CSE3081 (2반): 알고리즘 설계와 분석
<숙제 2 >

담당 교수: 임 인 성

2018년 10월 16일

마감: 11월 5일 월요일 오후 8시 정각
제출물, 제출 방법, LATE 처리 방법 등: 조교가 과목 게시판에 공고할 예정임.

목표: (1) 수업 시간에 배운 몇 가지 정렬 알고리즘의 구현을 통하여 정렬 기법에 대한 이해도를 높이도록 한다. (2) Divide-and-conquer 기법에 기반을 두고 있는 quick sort 알고리즘으로부터 출발하여 수행 성능을 제고할 수 있는 최적화 기법을 하나씩 적용해가면서 그 효과를 측정하여봄으로써, 최적의 소프트웨어 구현 기술에 대한 이해를 높이도록 한다.

1. 주어진 `unsigned int` 타입의 배열 데이터를 비감소 순서 (nondecreasing order)로 정렬해주는 함수 구현에 관한 숙제이다. 우선 다음과 같은 프로토타입을 가지는 정렬 함수를 가급적 효율적으로 구현하라.

   (a) **Insertion sort:** 수업 시간에 설명한 전형적인 insertion sort 방법을 사용함.

   ```
   int Insertion_Sort(int *data, int left, int right);
   ```

   정렬할 정수 데이터는 포인터 변수 `data`가 가리키는 곳부터 시작하여 순서대로 저장되어 있으며, 인덱스가 `left`부터 `right`까지인 영역의 데이터, 즉 `data[left]`부터 `data[right]`까지 `right-left+1`개의 정수 데이터를 정렬해주어야 한다. 또한 이 함수는 성공적으로 정렬이 끝났을 때에는 0값을, 어떠한 이유로건 정렬을 끝마치지 못하였을 경우에는 적절한 에러 코드를 리턴하도록 설계하라. 이는 아래의 모든 정렬 함수에 공통적인 사항임.

   (b) **Selection sort:** 수업 시간에 설명한 전형적인 selection sort 방법을 사용함.

   ```
   int Selection_Sort(int *data, int left, int right);
   ```

   (c) **Quick sort P (Pivot):** 배열 데이터의 원소의 개수가 4개 이상이면 무조건 재귀적으로 함수를 호출하고, 3개 이하이면 직접 정렬하는 가장 초보적인 quick sort 방법을 사용함. Pivot 원소는 시간 복잡도가 $O(1)$인 자신이 정한 방법을 사용하고, 이후 아래의 quick sort 변형 방법에 대해서도 동일한 pivoting 방법을 사용할 것.

   ```
   int Quick_Sort_P(int *data, int left, int right);
   ```

   (d) **Quick sort PSS (Pivot + Selection Sort):** 앞의 `Quick_Sort_S()` 함수의 변형으로서, 배열 데이터 원소의 개수가 M개 이상일 경우에만 재귀적으로 함수를 호출하고, 그 이하일 경우에는 해당 부분을 selection 방법을 적용하여 정렬함 ($M > 1$인 정수 중 튜닝 작업을 거쳐 최적의 M을 결정할 것). (주의: 이 방법은 quick sort을 모두 마친 후 전체적으로 selection sort 방법을 한 번 적용하는 것이 아니라, M개 미만인 부분에 대해서 바로 selection sort 방법을 적용하는 것으로서 그리 바람직한 방법은 아님)

   ```
   int Quick_Sort_PSS(int *data, int left, int right);
   ```

   (e) **Quick sort PIS (Pivot + Insertion Sort):** 앞의 `Quick_Sort_S()` 함수의 변형으로서, 배열 데이터 원소의 개수가 M개 이상일 경우에만 재귀적으로 함수를 호출함. 불완전한 퀵 정렬 이후 전체 데이터에 대하여 insertion sort 방법을 한 번 더 적용하여 전체 정렬을 마침 ($M > 1$인 정수 중 튜닝 작업을 거쳐 최적의 M을 결정할 것).

```
int Quick_Sort_PIS(int *data, int left, int right);
```

(f) **Quick sort PISTRO (Pivot + Insertion Sort + Tail Recursion Optimization):** 수업시간에 다룬 구현 기법, 즉 Pivot Strategy 사용, Insertion Sort와의 결합, 그리고 Tail Recursion Optimization 기법을 모두 적용하여 가급적 효율적으로 quick sort 함수를 구현할 것.

```
int Quick_Sort_PISTRO(int *data, int left, int right);
```

2. 각 정렬 방법의 성능을 평가하기 위하여 원소의 개수 n에 대해 ($n = 1024 \times 2^m$, $m = 0, 1, 2, 3, \cdots$ 중 적절한 간격으로 m을 선택) 자신이 처리할 수 있는 최대 크기까지 다음과 같은 성질을 가지는 데이터를 만든 후 실험을 하라.

   (a) **Entirely random:** 0과 MAX_RAND 사이의 난수를 n개를 생성하여 만든 데이터.

   (b) **Ascending:** 원소 값이 $0, 1, 2, \cdots, n-1$인 데이터 (우연히 이미 정렬되어 있는 데이터).

   (c) **Descending:** 원소 값이 $n-1, n-2, \cdots, 2, 1, 0$인 데이터 (완전히 순서가 반대로 주어진 데이터).

   (d) **Few swaps:** (b)번 데이터에서 출발하여, $int(\sqrt{n})$개의 random number pair (i, j)를 생성하여 i번째 원소와 j번째 원소를 서로 교환함 (이때 $|i-j| < \sqrt{n}$의 조건을 만족하도록 할 것).

3. 이러한 데이터를 각 n에 대하여 각 정렬 방법의 수행 시간을 가급적 정확하게 측정하라 (조교가 시간 측정에 대한 예제 프로그램을 이메일로 공지할 예정임).

4. 다음의 내용을 포함하여 자신이 적용한 구현 방법과 실험 방법, 그리고 분석 결과를 `HW2_S20179999.{hwp,doc,docx,txt}`와 같은 이름의 보고서에 기술하라.[1]

   - 사용한 CPU의 속도 및 메인 메모리의 용량을 기술하라 (프로그램 수행에 충분한 크기의 메모리를 장착한 컴퓨터를 사용할 것).

     ```
     OS: Window 7 Professional
     CPU: Intel(R) Core(TM)2 Duo CPU E6850 @ 3.00GHz 2.99Ghz
     RAM: 4.00GB
     Compiler: Visual Studio 12.0 Release Mode
     ```

   - 본 숙제의 중요한 목적 중의 하나는 정렬 방법의 이론적인 시간 복잡도와 실제 프로그램의 수행 시간과의 연관성을 찾아내는 것이다. 따라서 각 함수에 대하여 그러한 연관성을 가장 잘 나타낼 수 있는 *n* 값들을 신중히 선택할 것. 특히 시간 측정이 가능한 범위에서 매우 큰 *n* 값에 대해서도 실험할 것.

   - 각 정렬 함수에 대한 실험 결과를 적절한 형식의 테이블로 요약하라. 실제 측정한 시간 값과 이론적으로 구한 시간 복잡도의 연관성을 증명하기 위하여 그래프를 그린다던가 또는 수식으로 함수 관계를 보인다던가 하는 등의 **공학적인** 방법을 사용하여 자신의 주장을 **논리적으로** 상세히 기술할 것. 이때 다음과 같은 내용을 포함하여 자신이 발견한 사항에 대하여 기술하라.
     - 수행 시간과 이 방법들의 이론적인 시간 복잡도간에 어떤 연관성을 발견할 수 있는가?
     - **Entirely random** 데이터의 크기를 변화시켜가면서 실험을 하였을 때, **Insertion sort**와 **Selection sort** 간에 두드러진 속도 차이를 느낄 수 있는가?
     - **Entirely random** 데이터의 크기를 변화시켜가면서 실험을 하였을 때, **Quick sort P** 방법 대비 **Insertion sort** 방법의 수행 시간에 대하여 $\frac{n}{\log n}$와 같은 비율이 측정되었는가?
     - 과연 **Quick sort PSS (Pivot + Selection Sort)** 방법이 다른 방법에 비하여 의미가 있는가?
     - 이때 서로 다른 종류의 데이터에 대하여 수행 시간에 차이가 있는가? 있다면 그 이유는 무엇으로 추정되는가?
     - 과연 어떤 부류의 데이터에 대해 insertion sort 방법의 수행 시간이 quick sort 방법에 비해 그리 느리지 않은가?
     - *n* 값이 작을 경우 굳이 quick sort 방법을 사용할 필요없이 insertion sort 방법을 사용해도 크게 문제가 되지 않는다. 과연 이는 어느 정도까지의 *n* 값까지일 지 실험을 통하여 그 값을 밝혀라.

[1] 여기서 `S20179999`는 자신의 학번임.

  - 각 방법을 최대한 효율적으로 구현하기 위하여 어떤 노력을 기울였는가?
  - 기타 이번 숙제를 하면서 적용한 창의적인 방법이나 경험적으로 알게된 중요 사항을 기술하라.

- **중요:**
  - 숙제를 제출할 때, 자신이 구현한 다섯 개의 정렬 함수 코드를 이름이 `S20179999`인 디렉토리 안에 `IS`, `SS`, `QS_P`, `QS_PSS`, `QS_PIS`, 그리고 `QS_PISTRO` 디렉토리를 만들어 각 해당 디렉토리에 저장하라.
  - 각 디렉토리에는 조교가 코드를 수행하여 볼 수 있도록 데이터를 읽어들이고 정렬 함수를 호출하는 `main` 함수를 포함한 모든 코드를 제출하여야 함.
  - 각 main 함수 파일의 제일 첫 줄에는 다음과 같이 자신이 사용할 입력 파일의 이름이 매크로로 정의되어 있어야 한다. 기본적인 입력 디렉토리는 main 함수 파일이 존재하는 디렉토리이어야 한다.

    ```
    #define INPUT_FILE_NAME "input_array_1024.bin"
    ```

  - 본 숙제에서 사용할 배열 데이터는 ASCII 형식이 아니라 binary 형식을 사용하여 다음과 같이 파일에 저장되어있다고 가정한다: 먼저 첫 네 바이트에는 정렬할 배열의 원소 개수가 `unsigned int` 형식으로 저장되어 있고, 이후 연달아 그 개수만큼의 원소가 한 개당 네 바이트씩 `unsigned int` 형식으로 저장되어 있다.
  - 채점 시 배열 데이터는 조교 자신의 것을 사용할 예정이므로 제출할 필요가 없으며, 자신의 코드가 입력 파일 형식을 따르지 않을 경우 프로그램이 수행하지 않는 것으로 간주할 예정임.
  - 조교는 작은 크기의 샘플 데이터를 이메일로 공지할 예정임.
- 이번 숙제에 대한 배점은 (구현 내용 및 완성도 + 보고서 기술 내용 + 속도)로 구성됨.

- **[주의]**

1. 본인의 결과를 `HW1_S20179999.{hwp,doc,docx,txt}`와 같은 이름의 보고서에 기술하여 프로그램 및 데이터와 함께 제출하라.
2. 시간 측정은 수업 시간에 조교가 설명하는 방식을 사용할 것.
3. 조교는 자신의 명령어 파일과 입력 데이터를 사용하여 여러분의 프로그램이 정확한 값을 계산하는지 확인할 예정임.
4. 숙제 제출 기간 동안 조교가 숙제와 관련하여 중요한 공지 사항을 게시판에 올릴 수 있으니 항상 수업 게시판을 확인하기 바람.
5. 제출 화일에서 바이러스 발견 시 **본인 점수 X (-1)**이고, 다른 사람의 숙제를 복사할 경우 **관련된 사람 모두에 대하여 만점 X (-10)**임.