

자료구조응용

22. Recursive Merge Sort(10점)

2025.11.26

1. 다음 입력 리스트에 대해 재귀적인 합병정렬(recursive merge sort)을 수행하고자 한다.

입력 리스트 (26, 5, 77, 1, 61, 11, 59, 15, 48, 19)

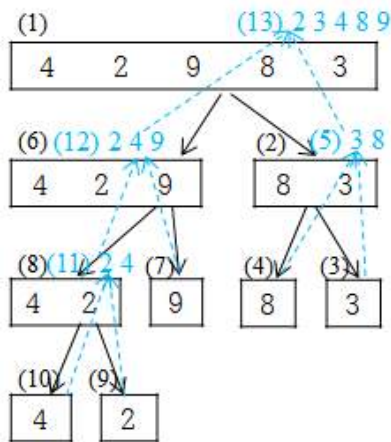
(1) recursion tree에 대해 RLV, 혹은 LRV 방식의 트리 순회를 통해 합병 정렬하는 과정을 각각 그린 후 그 결과를 보고서에 넣어라. 단, downward tree, upward tree를 따로 구분하지 말고 아래 그림과 같이 하나로 표현하라. (4점)

① RLV 방식

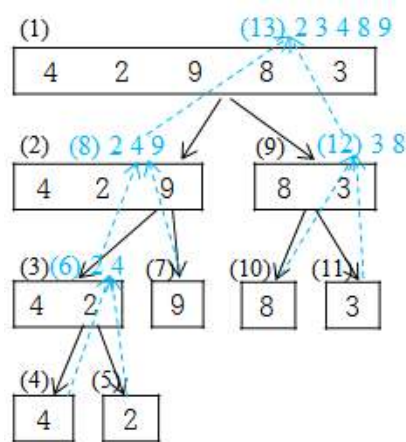
② LRV 방식

※ 입력 리스트가 (4, 2, 9, 8, 3)인 경우 작성 예

① RLV 방식



② LRV 방식



(2) (1)의 결과를 프로그램으로 확인해 보라. (6점)

<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

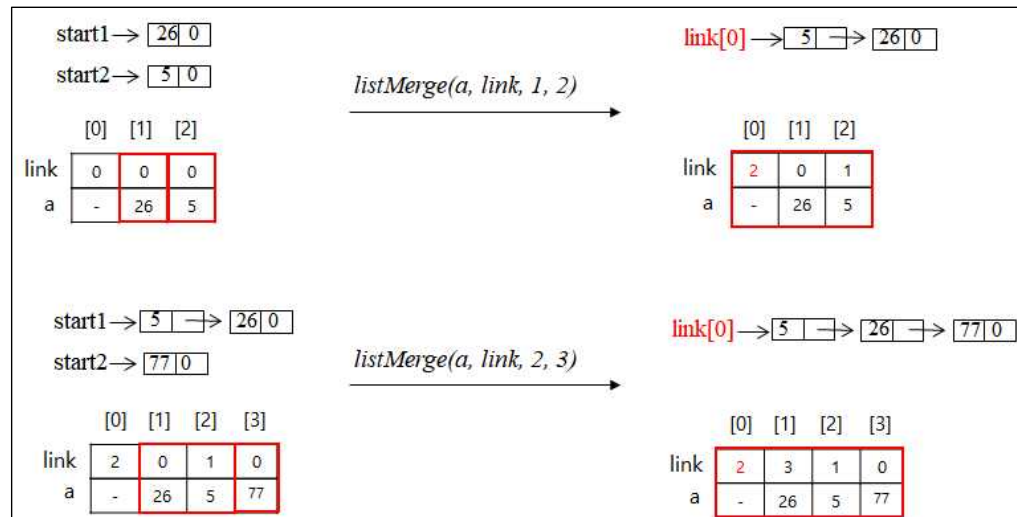
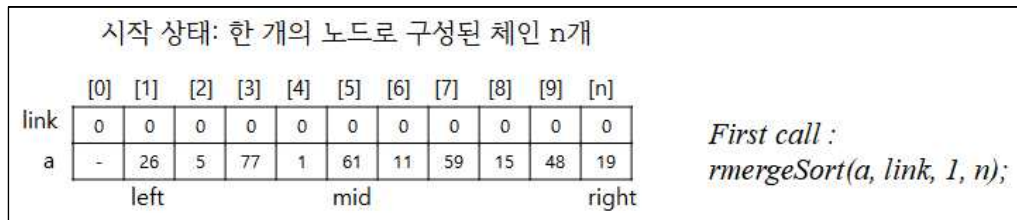
※ 첫 줄의 10은 입력키의 개수

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 실행 결과를 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

※ 주의

교재의 Program 7.10 및 7.11에 의한 합병정렬은 recursion tree에 대해 RLH 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.

```

// Program 7.10의 수정 함수
int rmergeSort(element a[], int link[], int left, int right)
{
    int mid, mergedSorted, leftSorted, rightSorted;

    if ( left >= right ) return left;
    mid = ( left + right ) / 2;

    // case 1: RLV
    rightSorted = rmergeSort(a, link, mid + 1, right);    // sort right half
    leftSorted = rmergeSort(a, link, left, mid);          // sort left half
    mergedSorted = listMerge(a, link, leftSorted, rightSorted );

    /*
    // case 2: LRV
    leftSorted = rmergeSort(a, link, left, mid);          // sort left half
    rightSorted = rmergeSort(a, link, mid + 1, right);    // sort right half
    mergedSorted = listMerge(a, link, leftSorted, rightSorted );
    */

    return mergedSorted;
}

```

```

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

① recursive merge sort - RLV

[illegible]

② recursive merge sort - LRV

```
C:\Windows\system32\cmd.exe - [X] X  
C:\Windows\system32\cmd.exe - [X] X  
  
<<<<<<<< starting from initial 10 chains. >>>>>>>>  
<<<<<<<< each of which has one node >>>>>>>>  
[ 0][ 1][ 2][ 3][ 4][ 5][ 6][ 7][ 8][ 9][10]  
link: 0 0 0 0 0 0 0 0 0 0 0  
a: - 26 5 77 1 61 11 59 15 48 19  
  
<<<<<<< executing recursive merge sort >>>>>>>>  
rmergeSort(a, link, 1, 10)  
rmergeSort(a, link, 1, 5)  
rmergeSort(a, link, 1, 3)  
rmergeSort(a, link, 1, 2)  
rmergeSort(a, link, 1, 1)  
rmergeSort(a, link, 2, 2)  
  
listMerged(a, link, 1, 2)  
[ 0][ 1][ 2][ 3][ 4][ 5][ 6][ 7][ 8][ 9][10]  
link: 2 0 1 0 0 0 0 0 0 0 0  
a: - 26 5 77 1 61 11 59 15 48 19  
  
rmergeSort(a, link, 3, 3)  
  
listMerged(a, link, 2, 3)  
[ 0][ 1][ 2][ 3][ 4][ 5][ 6][ 7][ 8][ 9][10]  
link: 2 3 1 0 0 0 0 0 0 0 0  
a: - 26 5 77 1 61 11 59 15 48 19  
  
rmergeSort(a, link, 4, 5)  
rmergeSort(a, link, 4, 4)  
rmergeSort(a, link, 5, 5)  
  
listMerged(a, link, 4, 5)  
[ 0][ 1][ 2][ 3][ 4][ 5][ 6][ 7][ 8][ 9][10]  
link: 4 3 1 0 5 0 0 0 0 0 0  
a: - 26 5 77 1 61 11 59 15 48 19  
  
listMerged(a, link, 2, 4)  
[ 0][ 1][ 2][ 3][ 4][ 5][ 6][ 7][ 8][ 9][10]  
link: 4 5 1 0 2 3 0 0 0 0 0  
a: - 26 5 77 1 61 11 59 15 48 19  
  
rmergeSort(a, link, 6, 10)  
rmergeSort(a, link, 6, 8)  
rmergeSort(a, link, 6, 7)  
rmergeSort(a, link, 6, 6)  
rmergeSort(a, link, 7, 7)
```

※ 위 실행예와 같이 각 단계의 합병 결과를 사각형으로 표시한 것을 보고서에 포함해야 함

■ 제출 형식

- 솔루션 이름 : DS 22
- 프로젝트 이름 : 1
- 각 소스파일에 주석처리
“학번 이름”
“본인은 이 소스파일을 다른 사람의 소스를 복사하지 않고 직접 작성하였습니다.”
- 제출 파일
 - ① 소스코드와 실행 결과가 보이도록 화면을 캡처한 보고서 파일(“학번.pdf”)
 - ※ 한글 [파일 → pdf로 저장하기...] 메뉴 사용
 - ② C 소스 파일을 하나의 디렉터리에 모아 압축한 파일 (“학번.zip”)
 - ※ “학번.pdf”와 “학번.zip”을 하나로 압축하지 말고 별도 파일로 제출

■ 주의

- 소스 복사로는 실력 향상을 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

-