HCMC University of Technology
Faculty of Computer Science & Engineering

Data Structures and Algorithms

Tut 7 – Heap & Hash

Heap

Question 1

a. Building a heap:

Given the following arrays:

```
int maxHeap1[8] = {56, 45, 4, 77, 60, 34, 35, 22};
int maxHeap2[9] = { 1, 3, 5, 7, 9, 2, 4, 6, 8};
```

Build a max heap for each of them. Draw the final result.

b. Delete heap:

After finishing building the two heaps above, apply the **DeleteHeap** operation on each of them **twice**. Draw the result.

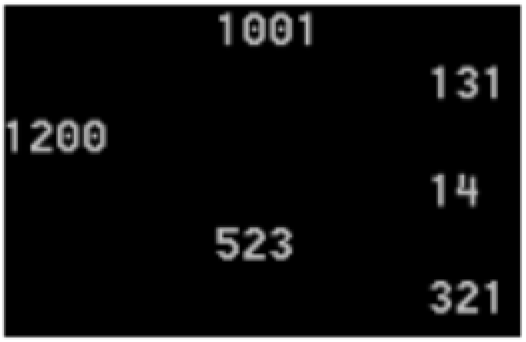
Question 2

Printing a heap as a rotated tree:

Write a program that prints the heap as a tree rotated 90 degree counter clockwise. For example, with the heap:

```
int maxHeap[6] = { 1200, 523, 1001, 321, 14, 131 };
```

We would have the following printing result:



```
            1001
                        131
1200
                        14
            523
                        321
```

Question 3

a. Heap property checking: write a function to check if an array is a max heap:

```
bool IsMaxHeap(int *arr, int size)
```

b. Delete an arbitrary node: The provided **deleteHeap** function is only able to delete the root of a heap. Write another function that allows you to delete any node in a heap:

```
bool DeleteHeapNode(int *&maxHeap, int delPosition, int &size,
int & dataOut)
```

Hash

Question 4

Given the following hash and probing function:

$$h(key) = key \% 256 + 1$$

$$hp(key) = key$$

a. Collisions:
You have a list **L1** of 1024 keys ranged from 1 to 1024. If you start hashing them one by one using the above functions, how many collisions will occur ?

b. More collisions:
Now, instead of the above probing function, you use the following linear probing function to resolve the collisions:

$$hp(key) = (h(key) + i) \% 512$$

How many probing you need to do if you have to hash the list **L2** = {0, 256, 512, 513, 1026, 1025}? Draw the hash table after you insert 512, 1025.

Question 5

Given the following four hash functions:

$$h_1(key) = key \% 997 + 1$$

$$h_2(key) = [digit_{end}][digit_{end-2}][digit_{end-3}]$$

$$h_3(key) = fold\ shift\ of\ 9\ digits$$

$$h_4(key) = (key \times 2016 + 11) \% 997 + 1;$$

For example:

$$h_1(224) = 224 \% 997 + 1 = 225$$

$$h_1(1000) = 1000 \% 997 + 1 = 3$$

$$h_2(2016) = [6][0][2] = 602$$

$$h_2(2) = h_2(0002) = [2][0][0] = 200$$

$$h_3(1) = h_3(000\ 000\ 001) = 000 + 000 + 001 = 1$$

$$h_3(135792468) = 135 + 792 + 468 = 1395 = 395$$

$$h_4(2) = (2 \times 2016 + 11) \% 997 + 1 = 56$$

a) **Implement the above hash functions**

b) **Performance of hash functions:**

Assume that you have a list **L3** of 997 positive integer values. The values of **L3** are defined using the following rule:

$$L3(0) = 0$$
$$L3(n) = L3(n-1) + 17 + (-1)^n \times 11,\ n < 997$$

With the list **L3**, which hash function in question a) would give us the most reliable performance? (For simplicity, fewer collisions mean better performance).

Note: to avoid more collisions after probing, which would further complicate our measurement, we assume that the probing function will be $hp(key) = key + 999 \times 3$.