자료구조응용

21. quick sort, iterative merge sort (10점)

2025.11.24

1. 다음 입력 리스트에 대해 퀵정렬(quick sort)를 수행하고자 한다.
   입력 리스트 (12, 2, 16, 30, 8, 28, 4, 10, 20, 6, 18)

(1) Figure 7.1과 같은 퀵정렬 과정을 직접 작성해 보이고 이를 통해 quickSort 함수가 몇 번 호출되는지 계산해 보라. (2점)

R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9	R_{10}	*left*	*right*
[26	5	37	1	61	11	59	15	48	19]	1	10
[11	5	19	1	15]	26	[59	61	48	37]	1	5
[1	5]	11	[19	15]	26	[59	61	48	37	1	2
1	5	11	[19	15]	26	[59	61	48	37]	4	5
1	5	11	15	19	26	[59	61	48	37]	7	10
1	5	11	15	19	26	[48	37]	59	[61]	7	8
1	5	11	15	19	26	37	48	59	[61]	10	10
1	5	11	15	19	26	37	48	59	61		

Figure 7.1: Quick sort example

※ 연습장에 적은 후 사진을 찍어도 되며 그 결과를 보고서에 넣을 것

(2) 입력리스트의 데이터를 파일로 입력받아 퀵정렬 수행결과 및 quickSort 함수호출 횟수를 출력하는 프로그램을 작성하여 (1)의 결과와 비교해 보시오. 단, 각 레코드는 하나의 int형 key 필드로 구성되어 있다. (3점)

<실행순서>

① 입력파일(input.txt)로부터 데이터를 읽어 들여 구조체 배열 a에 저장한다.

```
11
12 2 16 30 8 28 4 10 20 6 18
```

※ 첫 줄은 레코드의 정렬할 키의 개수

② 각 레코드의 key에 대해 퀵정렬을 실행한다.

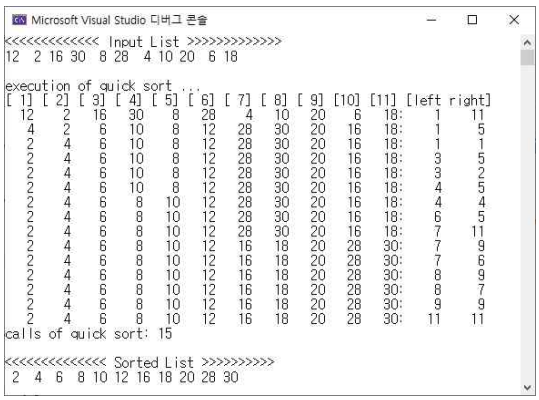
③ **정렬된 key값** 및 **quickSort 함수호출 회수**를 화면에 출력하라.

④ 정렬결과를 파일(output.txt)에 저장한다.

```
void quickSort(element a[], int left, int right)
{/* sort a[left:right] into nondecreasing order
    on the key field; a[left].key is arbitrarily
    chosen as the pivot key;  it is assumed that
    a[left].key <= a[right+1].key */
  int pivot,i,j;
  element temp;
  if (left < right) {
    i = left; j = right + 1;
    pivot = a[left].key;
    do {/* search for keys from the left and right
           sublists, swapping out-of-order elements until
           the left and right boundaries cross or meet */
       do i++; while (a[i].key < pivot);
       do j--; while (a[j].key > pivot);
       if (i < j) SWAP(a[i],a[j],temp);
    } while (i < j);
    SWAP(a[left],a[j],temp);
    quickSort(a,left,j-1);
    quickSort(a,j+1,right);
  }
}
```

Program 7.6: Quick sort

<실행예>



2. 다음 입력 리스트에 대해 반복을 통한 합병정렬(iterative merge sort)을 수행하고자 한다.
입력 리스트 (12, 2, 16, 30, 8, 28, 4, 10, 20, 6, 18)

(1) mergeSort(Program 7.9)의 while 문에서 각 mergePass 호출 후의 배열 a와 extra의 상태를 단계적으로 나타내 보라. 초기 입력 데이터는 배열 a[1:n] 에 있다. (2점)

※ 연습장에 적은 후 사진을 찍어도 되며 그 결과를 보고서에 넣을 것

(2) (1)의 결과를 프로그램으로 확인해 보라. (3점)

<실행순서>

① 입력파일(input.txt)로부터 key를 읽어 들여 구조체 배열 a에 저장한다.

※ element 타입은 key 필드만으로 구성된 구조체를 재정의한 것으로 가정한다.

input.txt
11
12 2 16 30 8 28 4 10 20 6 18

※ 첫 줄의 11은 입력키의 개수

② 각 레코드의 key에 대해 반복을 통한 합병정렬을 실행한다. 이때, mergeSort 함수를 수정하여 mergePass 수행마다 **세그먼트 크기(s), 배열 a와 extra 상태**를 화면에 출력하라.

③ 최종 정렬 결과를 화면에 출력한다.

```
void merge(element initList[], element mergedList[],
           int i, int m, int n)
{/*  the sorted lists initList[i:m] and initList[m+1:n] are
     merged to obtain the sorted list mergedList[i:n] */
   int j,k,t;
   j = m+1;        /* index for the second sublist */
   k = i;          /* index for the merged list */

   while (i <= m && j <= n) {
      if (initList[i].key <= initList[j].key)
         mergedList[k++] = initList[i++];
      else
         mergedList[k++] = initList[j++];
   }
   if (i > m)
   /* mergedList[k:n] = initList[j:n] */
      for (t = j; t <= n; t++)
         mergedList[t] = initList[t];
   else
   /* mergedList[k:n] = initList[i:m] */
      for (t = i; t <= m; t++)
         mergedList[k+t-i] = initList[t];
}
```

Program 7.7: Merging two sorted lists

```
void mergePass(element initList[], element mergedList[],
               int n, int s)
{/* perform one pass of the merge sort, merge adjacent
    pairs of sorted segments from initList[] into mergedList[],
    n is the number of elements in the list, s is
    the size of each sorted segment */
  int i,j;
  for (i = 1; i <= n - 2 * s + 1; i += 2 * s)
    merge(initList,mergedList,i,i + s - 1,i + 2 * s - 1);
 if (i + s - 1 < n)
    merge(initList,mergedList,i,i + s - 1,n);
 else
    for (j = i; j <= n; j++)
      mergedList[j] = initList[j];
}
```

Program 7.8: A merge pass

```
void mergeSort(element a[], int n)
{/* sort a[1:n] using the merge sort method */
   int s = 1; /* current segment size */
   element extra[MAX_SIZE];

   while (s < n) {
      mergePass(a, extra, n, s);
      s *= 2;
      mergePass(extra, a, n, s);
      s *= 2;
   }
}
```

Program 7.9: Merge sort

<실행예>

■ 제출 형식

- 솔루션 이름 : DS 21
- 프로젝트 이름 : 1, 2
- 각 소스파일에 주석처리

  "학번 이름"

  "본인은 이 소스파일을 다른 사람의 소스를 복사하지 않고 직접 작성하였습니다."
- 제출 파일

  ① 소스코드와 실행 결과가 보이도록 화면을 캡쳐한 **보고서** 파일(**"학번.pdf"**)

  ※ 한글 [파일 → pdf로 저장하기...] 메뉴 사용

② **C 소스 파일**을 하나의 디렉터리에 모아 압축한 파일 (**"학번.zip"**)
※ "학번.pdf"와 "학번.zip"을 하나로 압축하지 말고 별도 파일로 제출

■ **주의**

- 소스 복사로는 실력향상을 기대할 수 없습니다!!!
- 1차 마감 : 수업일 자정
- 2차 마감 : 수업 익일 자정(만점의 60%, 반올림)
- 문항 별로 1차 2차 나눠서 제출할 수 없으며, 최종 제출 시간에 따라 1차, 2차로 구분함

- 4시 40분에 제출 상황 체크함

- 완료된 과제를 제출하지 않고 일찍 퇴실한 학생은 0점 처리하겠음