자료구조응용

20. Sorting: merge sort (15점)

2020/5/25 (월)

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. (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 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

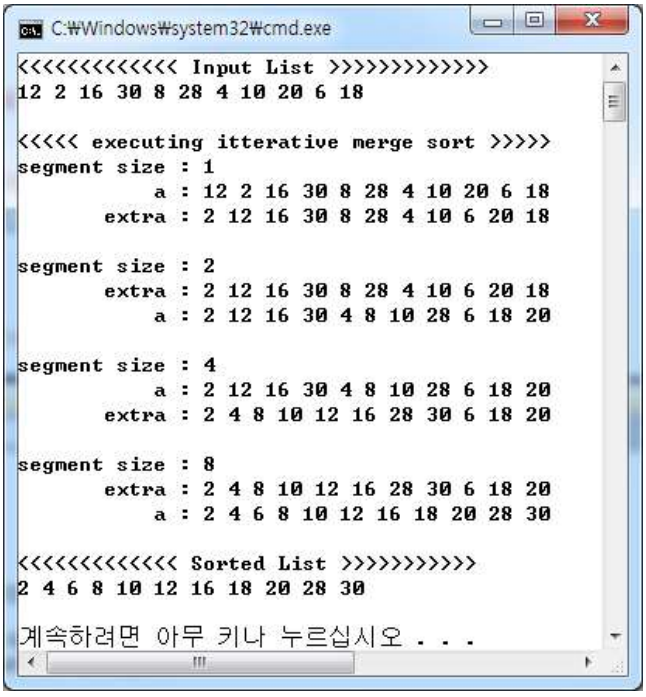
```
void mergePass(element initList[], element mergedList[],
               int n, int s)
{/* perform one pass of the merge sort, merge adjacent
    pairs of sorted segments from initList[] into mergedList[],
    n is the number of elements in the list, s is
    the size of each sorted segment */
  int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

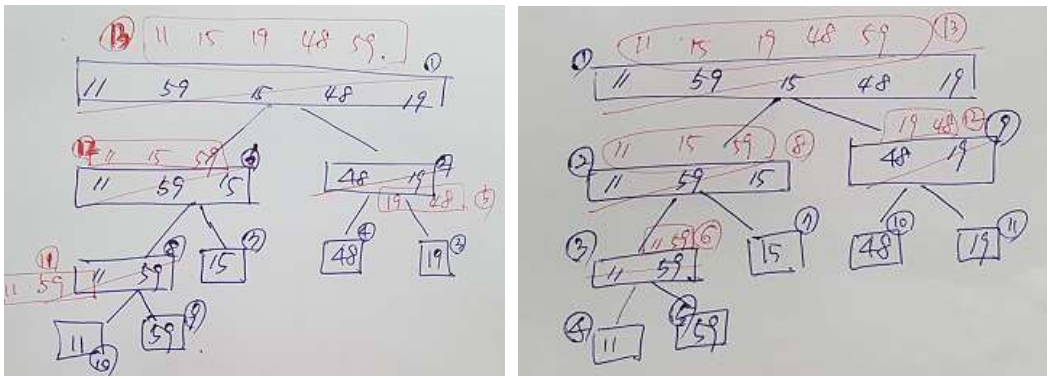
<실행예>



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 대해 RLV, 혹은 LRV 방식의 트리 순회를 통해 합병 정렬하는 과정을 각각 연습장에 적은 후 그 결과를 보고서에 넣어라. 단, downward tree, upward tree를 따로 구분하지 말고 하나로 표현하라. (2점)
① RLV 방식 ② LRV 방식

※ 입력 리스트가 (11, 59, 15, 48, 19)인 경우 작성 예
① RLV 방식 ② LRV 방식



파란색 번호는 rmergeSort 함수 호출을, 붉은색 번호는 listMerge 함수의 수행결과를 순서대로 나타냄. 이 예의 경우, 모두 13번의 함수 호출이 있음

※ 참고
교재의 Program 7.10 및 7.11에 의한 합병정렬은 recursion tree에 대해 RLV tree traversal을 통해 이루어진다. Program 7.10의 rmergeSort 함수의 마지막 문장을 보면 listMerge(a, link, rmergeSort(a, link, left, mid), rmergeSort(a, link, mid+1, right)) 가 반환하는 값을 다시 반환하는데, c언어에서 함수 인자는 오른쪽에서 왼쪽으로 평가되기 때문에 right half에 대한 rmergeSort 호출을 먼저 한 후 left half에 대한 rmergeSort 호출이 일어난다.

(2) (1)의 결과를 프로그램으로 확인해 보라. (8점)
<실행순서>
① 입력파일(input.txt)로부터 key를 읽어 들여 구조체 배열 a에 저장한다.
※ element 타입은 key 필드만으로 구성된 구조체를 재정의한 것으로 가정한다.

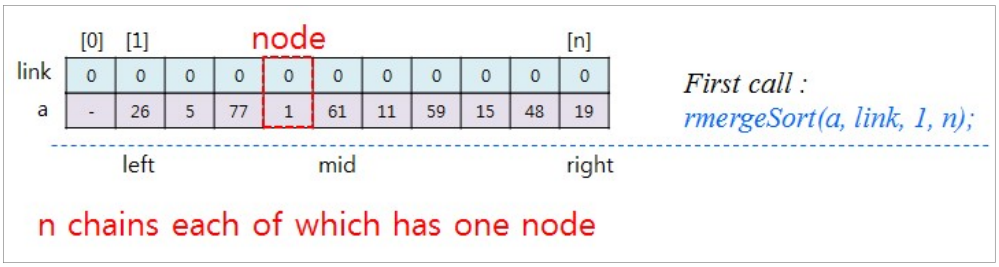
input.txt
10
26 5 77 1 61 11 59 15 48 19

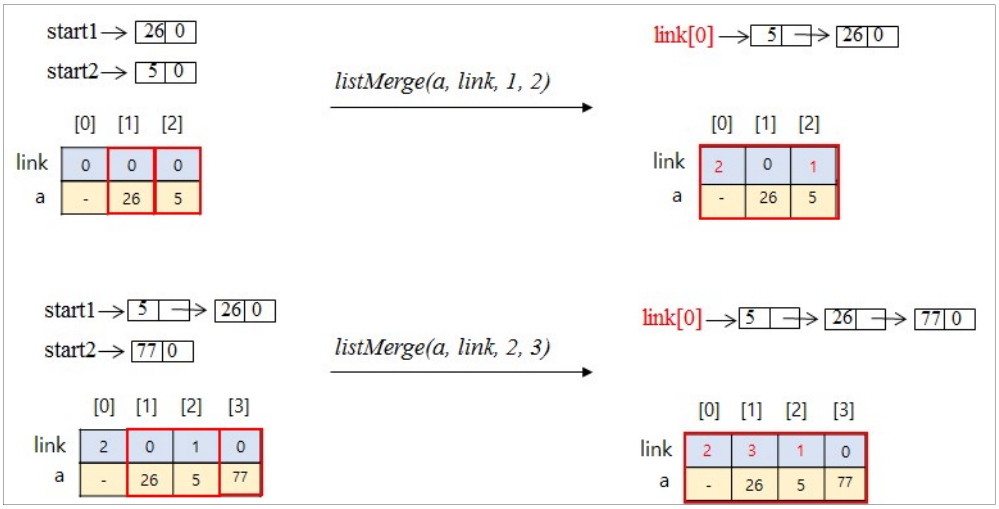
※ 첫 줄의 10은 입력키의 개수

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 실행예와 같이 합병정렬 과정을 볼 수 있도록 rmergeSort함수 등을 적절하게 수정하여 RLV, LRV 방식 두 가지 경우에 대해 각각 실행결과를 보여라.

<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 */
}
```

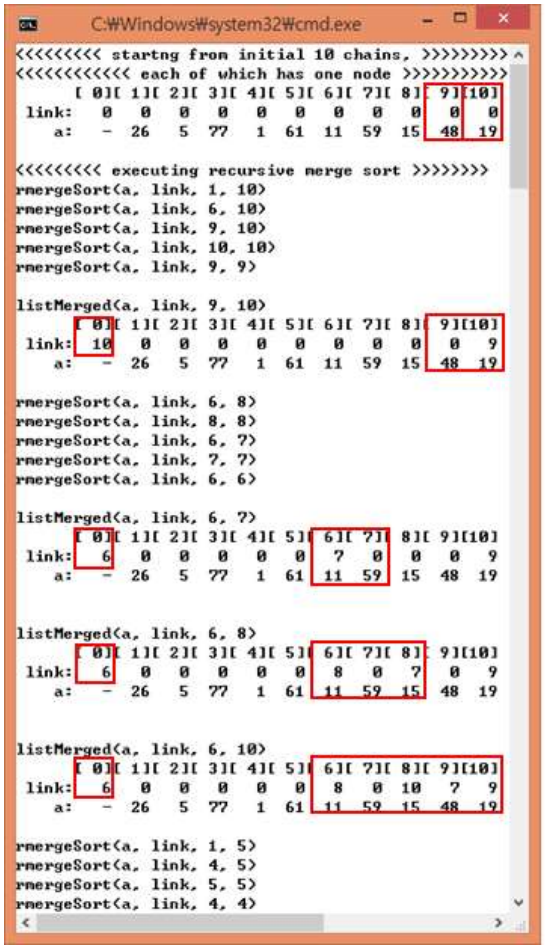
Program 7.10: Recursive merge sort

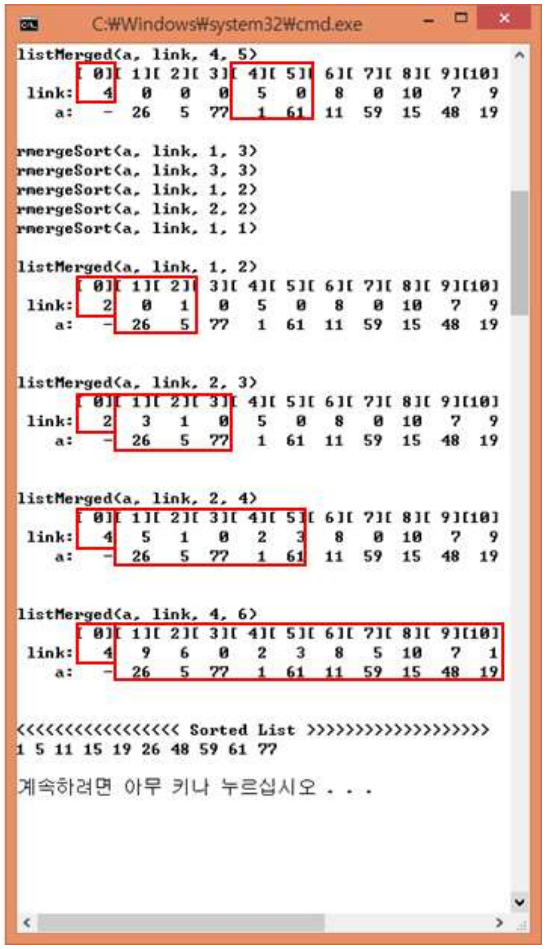
```
int listMerge(element a[], int link[], int start1, int start2)
{/* sorted chains beginning at start1 and start2,
    respectively, are merged; link[0] is used as a
    temporary header; returns start of merged chain */
   int last1, last2, lastResult = 0;
   for (last1 = start1, last2 = start2; last1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

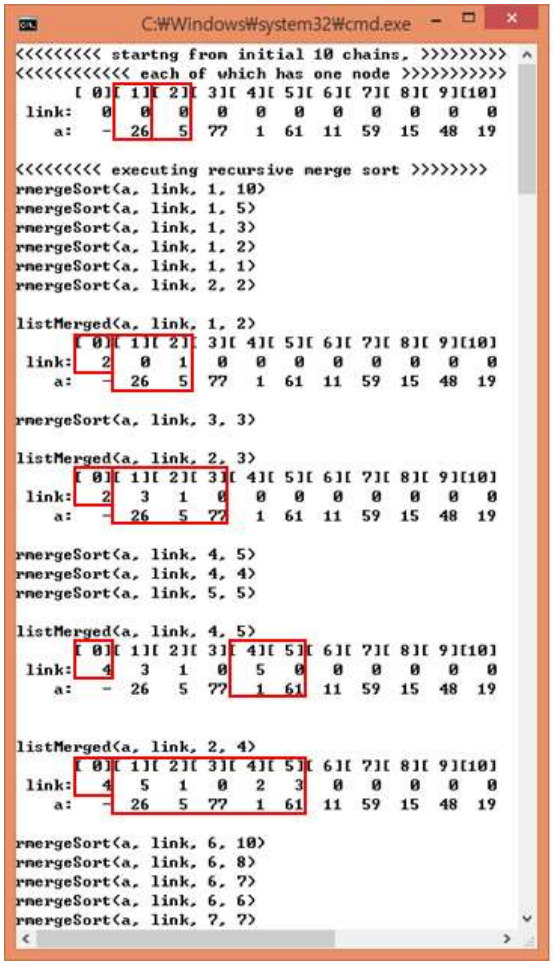
< 실행예 >

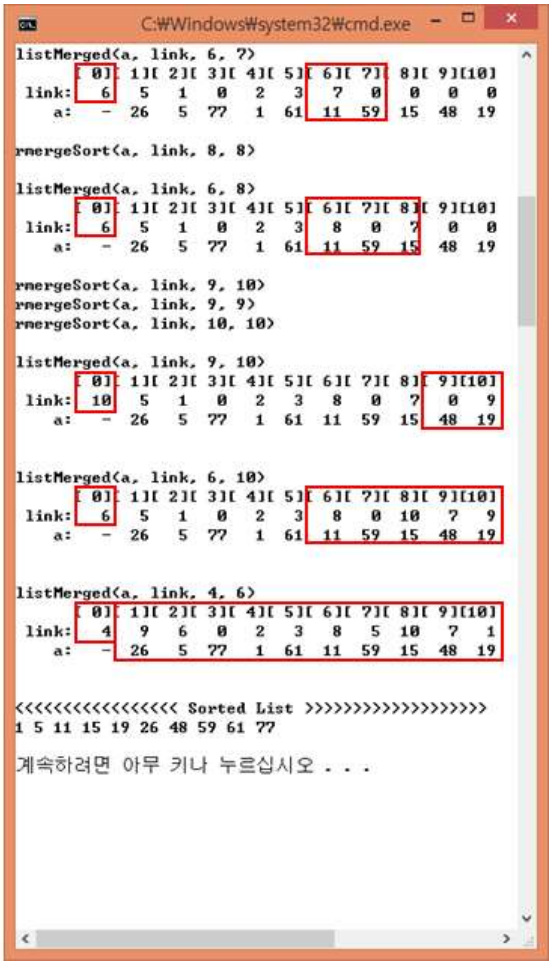
① recursive merge sort - RLV





② recursive merge sort - LRV





■ 제출 형식

- 솔루션/프로젝트 이름 : DS 20
- 소스파일 이름 : 1.c, 2.c
- 각 소스파일에 주석처리

  “학번 이름”

  “본인은 이 소스파일을 복사 없이 직접 작성하였습니다.”
- 보고서 :

  ① 1번 - 실행결과 화면 캡쳐

  ② 2번 - 입력데이터의 RLV, LRV 순회에 대해 각각 recursion tree를 그리고 실행결과 화면 캡쳐하여 합병과정을 표시할 것
- 솔루션 정리 후 보고서 및 솔루션 폴더를 “학번.zip”으로 압축하여 과제 게시판에 제출

■ 주의

- 소스 복사로는 실력향상을 기대할 수 없습니다!!!

- **1차 마감 : 5월 26일(화) 자정**
- **2차 마감 : 5월 27일(수) 자정(만점의 80%)**