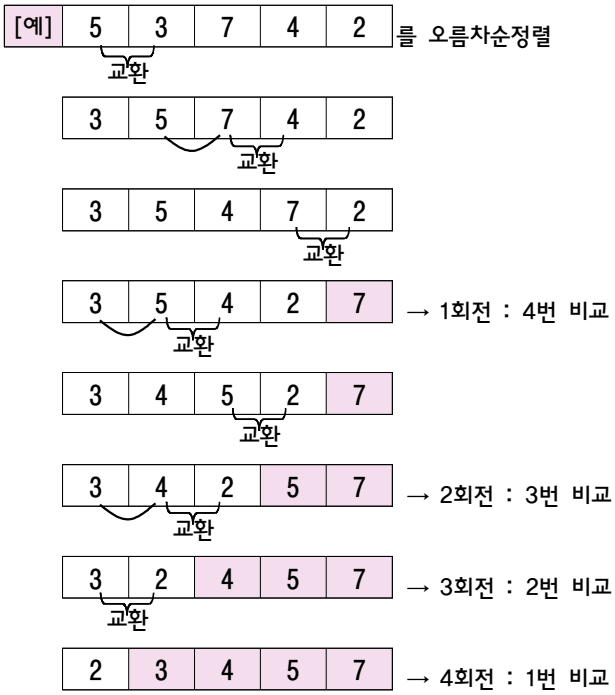
2. 거품정렬(bubble sort)

인접한 두 키의 크기를 비교하여 그 크기에 따라 교환하는 방식이다.



· 거품정렬에서 비교수

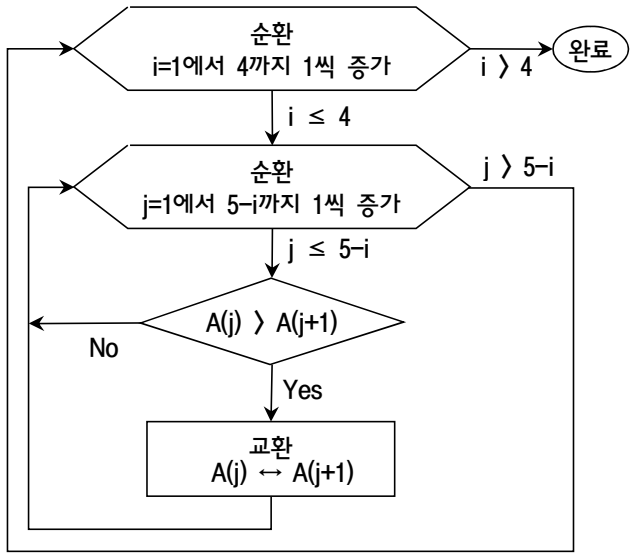
거품정렬은 5개의 자료를 정렬하기 위해서는 4단계의 처리 과정이 필요하고, 각 단계의 비교회수는 4, 3, 2, 1로 하나씩 줄어든다.

· 거품정렬 연산시간

$1 + 2 + 3 + \cdots + (n-1)$

$= \frac{n(n-1)}{2} = O(n^2)$

◈ **거품정렬 순서도(5개의 데이터를 오름차순정렬)**



2 http://cafe.daum.net/pass365(홍재연)

다음은 거품정렬을 C와 Python으로 구현한 것이다.

Bubble Sort(ascending) - C 언어	Bubble Sort(ascending) - Python

void bubble_sort(int a[], int cnt){
  int i, j, imsi, flag;
  for(i=0 ; i<cnt-1 ; i++){
  flag = 0;
  for (j=0 ; j<cnt-1-i ; j++)
  if(a[j] > a[j+1]){
  imsi = a[j],
  a[j] = a[j+1],
  a[j+1] = imsi;
  flag = 1; //자료 교환 중
  }
  if(flag==0) break; //도중 탈출
  }
}
 	
void main(){
  int a[] = {5, 3, 7, 4, 2};
  int i, cnt;
  cnt = sizeof(a)/sizeof(int);
  bubble_sort(a,cnt);
  printf("\n Result of Sort :");
  for(i=0;i<cnt;i++)
  printf("%4d",a[i]);
}
	// 변수 flag를 사용하는경우
	def bubble_sort(data): # 함수 정의
	  for i in range(len(data)-1): # i=0, 3, 1
	  flag = False
	  # j=0, 5-i-1, 1
	  for j in range(len(data)-i-1):
	  # 인접한 키 비교
	  if data[j] > data[j+1]:
	  temp = data[j] # 키 교환
	  data[j] = data[j+1]
	  data[j+1] = temp
	  flag = True # 키 교환 중
	  if not flag:
	  break # 도중 탈출
	data = [5, 3, 7, 4, 2] # 5개의 데이터 정렬
	bubble_sort(data)
	print(data) # 정렬된 결과 : [2, 3, 4, 5, 7]

◈ **거품정렬의 도중 탈출 – 플랙 변수 이용**

거품정렬에서는 플랙(flag) 변수를 이용할 수도 있고 하지 않을 수도 있는데, 플랙 변수를 이용할 경우에는 자료를 정렬하는 도중에 정렬이 끝난 상태에 도달하면(플랙 변수의 값으로 판단) 정렬 루틴을 탈출할 수 있다. 이렇게 되면 **n**개의 자료를 정렬하는 경우 무조건적으로 $\frac{n(n-1)}{2}$ 번을 비교하지 않아도 되고, **최악**의 경우에만 $\frac{n(n-1)}{2}$ 번을 비교하게 된다.

최선의 경우에는 **(n-1)**번만 비교하게 된다. - **O(n)**

거품정렬의 **도중 탈출**은 아주 **특별한 경우**에 발생되므로 알고리즘의 평균 연산시간이 개선되는 것은 아니다. **특별한 경우**에 발생되므로 플랙 변수를 사용하는 것도 바람직한 것은 아니다.

기출문제 분석

1. 다음 C 언어 함수에 의해 구현된 정렬 방식은? [2018년 국가 7급]

```
---------------------------------
 void whatsort(int a[ ], int size)
 {
   int i, j, temp;
   for (i = (size-1); i > 0; i--)
   {
     for (j = 1; j <= i; j++)
     {
       if (a[j-1] > a[j])
       {
         temp = a[j-1];
         a[j-1] = a[j];
         a[j] = temp;
       }
     }
   }
 }
---------------------------------
```

① 삽입정렬(insertion sort)
② 선택정렬(selection sort)
③ 힙정렬(heap sort)
④ 버블정렬(bubble sort)

♤ **버블정렬**

```
// 프로그램 분석
  if (a[j-1] > a[j])        // 인접한 배열요소를 비교하여 맞교환하는 원리(버블정렬)
  {
    temp = a[j-1];
    a[j-1] = a[j];
    a[j] = temp;
  }
```

정답 : ④

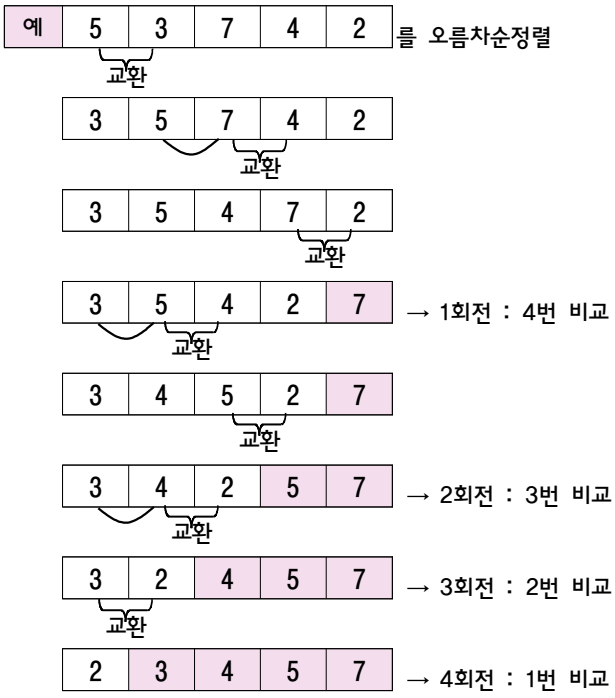
2. <보기>의 배열 A에 n개의 원소가 있다고 가정하자. 다음 의사코드에 대한 설명으로 가장 옳지 않은 것은? [2019년 서울 9급]

```
------<보기>-------------------------------------------------
Function(A[ ], n) {
  for last←n downto 2          //last를 n에서 2까지 1씩 감소
    for i←1 to last-1
      if(A[i]>A[i+1]) then A[i]↔A[i+1];  //A[i]와 A[i+1]를 교환
}
-----------------------------------------------------------------
```

① 제일 큰 원소를 끝자리로 옮기는 작업을 반복한다.
② 선택정렬을 설명하는 의사코드이다.
③ $O(n^2)$의 수행시간을 가진다.
④ 두 번째 for 루프의 역할은 가장 큰 원소를 맨 오른쪽으로 보내는 것이다.

♤ **정렬**

- **선택정렬**을 설명하는 의사코드이다.(×) → 주어진 의사코드는 **거품정렬**이다.
- **거품정렬**은 인접한 두 키의 크기를 비교하여 그 크기에 따라 교환하는 방식이다.



· 거품정렬에서 비교수

거품정렬은 5개의 자료를 정렬하기 위해서는 4단계의 처리 과정이 필요하고, 각 단계의 비교회수는 4, 3, 2, 1로 하나씩 줄어든다.

· 거품정렬 연산시간

$1 + 2 + 3 + \cdots + (n-1)$

$= \frac{n(n-1)}{2} = O(n^2)$

정답 : ②

3. 입력값으로 5, 2, 3, 1, 8이 주어졌을 때 버블정렬의 1회전(pass) 결과는? [2015년 서울 9급]

① 1, 2, 3, 5, 8 ② 2, 3, 1, 5, 8 ③ 2, 5, 3, 1, 8 ④ 8, 5, 3, 2, 1

♤ **버블정렬(bubble sort)**

// **버블정렬**은 **인접**한 두 키의 크기를 비교하여 그 크기에 따라 교환하는 방식이다.

5, 2, 3, 1, 8

2, 5, 3, 1, 8

2, 3, 5, 1, 8

2, 3, 1, 5, 8 → **1회전 결과** : 가장 큰 값이 **마지막**에 위치하게 된다.

정답 : ②

4. 다음은 C언어로 내림차순 버블정렬 알고리즘을 구현한 함수이다. ㉠에 들어갈 if문의 조건으로 올바른 것은? (단, size는 1차원 배열인 value의 크기이다) [2015년 국가 9급]

```
----------------------------------------
void BubbleSorting(int *value, int size)
{
  int x, y, temp;
  for(x = 0; x < size; x++)
  {
    for(y = 0; y < size - x - 1; y++) {
      if(    ㉠    ) {
          temp = value[y];
          value[y] = value[y+1];
          value[y+1] = temp;
      }
    }
  }
}
----------------------------------------
```

① value[x] 〉 value[y+1] ② value[x] 〈 value[y+1]

③ value[y] 〉 value[y+1] ④ value[y] 〈 value[y+1]

♤ **버블정렬 알고리즘 – 버블정렬은 인접한 자료를 비교하여 그 크기에 따라 교환**

- value[y] 〉 value[y+1] : 오름차순정렬
- value[y] 〈 value[y+1] : 내림차순정렬

정답 : ④

5. 데이터 수를 n이라고 할 때, 다음 정렬 알고리즘들에 대한 설명 중에서 옳지 않은 것은? [2004년 기술고시]

① 히프정렬은 안정적인(stable) 알고리즘이 아니다.
② 선택정렬은 입력 데이터가 정렬이 된 경우와 그렇지 않은 경우에 관계없이 비교횟수가 동일하다.
③ 퀵정렬의 최악 시간복잡도는 $O(n^2)$이다.
④ 히프정렬은 $O(1)$만큼의 추가적인 메모리를 사용한다.
⑤ 정렬되지 않은 인접한 데이터들의 맞교환을 반복하여 전체 데이터를 정렬하는 정렬 알고리즘들의 최악 시간복잡도는 $O(nlog_2n)$이다.

♤ **정렬 알고리즘**

- 항목 ⑤는 **인접**한 데이터들의 맞교환하는 **거품정렬**에 대한 설명이다. $O(n^2)$이다.
- 히프정렬에서 $O(1)$만큼의 추가적인 메모리는 자료 교환에 필요한 것이다.

정답 : ⑤