CSE3081/AIE3051 (1반): 알고리즘 설계와 분석
[HW 3] Quick Sort 방법의 효율적인 구현

담당 교수: 임 인 성

2025년 10월 27일

마감: 11월 10일 월요일 오후 8시 정각
제출물, 제출 방법, LATE 처리 방법 등: 조교가 과목 게시판에 공고할 예정임.

목표: Divide-and-conquer 기법에 기반한 교과서적인 quick sort 방법과 그의 변형 방법을 구현하여 실제 수행 시간을 측정해봄으로서, 정렬 이론과 실제에 대한 이해를 높이도록 한다. 특히, Standard C/C++ Library가 제공하는 quick sort 함수인 `qsort()`를 비교 대상으로 하여 성능 향상을 위한 최적화 기법을 적용하여 그 효과를 측정하여봄으로서 최적의 소프트웨어 구현 기술에 대한 이해를 높이도록 한다.

[구현 내용] 다음과 같이 원소당 16 바이트의 크기를 가지는 배열 데이터를 고려하자.

```
typedef struct {
    unsigned int score; char other_data[12];
} RECORD;
```

본 숙제에서는 key field에 해당하는 `score` 값을 기준으로 입력 배열의 원소들을 비감소 순서 (non-decreasing order)로 정렬을 하려한다.

1. 먼저 여러분의 프로그램은 다음과 같은 내용의 이름이 `HW3_commands.txt`인 ASCII 파일에서 데이터 정렬에 필요한 정보를 읽어들인다.

   ```
   1
   n
   unsorted_array.bin
   sorted_array.bin
   ```

   첫 줄의 정수 값은 실험에 사용할 함수 번호 (아래에서 설명), 그리고 두 번째 줄의 정수 값은 입력 배열의 원소 개수를 나타낸다. 다음 세 번째 줄의 파일 이름은 입력 데이터의 이름이며, 마지막 줄에는 데이터 정렬 후 저장할 파일 이름이다. 이 두 파일은 모두 Binary Format으로 저장이 되며 (따라서 두 파일의 크기는 각각 `sizeof(RECORD)*n`임), 조교는 이 명령 파일과 입출력 파일이 어떤 디렉토리에 존재해야 하는지를 분명히 할 예정임.

2. 첫 번째로, Standard C/C++ Library에서 제공하는 `qsort()` 함수를 호출하여 정렬을 해주는 함수를 작성하라. 참고로 이 함수는 다음과 같이 `qsort()` 라이브러리 함수와 동일한 프로토타입을 가지며, 각 인자가 의미하는 바는 수업 시간에 설명하였다. 이 함수의 번호는 1번이다.

   ```
   void my_qsort(void *, size_t, size_t, _Cmpfun *);
   ```

3. 다음 아래와 같은 프로토타입을 가지는 자신만의 정렬 함수 각각을 가급적 효율적으로 구현하라.

   (a) 구현을 해야할 네 가지 함수는 다음과 같다.

      i. 이름이 `my_qsort_orig()` 인 정렬 방법을 구현하라. 이 방법은 `qsort()` 함수와 동일한 프로토타입을 가지며,

```
void my_qsort_orig(void *, size_t, size_t, _Cmpfun *);
```

교과서적인 quick-sort 방법으로서 (i) 분할된 두 부분 각각에 대해 재귀 함수를 호출하며, (ii) 원소의 개수가 1개가 될 때까지 반복적으로 분할을 해야 한다. 또한 pivot element는 가장 왼쪽, 즉 정렬하려는 배열의 첫 원소의 key 값을 pivot element로 사용해야 한다. 이 함수의 번호는 21번이다.

ii. 이름이 `my_qsort_median_insert()`인 정렬 방법을 구현하라. 이 방법은 `qsort()` 함수와 동일한 프로토타입을 가지며,

```
void my_qsort_median_insert(void *, size_t, size_t, _Cmpfun
*);
```

적절한 개수의 원소만을 사용하는 median 선택 방식의 pivot strategy와 insertion sort 기법을 통한 최적화 기법을 적용해야 한다. insertion sort 기법을 적용할 때 적절한 임계값을 사용하도록 하라. 이 함수의 번호는 22번이다.

iii. 이름이 `my_qsort_median_insert_iter()`인 정렬 방법을 구현하라. 이 방법은 `qsort()` 함수와 동일한 프로토타입을 가지며,

```
void my_qsort_median_insert_iter(void *, size_t, size_t,
_Cmpfun *);
```

기본적으로 `my_qsort_median_insert()` 함수의 구현을 사용하나, 두 개의 분할된 부분 각각에 대하여 재귀적으로 함수를 호출하는 것이 아니라, 길이가 작은 부분에 대해서만 재귀적으로 함수를 호출하고 큰 부분에 대해서는 반복 (iteration)을 통하여 처리해주는 방식을 취해야 한다. 이 함수의 번호는 23번이다.

iv. (마지막으로 속도 측정을 통한 추가 점수 부여에 사용할) 이름이 `my_qsort_final()`인 정렬 방법을 구현하라. 이 방법은 `qsort()` 함수와 동일한 프로토타입을 가지며,

```
void my_qsort_final(void *, size_t, size_t, _Cmpfun *);
```

이 함수는 위에서 구현한 것과 동일한 것이어도 무방하고, 또는 자신만의 최적화 기법을 추가적으로 적용한 것이어도 된다 (만약 후자라면 반드시 보고서에 그러한 사실을 밝혀야 인정을 받음). 위의 세 함수는 배열의 원소를 옮길 때, 매번 전체 16 바이트를 이동 시켜야 하나, 이 네 번째 함수에서는 인덱스 배열을 내부적으로 할당하여 정렬 시에는 인덱스만 서로 바꾸고, 최종 정렬 후 원소들을 그 정보를 사용하여 출력 파일에 정렬 순서대로 저장을 해도 무방함. 단 조교가 지정한 크기의 배열을 처리할 때 인덱스 배열을 할당 받을 수 있어야 한다. 이 함수의 번호는 24번이다.

(b) 이 네 개의 함수들도 `qsort()` 함수 경우와 동일한 명령 파일 `HW3_commands.txt`에서 필요한 정보를 읽어들인 후 정렬을 수행토록 하라. 단 이때 함수 번호가 1이 아니라 21, 22, 23, 또는 24 중 하나를 사용하여 해당 함수를 통하여 정렬토록 한다.

[평가 항목] 평가는 3단계로 진행한다.

1. 먼저 조교는 자신이 생성한 샘플 데이터를 사용하여 번호 1, 21, 22, 그리고 23에 해당하는 함수 각각에 대하여 (i) 요구 사항대로 구현이 되었는지와 (ii) 올바르게 정렬이 되었는지를 평가한다. **[80점 만점]**

   - 채점은 최대 8,388,608 ($= 8 \times 1024 \times 1024$)개의 원소를 가지는 배열을 사용한다.
   - 입력 데이터는 본 과목에서 제공하는 코드의 Fisher-Yates shuffle algorithm을 사용하여 각자 생성한다.

2. (위의 네 개의 방법에 대해 모두 정렬이 제대로 이뤄질 경우에만) 보고서 내용을 평가한다. **[20점 만점]**

   (a) 최소한 다섯 개 이상의 서로 다른 크기를 가지는 입력 데이터를 생성하라. 이때 충분히 넓은 범위의 원소 개수가 포함되도록 하라. 이후 각 데이터에 대하여 각 정렬 방법의 수행 시간을 가급적 정확하게 측정하라.

   (b) 실험을 통하여 자신이 발견한 사항을 정량적인 실험 결과와 함께 간결히 요약하라. 이때 다음과 같은 사항을 포함해야 한다.

- 다음의 예처럼 실험에 사용한 CPU의 속도 및 메인 메모리의 용량 등의 실험 환경을 기술하라 (프로그램 수행에 충분한 크기의 메모리를 장착한 컴퓨터를 사용할 것).

```
OS: Window 11 Pro
CPU: AMD Ryzen 9 5900HS with Radeon Graphics (3.30GHz)
RAM: 24.00GB
Compiler: MS Visual Studio Community 2022 (64-bit) 버전 17.14.0
```

- 다음은 보고서에 기술할 수 있는 내용의 예이다: 자신이 quick sort 방법에 적용한 최적화 기법이 얼마나 효과가 있었는가? 자신이 사용한 pivot 원소 설정 방법이 효과가 있었는가? Insertion sort 방법과 결합한 것이 효과가 있었는가? 재귀적인 방법과 반복적인 방법 간에 눈에 띄는 속도 차이가 발견되었는가? 특히 반복적인 방법을 통하여 원소의 개수가 작은 부분에 대해서만 재귀 함수를 호출함으로서 시스템 스택의 사용을 감소 시킨 것이 효과가 있었는가? 기타 등등.

3. 24번 함수를 사용하여 수행 시간을 측정하여 추가적인 점수를 부여할 수 있다. **[최대 20점]**

   - 조교는 자신이 지정하여 배포한 샘플 데이터에 대한 정렬 시간을 측정하여, 상위 10% 이내 (정확한 숫자는 상황에 따라 추후 결정)의 눈에 띄게 속도가 빠른의 학생에게 추가 점수를 부여한다.
   - 이 함수 구현 시, 2학년 2학기 컴퓨터공학 전공자가 생각할 수 있는 최적화 기법의 적용을 권장한다 (인터넷에서 "C/C++ Code Optimization" 주제로 검색을 해볼 것). 이러한 노력을 인정 받기 위해서는 보고서에 자신이 적용한 최적화 노력에 대하여 명확히 기술할 것.
   - 추가적인 점부를 받는 학생의 코드에 대해서는 특별히 자세한 검토와 최적화 적용 방식에 대한 설명 요청 있을 예정임.

[주의]

1. 제출물은 다음과 같다.

   (a) 프로그램 원시 코드 (자신이 사용한 데이터는 제출하지 말것!)

   - 자신이 구현한 네 개의 함수에 대한 원시 코드를 파일 `my_quick_sorts.cpp`에 저장하라.
   - 헤더 파일 `my_quick_sorts.h`에 자신이 구현한 네 개의 함수의 프로토타입을 선언하라. 특히 그러한 함수 선언 직전에 다음과 같은 타입 선언 문장을 삽입하라.

```
typedef int _Cmpfun(const void *, const void *);
```

   - 위의 두 파일에 기반을 두어 명령어 파일을 읽어들어 원하는 작업을 수행해주는 메인 함수를 이름이 `HW3_S20249999.cpp`인 파일에 저장하라. 메인 프로그램은 1, 21, 22, 23, 그리고 24번 방법을 모두 처리할 수 있어야 한다. 당연히 `_Cmpfun()` 함수는 이 파일에서 구현이 되어야 하며, `qsort()` 사용에 필요한 헤더 파일도 이 파일에서 포함되어야 한다. 조교는 자신만의 메인 함수를 작성하여 여러분이 작성한 파일과 함께 평가를 함.

   (b) 보고서

   - 본인의 결과를 `HW3_S20249999.{hwp,docx,pptx,txt}`와 같은 이름의 보고서에 위에서 기술한 내용을 포함하여 간결하게 작성하라. 특히 부분 점수라도 받기 위해서는 자신이 구현한 항목과 구현하지 못한 항목에 대하여 정확히 구별하여 기술하라.

2. 숙제 제출 기간 동안 조교가 숙제와 관련하여 중요한 공지 사항을 게시판에 올릴 수 있으니 항상 수업 게시판을 확인하기 바람.

3. 제출 화일에서 바이러스 발견 시 **본인 점수** x **(-1)**이고, 다른 사람의 숙제를 복사할 경우 **관련된 사람 모두에 대하여 만점** x **(-10)**임.