**System Programming 개인 과제**

미디어학과 201421104 허준영

|  |  |
| --- | --- |
| 201421104 | BUBBLE |

1. 알고리즘에 대한 개요

**Bubble sort**

- 배열을 제일 앞 index부터 순서대로 탐색하며, 첫번째 index의 배열 값과 그 다음 Index의 배열값을 비교하여 앞의 index의 배열 값이 더 큰 경우, 위치를 바꾸고 마지막 index까지 탐색을 완료하면 다시 처음부터 탐색하여 과정을 반복한다. 이때 배열의 길이만큼 반복하며 시간 복잡도는 O(n^2)이다.

**Selection sort**

- 먼저, 첫번째 index부터 시작하여 배열에서 제일 작은 값을 탐색한다. 그리고 가장 작은 값을 찾고 나면 배열의 첫번째 index에 찾은 값을 넣고 남은 배열 값들을 오른쪽으로 밀어넣는다. 그리고 두번째 index로 update하여 앞의 index를 제외하고, 남은 배열 값 중 제일 작은 값을 찾는다. 마찬가지로 두번째 index에 찾은 값을 넣고 남은 배열 값들을 오른쪽으로 밀어넣는다. 이를 반복하며 정렬을 완료한다. 마찬가지로 이때 시간복잡도는 O(n^2)이다.

**Insertion sort**

- 배열의 앞부터 값들을 하나씩 추가해가며 정렬해 나가는 알고리즘이다. 먼저, 첫번째 index의 배열 값은 앞에 비교할 값들이 없으므로, 바로 두번째 index부터 기준으로 잡고 알고리즘을 실행한다. 현재 기준으로 정한 index에서 앞의 index들의 배열 값들을 앞에서부터 탐색해가며 자신보다 큰 값이 나오면 그 값부터 뒤의 모든 값을 오른쪽으로 밀고 그 자리에 현재 기준으로 정한 index의 값을 넣는다. 이를 반복해가며 정렬을 완료하고, 마찬가지로 시간복잡도는 O(n^2)이다.

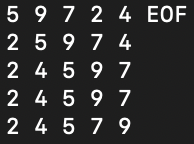
2. 실행 결과 스크린샷

Bubble sort

장치이(가) 표시된 사진

자동 생성된 설명

Selection sort



Insertion sort

측정기, 표시중, 사진, 장치이(가) 표시된 사진

자동 생성된 설명

3. 결과에 대한 설명

먼저, 5 9 7 2 4 EOF를 입력받아, EOF를 만나면 sorting을 시작한다.

1) Bubble sort

Sort를 실행하면서 5와 9를 비교하여 9가 더 크므로 변화가 없고, 다음 index로 넘어가 9와 7을 비교한다. 뒤의 index인 7이 더 작으므로 swap하여주고 변화가 있으므로 출력한다. 그 다음은 9와 2를 비교하여 swap하고 출력, 다음은 9와 4를 비교하여 swap하고 출력한다. 배열의 모든 부분을 탐색했으므로 처음 index로 돌아가 과정을 반복한다. 5와 7은 변화가 없고, 7과 2는 swap 후 출력, 7과 4도 swap 후 출력, 7과 9는 변화가 없다. 다시 앞으로 돌아와 5와 2를 비교하고 swap하여 출력하고, 5와 4를 비교하고 swap하여 출력, 나머지 뒷 부분은 정렬이 완료되었으로 변화가 없고, 배열의 모든 값들이 정렬되었다.

2) Selection sort

Sort가 실행되면 먼저 배열의 0번 index를 기준으로 잡고 가장 작은 값을 찾는다. 2가 제일 작으므로 이때의 index와 값을 저장하고, 2 앞의 모든 값들(5,9,7)을 오른쪽으로 밀고, 0번 index에 2를 넣는다. 그리고 기준 index를 1번 index로 잡고, 0번 index를 제외한 나머지 값들 중 가장 작은 값을 찾는다. 4이므로 1번 index에 4를 넣고, 기준 index를 update한다. 3번째 line과 4번째 line이 같은 이유는 2번 index를 기준으로 작은 값을 찾을 때, 제일 작은 값이 5인데 이미 2번 index에 있기 때문에 같은 결과가 나온다. 이후 과정을 반복하며 정렬이 완료된다.

3) Insertion sort

Sort가 실행되면 먼저 1번째 index를 기준으로 실행되며 따라서 9가 기준이 된다. 앞의 Index의 배열 값을 비교하였을 때, 5는 9보다 작으므로 변화가 없다. 다음 2번째 index로 기준이 update되고, 7은 5보다 크고 9보다 작으므로 이 두 값의 사이가 되므로 9가 있는 자리를 비우고 9를 포함하여 뒤의 모든 값들을 오른쪽으로 밀어놓는다. 그리고 9가 있던 자리에 7을 넣는다. 다음은 3번째 index가 기준이 되고, 2는 5보다 작으므로 5,7,9 값을 모두 오른쪽으로 밀고 0번째 index자리에 2를 넣는다. 마지막으로 4번째 index가 기준이 되고 4는 2보다 크고 5보다 작으므로 5, 7, 9를 모두 오른쪽으로 밀고 빈자리에 4를 넣는다.

4. 구현사항 / 미구현사항 / 버그 / 개선점 등에 대한 설명

1) 구현사항 : Bubble sort, Selection sort, Insertion sort 구현,

띄어쓰기로 구분하고 character로 숫자 입력 가능,

1~99999까지 character로 출력 가능,

99999가 넘어가는 수를 input으로 넣을거나 input 개수가 8개를 넘어가면 종료

숫자, EOF를 제외하고 다른 알파벳 등이 들어오면 종료

2) 미구현사항 :

3) 버그

: 한번 입력을 받고 sorting 후에 sictools를 완전히 종료하고 다시 실행하여 입력 받아야 sort가 정상 실행됨

4) 개선점 : 1000 같은 경우 1 뒤에 0을 character로 구현하기 어려웠는데 자리별 처리 함수를 단계적으로 처리하여 해결하였다.( ex) 1020 의 경우, 먼저 1000자리를 출력하는 함수를 수행하고 이후 원래 함수가 불렸던 자리로 가는 것이 아니라 100의 자리를 출력하는 함수로 바로 이동시켜 출력하게 하였고, 이후 10의 자리, 1의 자리도 마찬가지로 처리하였다. )

5. 미구현사항 / 버그의 경우 타당한 이유 제시  
 1) 미구현사항

2) 버그

6. 고찰

코딩을 하면서, high-level language만 사용하다 보니 low-level에 대해서는 무시하고 코딩을 했던 것 같다. 처음에 sorting의 경우, 쉽게 짤 수 있는 코드라 생각하여 금방 할 수 있을 줄 알았는데, 직접 어셈블리어로 sictools를 이용하여 경험해보니 한 단계만 Low한 language임에도 정말 구상하기가 힘들었다. Register에 값을 load하고 store하면서 관리를 해주어야 하는데 이를 무시하여 이상한 값이 들어가는 경우도 있었고, 컴퓨터 구조를 공부했어서 그래도 쉽게 감을 잡을 수 있었다. 그리고 나니 지금까지 C 언어나 C++ 언어를 사용해 코딩을 해왔던 것보다 훨씬 직관적이어서 생각을 코드로 옮기기가 더 수월했던 것 같고, 정말 유익한 과제였다. 앞으로 코드를 작성할 때 잘못된 부분, 잘못될 수 있는 부분에 대해서 더 신중하게 생각하게 될 것 같다.