자료구조응용
7. Sorting: merge sort, heap sort

1. 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.

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

(2) (1)의 결과를 프로그램으로 확인해 보라.

<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 정렬결과를 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

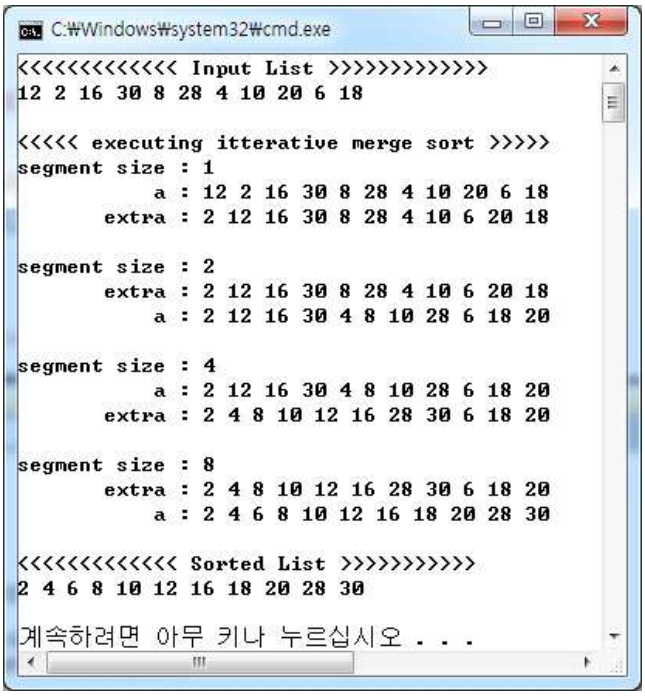
```
void mergePass(element initList[], element mergedList[],
               int n, int s)
{/* perform one pass of the merge sort, merge adjacent
    pairs of sorted segments from initList[] into mergedList[],
    n is the number of elements in the list, s is
    the size of each sorted segment */
  int 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>



2. 다음 입력 리스트에 대해 재귀적인 합병정렬(recursive merge sort)을 수행하고자 한다.
입력 리스트 (26, 5, 77, 1, 61, 11, 59, 15, 48, 19)

(1) recursion tree에 대한 downward pass, upward pass 상태를 각각 그려라.
※ 연습장에 적은 후 사진을 찍어도 되며 그 결과를 보고서에 넣을 것

(2) (1)의 결과를 프로그램으로 확인해 보라.
<실행순서>
① 입력파일(input.txt)로부터 key를 읽어 들여 구조체 배열 a에 저장한다.
※ element 타입은 key 필드만으로 구성된 구조체를 재정의한 것으로 가정한다.

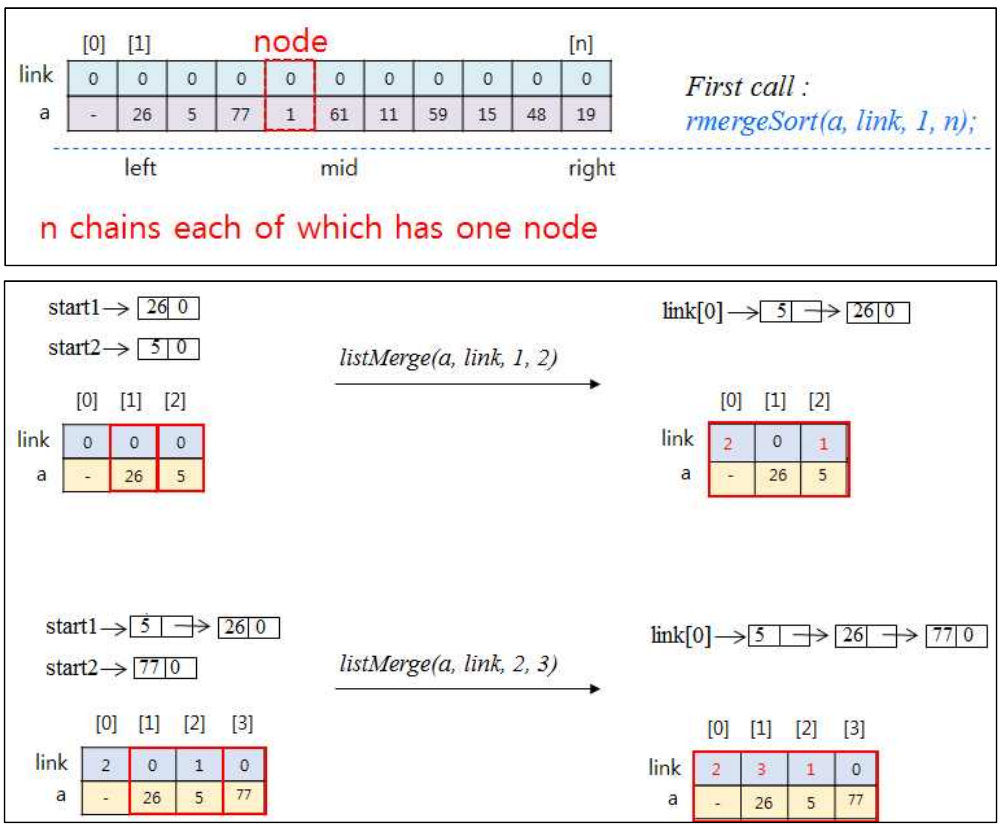
input.txt
10
26 5 77 1 61 11 59 15 48 19

※ 첫 줄의 10은 입력키의 개수

② 각 레코드의 key에 대해 재귀적인 합병정렬을 실행한다.
※ Program 7.11코드 수정 : if(a[last1] <= a[last2]) -> if(a[last1].key <= a[last2].key)

③ 연결리스트를 출력하는 함수를 정의하여 정렬된 결과를 화면에 출력하라.

<자료구조 - Chain >



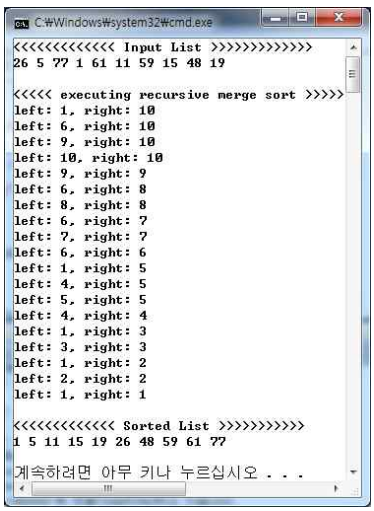
```
int rmergeSort(element a[], int link[], int left, int right)
{/* a[left:right] is to be sorted, link[i] is initially 0
    for all i, returns the index of the first element in the
    sorted chain */
   int mid;
   if (left >= right) return left;
   mid = (left + right) / 2;
   return listMerge(a, link,
                    rmergeSort(a, link, left, mid),
                           /* sort left half */
                    rmergeSort(a, link, mid + 1, right));
                           /* sort right half */
}
```

Program 7.10: Recursive merge sort

```
int listMerge(element a[], int link[], int start1, int start2)
{/* sorted chains beginning at start1 and start2,
    respectively, are merged; link[0] is used as a
    temporary header; returns start of merged chain */
   int last1, last2, lastResult = 0;
   for (last1 = start1, last2 = start2; last1 && last2;)
      if (a[last1] <= a[last2]) {
         link[lastResult] = last1;
         lastResult = last1; last1 = link[last1];
      }
      else {
         link[lastResult] = last2;
         lastResult = last2; last2 = link[last2];
      }

   /* attach remaining records to result chain */
   if (last1 == 0) link[lastResult] = last2;
   else link[lastResult] = last1;
   return link[0];
}
```

Program 7.11: Merging sorted chains

< 실행예 >



3. 다음 입력 리스트에 대해 힙정렬(heap sort)을 수행하고자 한다.

입력 리스트 (12, 2, 16, 30, 8, 28, 4, 10, 20, 6, 18)

(1) heapSort(Figure 7.13) 함수에서 입력 리스트의 트리에 대해 ① 첫 번째 for문 ② 두 번째 for문 수행과정에서 리스트의 상태를 단계적으로 나타내어라. 매번 adjust를 수행한 직후의 상태에 대해서만 트리를 그리면 된다. ②는 정렬된 데이터도 같이 표현하라. (2점)

※ ①은 강의 슬라이드 25, ②는 강의슬라이드 26 참고

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

(2) (1)의 결과를 프로그램으로 확인해 보라.

<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 힙정렬을 실행한다.

※ adjust 함수 수행마다 배열(a)의 인덱스 순서대로 key값을 화면에 출력한다.

※ **출력함수**를 정의하여 사용해야 한다.

③ 정렬결과를 파일(output.txt)에 저장한다.

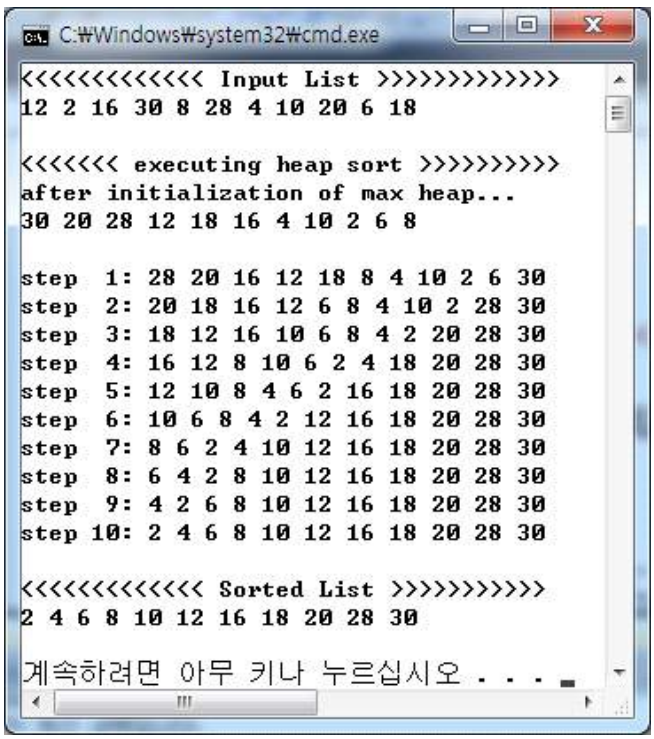
```
void heapSort(element a[], int n)
{/* perform a heap sort on a[1:n] */
   int i,j;
   element temp;

   for (i = n/2; i > 0; i--)
      adjust(a,i,n);
   for (i = n-1; i > 0; i--) {
      SWAP(a[1],a[i+1],temp);
      adjust(a,1,i);
   }
}
```

Program 7.13: Heap sort

```
void adjust(element a[], int root, int n)
{/* adjust the binary tree to establish the heap */
   int child,rootkey;
   element temp;
   temp = a[root];
   rootkey = a[root].key;
   child = 2 * root;                          /* left child */
   while (child <= n) {
      if ((child < n) &&
      (a[child].key < a[child+1].key))
         child++;
      if (rootkey > a[child].key) /* compare root and
                                     max. child */
         break;
      else {
         a[child / 2] = a[child]; /* move to parent */
         child *= 2;
      }
   }
   a[child/2] = temp;
}
```

Program 7.12: Adjusting a max heap

<실행 예>



■ 제출 형식

- 제출 : 12월 6일자 채점서버에 소스 업로드(12월 19일까지)
- 실행화면을 캡쳐하여 한글파일 보고서(**DS 7_학번.hwp**)에 추가 후 과제에 제출 (12월 19일까지)
- 각 소스파일에 주석처리 추가

  “학번 이름”

  “본인은 이 소스파일을 다른 사람의 소스를 복사하지 않고 직접 작성하였습니다.”

■ **주의**

- 소스복사로는 실력향상을 기대할 수 없습니다!!!

- 먼저 제출한 학생은 남은 시간 동안 자료구조 관련 개인학습을 하거나 동료를 도와 줄 것
- 수강생 끼리 서로 물어보고 논의를 해도 됨
- 채점서버에 제출하지 않는 경우 최종 점수의 0점으로 처리함

(사용법을 잘 모르겠다면 개인적으로 튜터나 TA를 찾아와서 물을 것)