

알고리즘설계 HW #1

Sorting

보고서 작성 서약서

1. 나는 타학생의 보고서를 베끼거나 여러 보고서의 내용을 짜집기하지 않겠습니다.

2. 나는 보고서의 주요 내용을 인터넷사이트 등을 통해 얻지 않겠습니다.

3. 나는 보고서의 내용을 조작하지 않겠습니다.

4. 나는 보고서 작성에 참고한 문헌의 출처를 밝히겠습니다.

5. 나는 나의 보고서를 제출 전에 타학생에게 보여주지 않겠습니다.

나는 보고서 작성시 윤리에 어긋난 행동을 하지 않고 정보통신공학인으로서 나의 명예를 지킬 것을 맹세합니다.

2020년 5월 1일

학부 정보통신공학과

학년 4

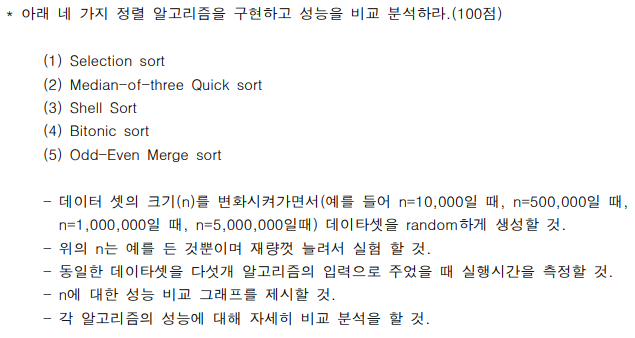
성명 이정우

학번 12171833



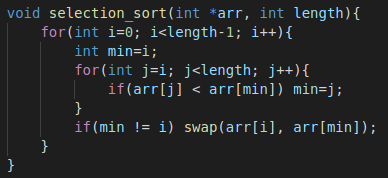
최소한 아래의 내용을 포함할 것.

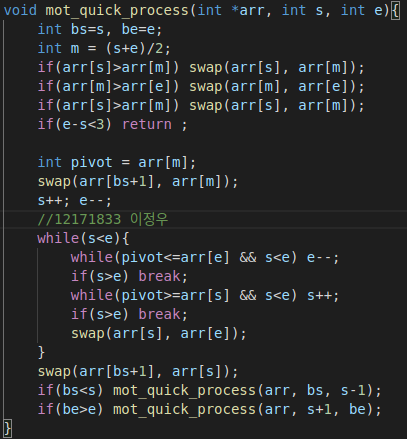
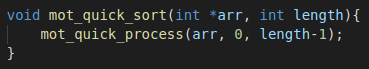
1. 개요



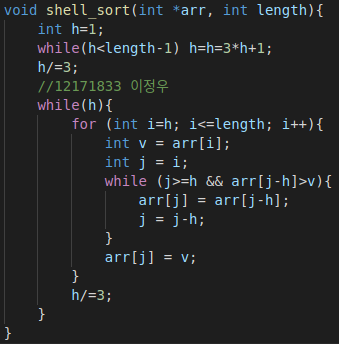
1. 상세 설계내용

(1) Selection sort

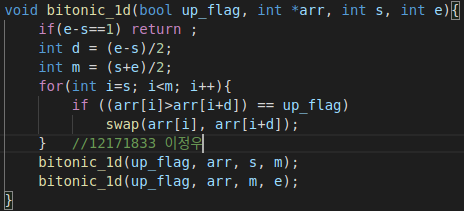
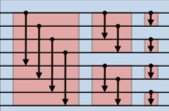
(2) Median-of-three Quick sort

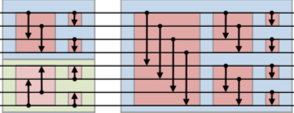
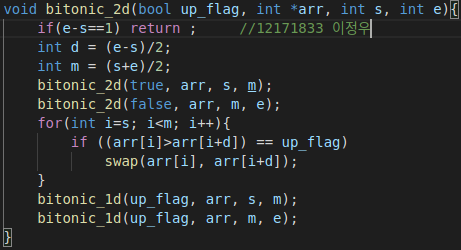
맨 처음 array의 처음, 중간, 끝 값을 비교하여 중간 값을 pivot으로 사용한다.  
처음과 끝 값 사이의 차이가 3보다 작을 경우 pivot을 찾는 과정에서 정렬의 완료되기 때문에 return 해준다.  
중간 값의 위치를 처음+1의 위치로 옮기고 2개의 값은 이미 비교를 하였기 때문에 처음+1, 끝-1부터 pivot과 비교를 하여 값을 swap해준다.  
마지막에 pivot값을 배열의 pivot보다 작은 값들의 맨 끝으로 옮기고 pivot의 좌, 우 배열을 quick sort하는 것을 반복한다.

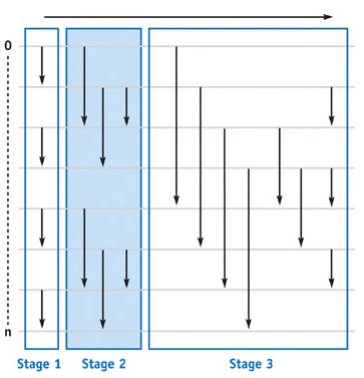
(3) Shell sort

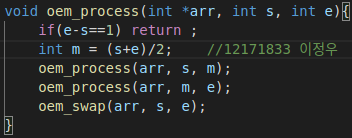
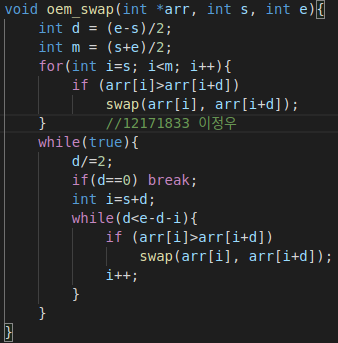
맨 처음 h의 크기를 찾은 후 h의 크기가 1의 될 때까지 3으로 나누어 주며  
h의 간격대로 insertion sort를 수행한다.

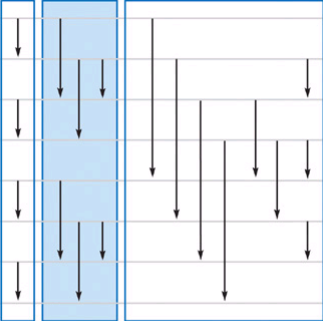
(4) Bitonic sort

배열의 s~e-1사이의 값을 같은 방향으로 array의 크기를 반으로 나누며 반복 정렬한다.

처음의 배열을 각기 다른 방향으로 반 씩 나누어 bitonic\_2d를 불러준다.   
다음으로 자기 자신을 up\_flag방향으로 정렬하고, 그 bitonic\_1d를 불러주어 반으로 나누어가며 up\_flag방향으로 계속 정렬하도록 한다.

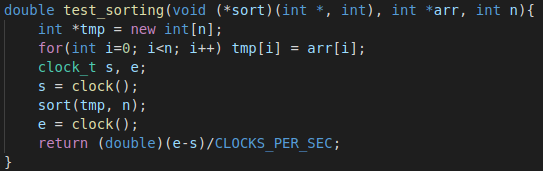
(5) Odd-Even Merge sort

처음에 (e-s)/2의 간격 만큼으로 s부터 e까지 정렬을 한 후  
d를 2로 나눠가면서 s+d부터 s-d까지 d간격으로 정렬을 d가 0보다 큰 동안 반복한다.





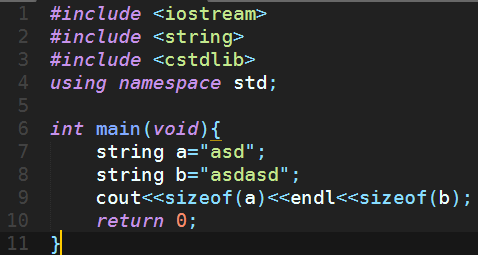
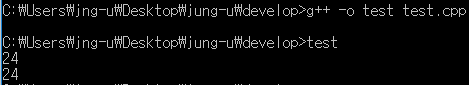
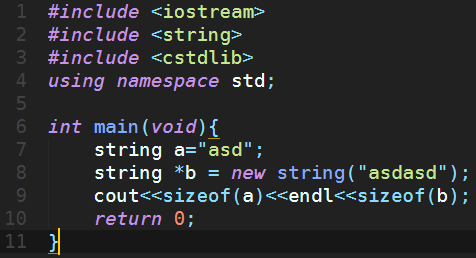
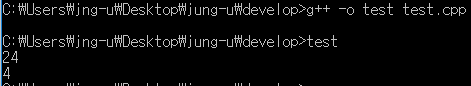
1. 실행 화면



(1), (2), (3)번 sorting 비교

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sort / Data number | **1,000** | **10,000** | **100,000** | **1,000,000** |
| **Selection sort** | 0.00366 (s) | 0.163801 (s) | 13.2164 (s) |  |
| **Median-of-three Quick sort** | 0.000189 (s) | 0.001344 (s) | 0.018529 (s) |  |
| **Shell sort** | 0.000239 (s) | 0.001926 (s) | 0.037895 (s) |  |

1. 분석 및 결론

프로그램을 코딩하면서 처음에 설계했던 UML대로만으로는 개발되지 않았다.   
UML을 짰을 당시에는 다형성을 고려하지 못해 OpenedSNS와 ClosedSNS에 SNS에 없는 변수를 만들어 사용하도록 설계 했었는데 실제로 프로그램을 실행해보니 그렇게 하면 다형성을 활용하지 못함을 알고 코드를 수정하였다.  
또한, 처음에 header.h라는 파일에 모든 파일의 헤더를 정의 해놓고, 모든 파일에 header.h.만을 include하면 된다고 생각하였었는데, 그렇게 하니 헤더파일이 꼬여서 컴파일 하였을 때 에러가 발생하였으나, 헤더파일을 따로 include해주고 난 뒤에 해결되었다.  
string클래스를 제대로 활용해 본 것은 이번이 처음인데, 수업시간에 string의 경우 문자열이 heap메모리 영역에 저장된다고 한 것이 기억나 한번 실제로 확인해보았다.   
  
  
이것을 통해 stirng클래스에서 문자열이 heap메모리에 저장된다는 것을 알 수 있었다.  
  
  
그러나 string클래스 자체가 24bytes를 차지하기 때문에 프로그램을 짜면서 string객체는 모두 포인터를 이용하여 선언하였다.  
클래스를 상속할 때 protected를 이용하지 않고 private과 scope operator를 사용하여 코딩하였다.  
크기가 큰 객체의 경우는 될 수 있다면 포인터를 이용하여 stack memory사용을 줄였다.  
코딩을 하면서 파일의 개수가 많아지니 부모클래스에서의 사소한 것이 바뀌면 자식클래스에까지 영향을 끼쳐 수정하는데 굉장히 번거로웠다. 클래스를 처음 설계할 때 잘 설계하는 것이 중요함을 느꼈다.