X86 배열 할당 및 정렬 알고리즘과 이진 탐색 코드

32214913\_홍석일

1. 문제 기술
   1. 1. C언어로 정렬 프로그램을 작성한 후 어셈블리 코드를 분석
   2. 2. 이진 탐색 알고리즘을 어셈블리 프로그램으로 구현
2. 문제
3. 텍스트, 스크린샷, 폰트이(가) 표시된 사진

   자동 생성된 설명C언어로 정렬 프로그램을 작성한 후 어셈블리 코드 분석.
   1. mkarr 함수를 이용하여 인자로 받은 배열의 길이인 leng크기의 배열을 동적할당한 후, 정수를 입력받도록 했습니다.
   2. 이후 bubble sort 함수를 이용해 입력받은 정수들을 정렬했습니다.
   3. 반복문을 통해 배열을 선회하며 정렬된 배열의 원소들을 출력했습니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

Bubble sort의 코드는 다음과 같습니다.

1. 정렬 프로그램의 어셈블리 코드 분석.
   1. 텍스트, 폰트, 스크린샷, 번호이(가) 표시된 사진

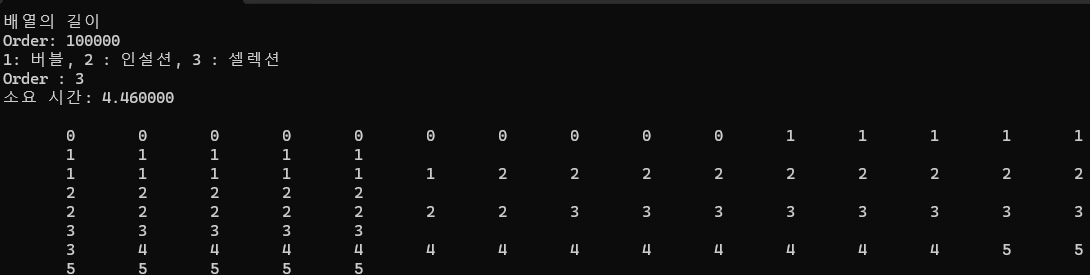
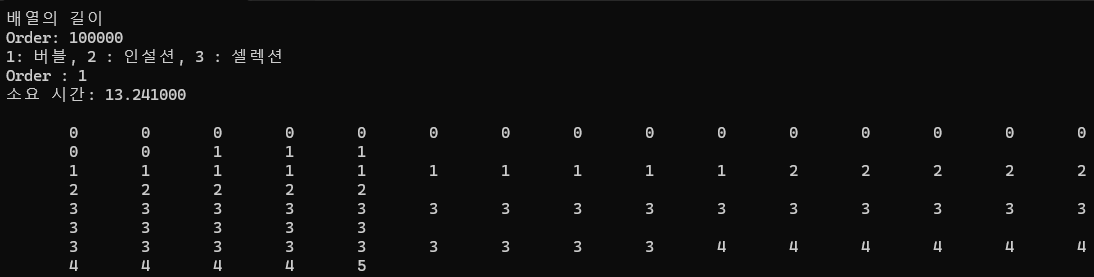
      자동 생성된 설명함수 시작부 분석
      1. ebp를 스택에 push한 뒤, ebp레지스터에 esp레지스터의 내용물을 할당합니다.
      2. esp레지스터에 16을 뺌으로 스택 프레임을 형성하는 작업을 끝냅니다.
      3. 이후 ebp – 4 메모리에 상수 0을 할당합니다. 첫 for문의 int i = 0부분에 해당하는 코드로 예상됩니다.
      4. 함수 반복부 분석
2. 버블(Bubble)정렬의 설계  
   뒤로 가며 인접한 두 원소의 크기를 비교해 Swap함수를 이용해 순서를 바꿔가며 뒤로 정렬한다. 마지막 값까지 정렬한 뒤, 그 다음 정렬시 마지막으로 정렬한 대상 직전까지 비교해 비교대상을 1씩 줄여가며 정렬한다.  
   스크린샷, 텍스트, 폰트이(가) 표시된 사진

   자동 생성된 설명
3. 인서션(Insertion)정렬의 설계  
   배열의 두 번째 값부터 시작하여 선택한 원소에 대해 이전 순번의 원소들과 비교해 크기를 비교하여 정렬 방향에 맞게 이전 순번의 원소들을 뒤로 미룬 뒤, 선택한 원소의 알맞은 정렬 위치에 집어넣는다.  
   텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

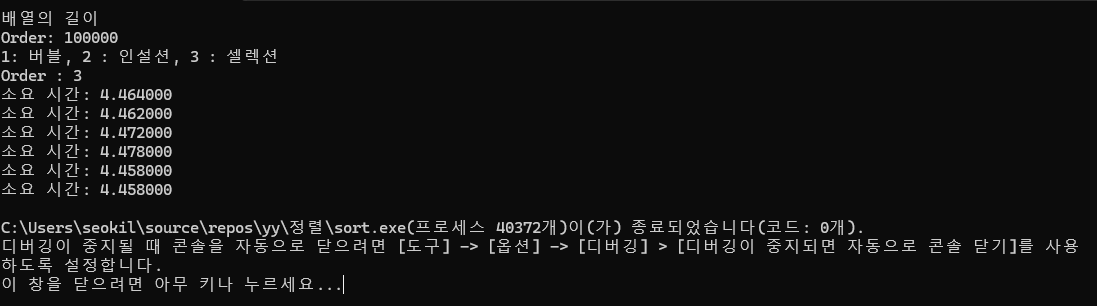
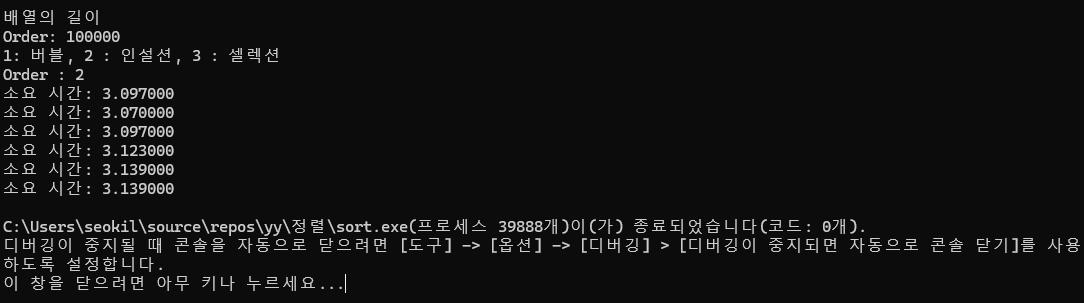
   자동 생성된 설명
4. 셀렉션(Selection)정렬의 설계  
   첫 번째 원소부터 시작해서 뒤의 원소들을 지나가 첫 원소보다 작은 원소들 중 가장 작은 값의 원소와 첫 번째 원소를 swap연산한 뒤, 그 다음 원소에 대해 다시 해당 원소보다 작은 값을 색적하는 연산을 배열 전반에 걸쳐 시행한다.  
   텍스트, 스크린샷, 폰트이(가) 표시된 사진

   자동 생성된 설명
5. 실행시간 측정  
   각 정렬연산 전에 time\_t형식의 start변수를 이용해 연산 시작 직전 시간을 기록한다.  
   이후 연산이 종료된 후 end 변수로 시간을 기록한 후 end-start연산을 통해 소요된 시간을 출력한다.  
   텍스트, 스크린샷, 폰트이(가) 표시된 사진

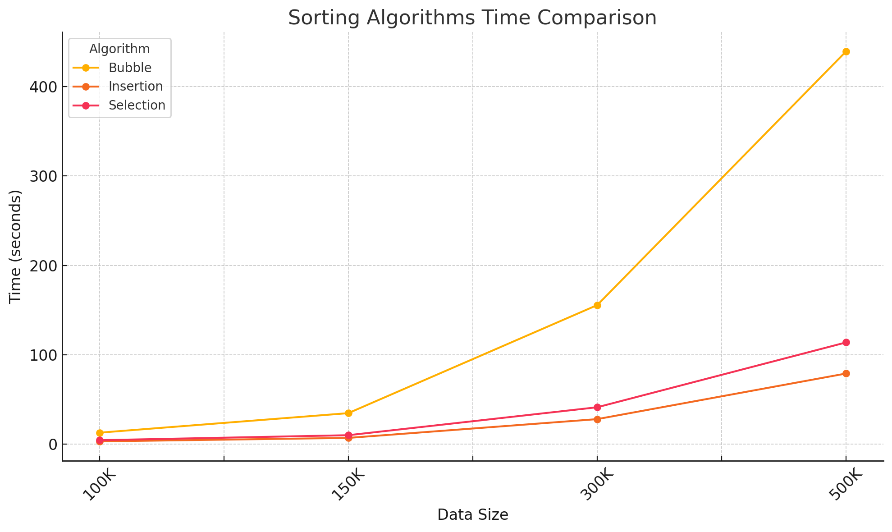
   자동 생성된 설명
6. 실행

텍스트, 스크린샷, 블랙, 디자인이(가) 표시된 사진

자동 생성된 설명  
배열크기 10K에 대한 연산 실행 결과 확인  
정렬 후, 배열의 값을 출력하도록 하여 실제로 정렬이 잘 이루어졌는지 확인 완료  
  
텍스트, 스크린샷, 블랙, 폰트이(가) 표시된 사진

자동 생성된 설명  
배열에 대한 정렬연산을 5회씩 시행해 실행시간에 대한 편차가 있는지 확인 완료

1. 연산시간 비교

  
배열크기에 따른 연산시간 차이  
학습물에서 배우기론 삽입정렬의 경우 최선의 경우 n의 시간복잡도를 가지며, 세 함수 모두 평균적으로 n^2의 복잡도를 갖는다고 배웠다.  
하지만, 실제 정렬 결과의 경우 항상 배열의 뒷부분을 정해진 위치까지 쭉 탐색하는 버블정렬이 가장 높은 연산시간을 보인다.  
삽입정렬과 선택정렬의 경우 비교적 크지 않은 연산시간 차이를 가진다. 결과적으로 모든 정렬 알고리즘들의 시간복잡도는 배열의 크기가 커질수록 더 가파르게 증가하는 n^2의 시간복잡도에 일치하는 모습을 보였지만, 알고리즘 자체의 효율성 때문인지 버블정렬의 연산시간이 두드러지게 차이가 났다.

1. 토의 및 개선점
   1. 행렬의 계산때와는 달리 1차원 배열을 사용하는 정렬 알고리즘의 특성상 메모리가 부족해 계산 자체가 불가능한 경우는 없어서 비교적 수월했다.
   2. 학습할때 배운 시간복잡도가 n^2으로 세 알고리즘 모두 보통의 경우 같은 시간복잡도를 갖는다고 나와있지만, 실제 테스트 결과 버블정렬이 동일한 배열 크기에서 유의미하게 더 많은 시간을 소모한다는 사실을 알았다.
   3. 모듈라 프로그래밍을 하기에 애매한 코드 길이의 프로그램이다.  
      배열의 할당 및 데이터 삽입, 출력 등 여러 부가기능을 위해 module파일을 만들었지만, 각 기능을 함수로 구현할시 사용되는 코드의 길이가 길지 않아 모듈라 프로그래밍의 장점을 발휘하지 못 했다고 생각한다.
2. Makefile  
   텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

   자동 생성된 설명

텍스트, 스크린샷, 라인, 폰트이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 라인, 폰트이(가) 표시된 사진

자동 생성된 설명



텍스트, 스크린샷, 번호이(가) 표시된 사진

자동 생성된 설명