

선택정렬

백석대학교 강윤희



선택정렬

- 각 루프마다
 - 최대 원소를 찾는다
 - 최대 원소와 맨 오른쪽 원소를 교환한다
 - 맨 오른쪽 원소를 제외한다
- 하나의 원소만 남을 때까지 위의 루프를 반복



selectionSort(A[], n) ▷ 배열 A[1 ... n]을 정렬한다

```
{  
    for last ← n downto 2 { ----- ①  
        A[1 ... last] 중 가장 큰 수 A[k]를 찾는다; ----- ②  
        A[k] ↔ A[last]; ▷ A[k]와 A[last]의 값을 교환 ----- ③  
    }  
}
```

✓ 수행 시간:

- ①의 **for** 루프는 $n-1$ 번 반복
- ②에서 가장 큰 수를 찾기 위한 비교횟수: $n-1, n-2, \dots, 2, 1$
- ③의 교환은 상수 시간 작업

✓ $(n-1)+(n-2)+\dots+2+1 = \Theta(n^2)$



선택정렬

- 큰값을 찾아 마지막값으로 보낸다

```
void _SelectionSort(int A[], int size)
{
    int last = size - 1;
    int temp;
    for (; last >= 1; last--)
    {
        int k;
        k = theLargest(A, last);
        //printf("largerst =%d Wn", A[k]);
        temp = A[last];
        A[last] = A[k];
        A[k] = temp;
    }
}
```



선택정렬

```
void SelectionSort(int A[], int size)
{
    int last = size - 1;
    if (last == 0)
        return;
    else
    {
        int k;
        int temp;

        k = theLargest(A, last); // k는 배열 A의 큰값위치
        printf("largerst =%d \n", A[k]);
        printf("size =%d \n", last);
        temp = A[last];
        A[last] = A[k];
        A[k] = temp;
        SelectionSort(A, size - 1);
    }
}
```

```
Given array is
12 11 13 5 6 7
largerst =13
size =5
largerst =12
size =4
largerst =11
size =3
largerst =7
size =2
largerst =6
size =1
```

```
Sorted array is
5 6 7 11 12 13
```

