자료구조 13주차 과제

제출일: 2022-11-30
학번/이름: 2016121150 / 윤준영

문1) lab_13.c 프로그램에 대하여 다음에 답하시오.

1) 다음 동작을 하도록 main 함수에 코드를 추가하시오. (포함된 함수를 사용) (2점)

* 12, 32, 79, 11, 22 삽입
* 23 검색 (있/없으면 있/없다고 출력)
* 32 검색 (있/없으면 있/없다고 출력)
* 현재 해쉬 테이블 내용 출력

다음과 같이 main 함수를 추가하여 구현하였다. hsc_create는 첫 버킷의 주소, 요소의 개수 np를 가리키는 주소, 버킷의 개수 N을 받아 np를 0으로 초기화한 후 N개의 버킷을 NULL로 초기화한다. hsc_insert는 추가할 key, 첫 bucket, 요소의 개수 np 주소를 받아 key값을 가지는 node를 할당한 후 hash_func를 이용하여(10으로 나눈 나머지) key값에 해당하는 버킷의 next에 노드를 추가한 후 새 노드의 next는 이전의 버킷의 next로 지정한다. 그 후 np를 1 더한다. hsc_search는 찾을 값과 첫 버킷, np의 주소를 받아 해당하는 값이 있는 버킷을 next해가며 찾아 반환한다. 만약 없다면 NULL을 반환하게 된다. hsc_list함수는 첫 버킷과 버킷의 개수를 받아 처음부터 마지막 버킷까지 반복하여 버킷 next 노드부터 NULL이 나올 때까지 next로 이동하며 key값을 출력한다.
실행 결과는 우측과 같다.

```
124  void main(void)
125  {
126      node* bucket;
127      int nitem = 0;
128      hsc_create(&bucket, &nitem, TABLE_SIZE);
129
130      hsc_insert(12, bucket, &nitem);
131      hsc_insert(32, bucket, &nitem);
132      hsc_insert(79, bucket, &nitem);
133      hsc_insert(11, bucket, &nitem);
134      hsc_insert(22, bucket, &nitem);
135
136      int f;
137      printf("\n%d ", f=23);
138      if (hsc_search(f, bucket, &nitem)==NULL) printf("없음");
139      else printf("있음");
140      printf("\n%d ", f=32);
141      if (hsc_search(f, bucket, &nitem)==NULL) printf("없음");
142      else printf("있음");
143
144      hsc_list(bucket, TABLE_SIZE);
145      hsc_deleteall(bucket, &nitem, TABLE_SIZE);
146      free(bucket);
147  }
```

```
23 없음
32 있음
Bucket 0 :
Bucket 1 : -> 11
Bucket 2 : -> 22 -> 32 -> 12
Bucket 3 :
Bucket 4 :
Bucket 5 :
Bucket 6 :
Bucket 7 :
Bucket 8 :
Bucket 9 : -> 79
```

2) 1)에서 22를 검색한 결과를 다음과 같이 보여주도록 hsc_search 함수를 수정하시오. (제일 처음 버킷(노드)은 0번째 버킷임) (2점)

22 : _번째 버킷 _번째 노드에 있음

다음과 같이 hsc_search 함수를 변경하였다. 사용하는 버킷을 지정하는 hash_func함수의 결과를 정수 b에 저장한 후, 노드를 찾을 때에 변수 n을 지정하여 반복할 때마다 +1하여 노드가 NULL이 아니면 버킷과 노드의 인덱스를 출력한다. 만약 찾은 노드 t가 NULL이면 찾지 못했다고 출력하였다. 우측은 함수의 실행 결과이다. 22는 2번 버킷 0번째에 있는 것을 확인할 수 있다.

```
node* hsc_search(int key, node a[], int* np)
{
    node* t;
    int b = hash_func(key);
    t = a[b].next;
    int n = 0;
    while (t != NULL && t->key != key) {
        t = t->next;
        n += 1;
        }
    if (t != NULL) printf("%d번째 버킷 %d번째 노드에 있음", b, n);
    else printf("%d없음", key);
    return t;
}
```

```
2번째 버킷 0번째 노드에 있음
Bucket 0 :
Bucket 1 : -> 11
Bucket 2 : -> 22 -> 32 -> 12
Bucket 3 :
Bucket 4 :
Bucket 5 :
Bucket 6 :
Bucket 7 :
Bucket 8 :
Bucket 9 : -> 79
```

3) 1)의 수행에서 버킷에 삽입이 집중되어 성능이 저하되었다. 위 프로그램의 해쉬 함수를 적절히 수정해서 위 입력에 대한 성능이 향상되도록 하시오. (1점)

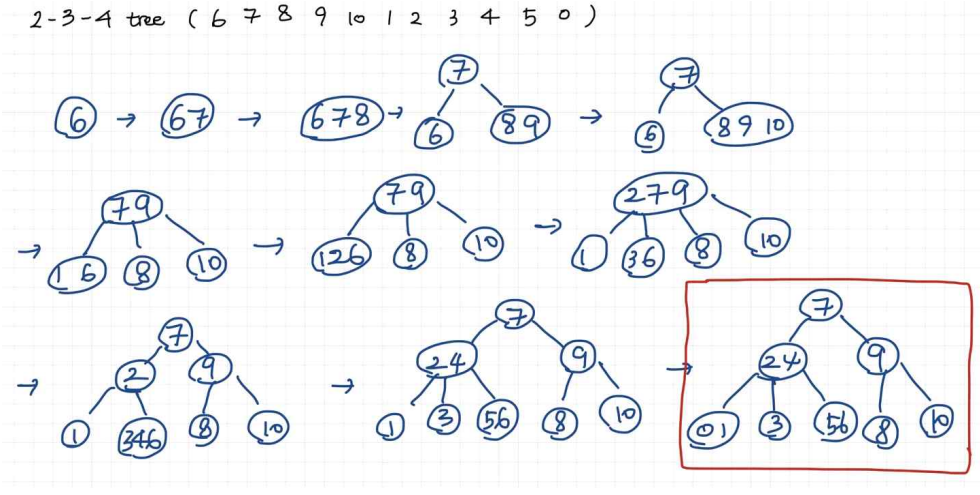
다음과 같이 hash_func 함수를 수정하였다. 입력하는 두 자리의 수를 더한 뒤 그 값의 1의 자리 수가 버킷의 번호가 된다. 실행 결과는 우측과 같다. 잘 분배된 것을 확인할 수 있다.

```
int hash_func(int key)
{
    int h;
    h = key / TABLE_SIZE + key % TABLE_SIZE;
    h = h % TABLE_SIZE;
    return h;
}
```

```
Bucket 0 :
Bucket 1 :
Bucket 2 : -> 11
Bucket 3 : -> 12
Bucket 4 : -> 22
Bucket 5 : -> 32
Bucket 6 : -> 79
Bucket 7 :
Bucket 8 :
Bucket 9 :
```

문2) 2-3-4 나무에 대하여 다음 물음에 답하시오.

1) 6,7,8,9,10,1,2,3,4,5,0 의 순서대로 원소를 삽입한 결과를 그리시오.(1점)



678

7
6 8910

7
126 8910

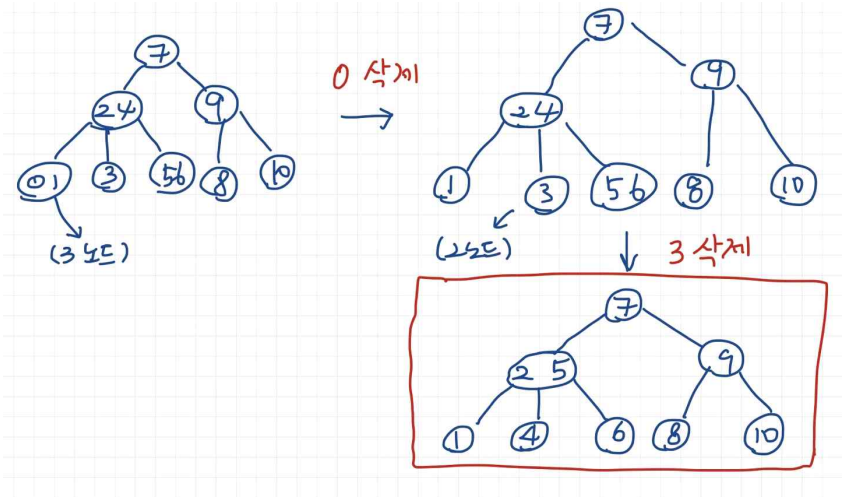
27
1 36 8910

27
1 346 8910

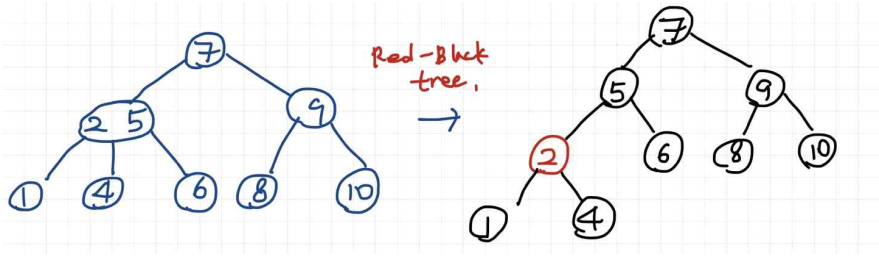
247
1 3 56 8910

4
2 7
01 3 56 8910

2) 위 트리에서 0과 3을 삭제한 결과를 그리시오.(1점)

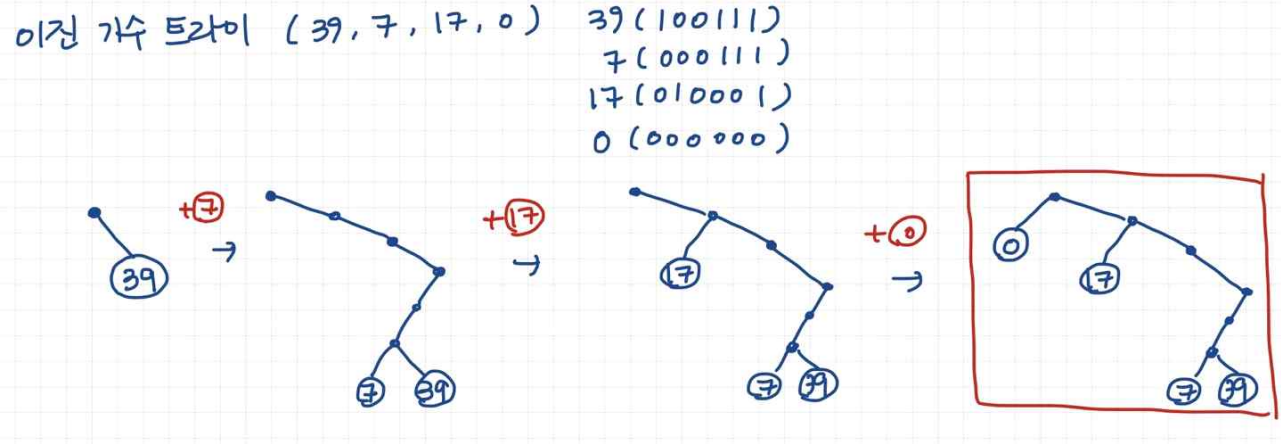


3) 위 트리를 red-black 트리로 나타내시오. (1점)



문3) 이진 기수 트라이에 대하여 다음 물음에 답하시오.

1) 이진 기수 트라이에 39, 7, 17, 0을 삽입한 결과를 그리시오.(1점)



2) 1)의 구조에서 17 -> 7의 순서대로 삭제한 결과를 각각 그리시오.(1점)