자료구조응용

21. Sorting: heap sort, radix sort (15점)

2020/5/27 (수)

- 1. 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. <u>매번 adjust를 수행한 직후의 상태</u>에 대해서만 트리를 그리면 된다. ②는 정렬된 데이터도 같이 표현하라. (2점)
 - * ①은 강의 슬라이드 25, ②는 강의슬라이드 26 페이지 참고
 - * 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 힙정렬을 실행한다.
- * adjust 함수 수행마다 배열(a)의 인덱스 순서대로 key값을 화면에 출력한다.
- ※ 출력함수를 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,l,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))</pre>
       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>



- 2. 다음 입력 리스트에 대해 기수정렬(radix sort)을 수행하고자 한다. 입력 리스트 (12, 2, 16, 30, 8, 28, 4, 10, 20, 6, 18)
- (1) 기수정렬 과정을 Figure7.9((a), (b), (c), (d))와 같이 순서대로 그림을 그리고 나타내어라. 단, r=10을 사용하라. (2점)
- ※ 연습장에 적은 후 사진을 찍어도 되며 그 결과를 보고서에 넣을 것
- (2) (1)의 결과를 프로그램으로 확인해 보라. (8점)

<실행순서>

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

input.txt	
2 11	첫째 줄: d, n
12 2 16 30 8 28 4 10 20 6 18	둘째 줄: Key 리스트

- * element 타입은 key 필드만으로 구성된 구조체를 재정의한 것으로 가정한다.
- * 튜터는 다른 d 입력에 대해 테스트하여야 한다.
- ② 각 레코드의 key에 대해 기수정렬을 실행한다.
- ** (a)(b)(c)(d) 각 단계가 끝난 후의 체인의 Key값을 link 순서대로 화면에 출력하라.
- ③ 정렬 결과를 파일(output.txt)에 저장한다.

<구현세부사항>

- ① r = 10으로 고정해서 사용
- ② printList 함수
- a, link 배열에 대해 인덱스 순서대로 출력
- first를 출력
- 정렬된 결과를 chain 순서대로 출력
- ③ printQueues 함수
- front, rear 배열에 대해 인덱스 순서대로 출력
- ④ digit 함수
- 파일로부터 입력된 임의의 d에 대해서도 실행되도록 할 것
- 필요하다면 함수의 파라미터를 추가할 수 있음
- r이 10인 경우에 대해서만 구현하면 됨
- ⑤ radixSort 함수
- front, rear 배열 선언 시 배열크기에 변수를 사용할 수 없음 (에러)
- front, rear 배열을 r 크기만큼 동적 할당 받는 것으로 수정할 것
- 이때, 교재의 MALLOC 매크로함수 대신 C 라이브러리 함수 malloc을 사용

- front[i]를 0으로 초기화 하는 부분에서 rear[i]도 0으로 같이 초기화할 것 (front[i]만 0으로 초기화해도 알고리즘에는 문제가 없음. rear[i]를 출력하기 위함임) - front[i] = 0 인 큐는 empty queue로 간주됨

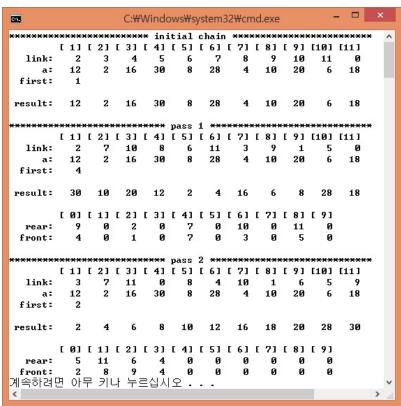
```
int radixSort(element a[], int link[], int d, int r, int n)
{/* sort a[1:n]) using a d-digit radix-r sort, digit(a[i],j,r)
   returns the jth radix-r digit (from the left) of a[i]'s key
   each digit is in the range is [0,r); sorting within a digit
   is done using a bin sort */
   int front[r], rear[r]; /* queue front and rear pointers */
   int i, bin, current, first, last;
   /* create initial chain of records starting at first */
   first = 1;
   for (i = 1; i < n; i++) link[i] = i + 1;
   link[n] = 0;
   for (i = d-1; i >= 0; i--)
   {/* sort on digit i */
      /* initialize bins to empty queues */
      for (bin = 0; bin < r; bin++) front[bin] = 0;
      for (current = first; current; current = link[current])
      {/* put records into queues/bins */
          bin = digit(a[current],i,r);
          if (front[bin] == 0) front[bin] = current;
          else link[rear[bin]] = current;
          rear[bin] = current;
      /* find first nonempty queue/bin */
      for (bin = 0; !front[bin]; bin++);
      first = front[bin]; last = rear[bin];
      /* concatenate remaining queues */
      for (bin++; bin < r; bin++)
          if (front[bin])
             {link[last] = front[bin]; last = rear[bin];}
      link[last] = 0;
   return first;
}
```

Program 7.14: LSD radix sort

<실행결과>

Case 1:



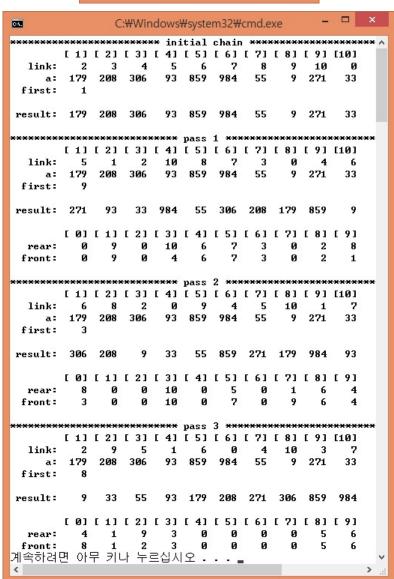


Self Check

- ① 각 단계에서 first부터 시작해서 노드 링크를 따라가며 키를 출력하면 result가 나오는가?
- ② pass 1에서 front[0] = 4, rear[0] = 9 의 의미는?
- ③ pass 2에서 front[4] = 4, rear[4] = 4 의 의미는?
- ④ 각 단계에서 queue 정보로부터 link 배열 값을 결정하는 과정을 따라가 보라.

Case 2:





Self Check

교재 Figure 7.9의 (a)(b)(c)(d)에서의 결과와 일치하는지 확인하라.

■ 제출 형식

- 솔루션/프로젝트 이름 : DS 21

- 소스파일 이름 : 1.c, 2.c

- 각 소스파일에 주석처리

"학번 이름"

"본인은 이 소스파일을 복사 없이 직접 작성하였습니다."

- 솔루션 정리 후 <u>보고서 및 솔루션 폴더</u>를 "학번.zip"으로 압축하여 과제 게시판에 제출

■ 주의

- 소스 복사로는 실력향상을 기대할 수 없습니다!!!

- 1차 마감 : 5월 28일(목) 자정

- 2차 마감 : 5월 29일(금) 자정(만점의 80%)