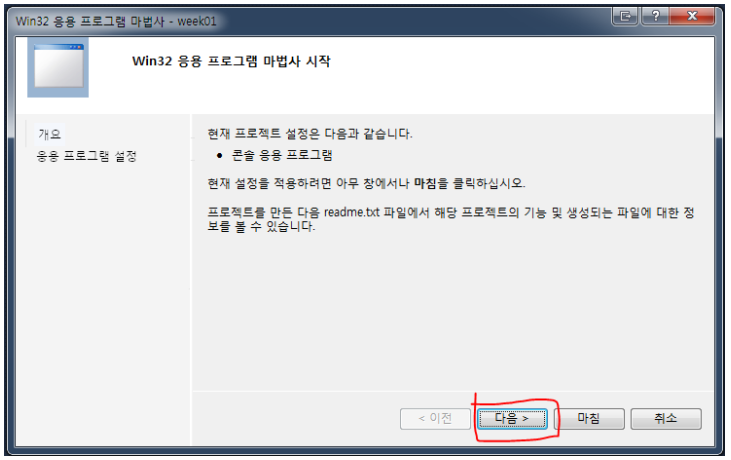
1.



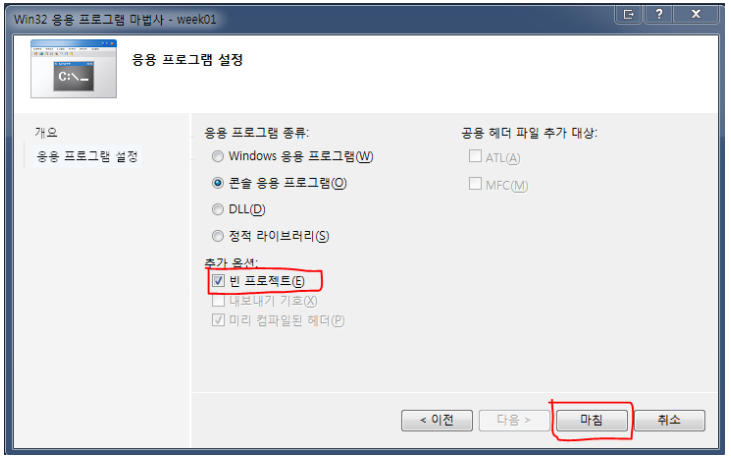
2.



3.



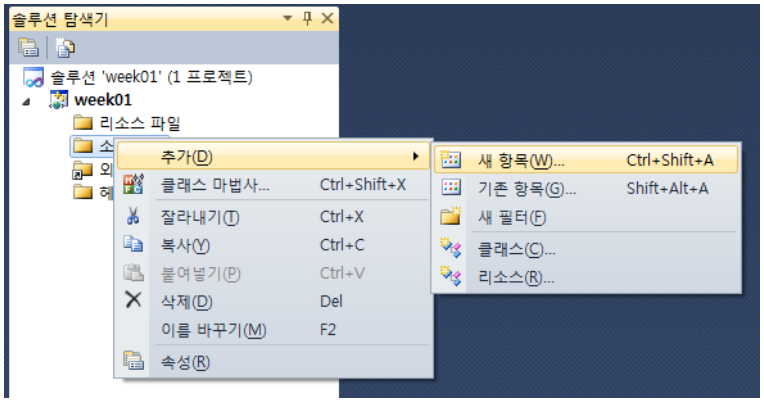
4.



Trouble Shooting

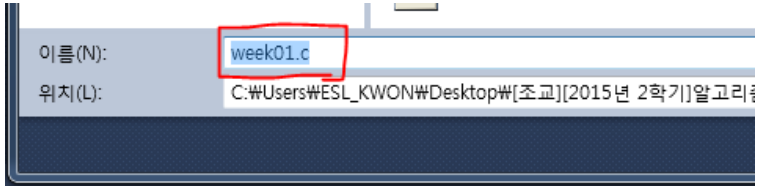
Visual Studio 2010 프로젝트 생성

5.



6.

7.



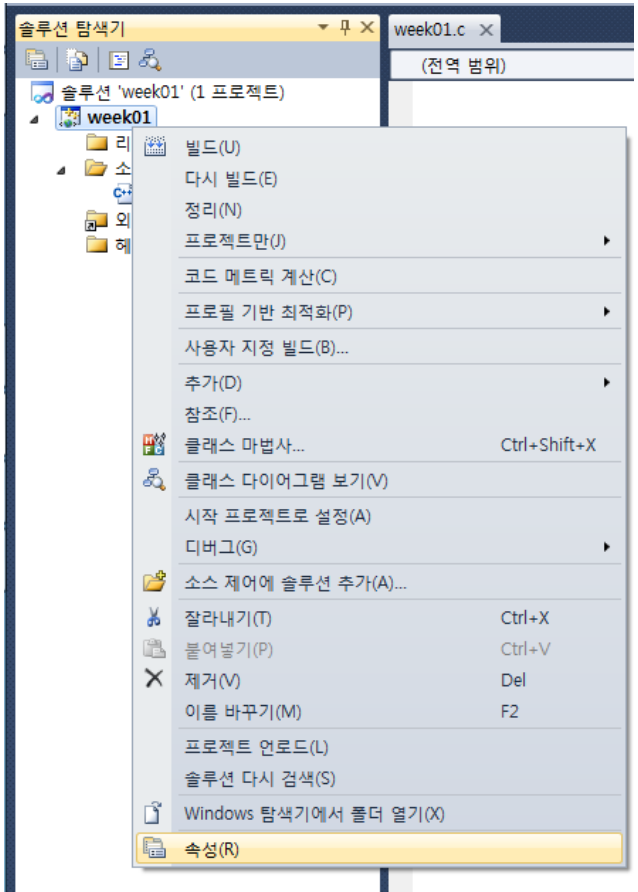
반드시 .c 로 이름 변경!!

Trouble Shooting

Visual Studio 2010 메니페스트 오류 해결

```
1>LINK : fatal error LNK1123: COFF로 변환하는 동안 오류가 발생했습니다. 파일이 잘못되었거나 손상되었습니다.
1>
1>빌드하지 못했습니다.
```

1.

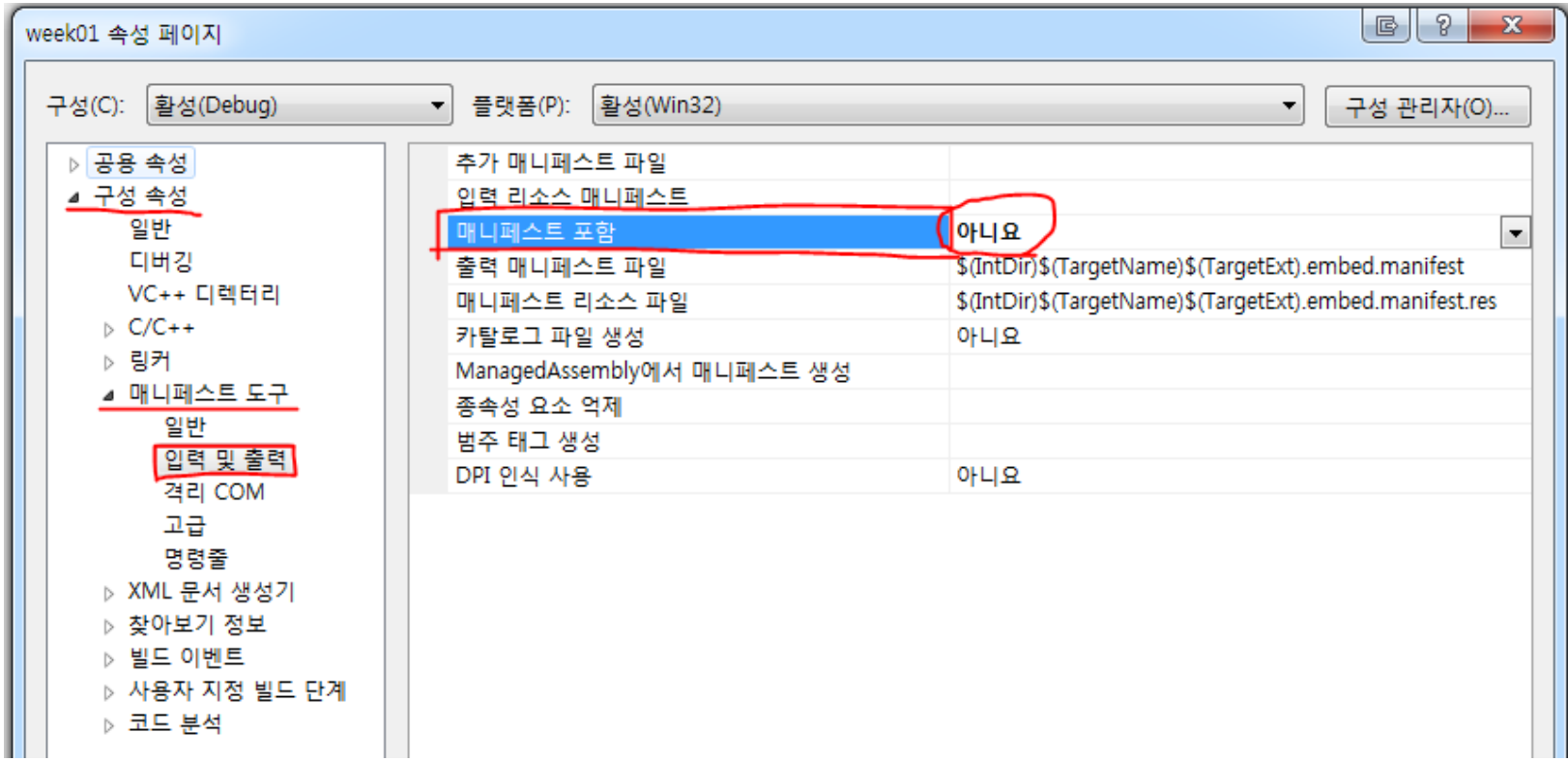


< 프로젝트 속성열기

Trouble Shooting

Visual Studio 2010 메니페스트 오류 해결

2.



구성 속성 -> 매니페스트 도구 -> 입력 및 출력 ->

메니페스트포함 : “아니요 "

Trouble Shooting

Visual Studio 2010 메니페스트 오류 해결
3. 매니페스트 문제 영구적 해결 방법

Visual Studio Service Pack 1 다운로드.
(>600MB 오래 걸림…)

https://www.microsoft.com/en-us/download/confirmation.aspx?id=23691