Homework 4

자료구조. 2024 1학기. 총 5 문제. 제출일: 6/11

1. Key 값이 다음과 같은 자료들을 아래의 해싱(hashing) 함수 h(x)에 따라 해시 테이블에 차례로 삽입하고 탐색을 수행한다고 하자. 해시 테이블의 크기는 b=10, s=1 이다.

{371, 323, 173, 199, 344, 679, 979}
h(x) = x % 10
Hash table:

i	key
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

1) Linear probing을 하였을 때, 자료가 모두 삽입된 후 해시 테이블을 보이고, successful search의 평균 키 비교 회수는 얼마인지 계산하시오.

2) h'(x) = 7 – (x % 7)로 double hashing을 하였을 때, 자료가 모두 삽입된 후 해시 테이블을 보이고, successful search의 평균 키 비교 회수는 얼마인지 계산하시오.

3) Chaining을 하였을 때, 자료가 모두 삽입된 후 해시 테이블을 보이고, successful search의 평균 키 비교 회수는 얼마인지 계산하시오.

2. 정수 key들이 다음과 같이 저장된 리스트를 다음 각 알고리즘으로 정렬하는 과정을 보이시오.

7, 13, 14, 2, 6, 11, 9, 4, 15, 1, 5, 10, 16, 3, 12, 8

1) 퀵 정렬(quick sort) : 피벗(pivot)은 배열의 첫 번째 원소로 하고, 매번 partition이 일어난 후의 배열 내용을 보이시오.

2) 합병 정렬(merge sort) : merge가 size 2, size 4, size 8, size 16 인 리스트를 만드는 각 단계 후의 리스트 내용을 보이시오.

3) Key가 N 개 있을 때, merge sort의 worst-case time complexity는 O(N log N) 임을 보이시오.

3. 아래의 알파벳 소문자 3자로 주어지는 key들을 기수 정렬(radix sorting) 한다고 하자.
r = 26 일 때, Sorting이 진행되는 3 단계 과정을 나타내보시오.

1) can, nut, and, hat, fog, god, has, log, sir, far, lot, car, big, not, sun, get, bat, see

2) 위와 같은 key가 N 개 있을 때, 이 radix sorting의 time complexity는 얼마인가?

4. (Programming) Hashing Simulation

Linear probing을 수행하는 insert(int key, int value), search(int key) 함수를 작성하시오.
다음과 같은 조건에서 random key를 생성하여 loading density = 0.5 (500 keys)와
0.9 (900 keys)인 경우에 각각 simulation을 수행하여, successful search의 평균 key 비교 회수와 최악의 경우 key 비교 회수를 구해보시오. Search는 모든 key가 insert된 후에 수행된다.

- hash table : b = 1001, s = 1
- hash function : h(key) = key % b
- key : random integers in [0, 9999]
- value : key+10000 (to check the correctness of search result)

```
#include <stdlib.h>
#define TABLE_SIZE 1001

typedef struct {
       int    key;
       int    value;
} element;
element  ht[TABLE_SIZE];

void linear_probing_insert(element item);
void linear_probing_search(element item);
```

5. (Programming) Sorting Simulation

삽입 정렬과 퀵 정렬 프로그램을 작성하시오. 다음과 같은 갯수의 random interger를 생성하여 각각의 경우에 대해 insertion sort와 quick sort를 수행하고, 정렬이 끝날 때 까지 비교 회수와 실행 시간을 측정하여 이론상 time complexity와 비교해 보시오.

```
#include <stdlib.h>
#define SIZE 1000   // random interger 개수. SIZE = 1000, 2000, ... , 10000

int list[SIZE];

void insertion_sort(int list[], int n);
void quick_sort(int list[], int left, int right);
```