CSC3100 Solution 3

Preface

This reference solution is compiled by CSC3100 Teaching Team in 2022-2023 Fall semester. If you find any mistakes or typos, please contact us by email 120090350@link.cuhk.edu.cn.

Problem 1

> Give an example that deletion (always uses successors — the left-most node of the right child — to take the position of the deleted internal node) might make the left sub-trees deeper than the right. Try to give some solutions to avoid the imbalance.

For example, given a binary search tree:

```
     4
    /  \
  2     5
 / \
1   3
```

If we want to delete 4, then it gives a new binary search tree like

```
     5
    /
  2
 / \
1   3
```

Thus, it gives the imbalance.
To avoid the imbalance, rotations are needed to rebalance.
For example, a single right rotation gives a balanced binary search tree like:

```
     2
    /  \
  1     5
       /
      3
```

Problem 2

> Given an array representing a max-heap, please convert it into a min-heap.
>
> For example, given a max-heap:
> ```
>       9
>     /   \
>    4     7
>   / \   / \
>  1  -2 6   5
> ```
> It gives a min-heap like
> ```
>       -2
>     /   \
>    1     5
>   / \   / \
>  9   4 6   7
> ```
> Write code or pseudocode and also analyze the time complexity. An algorithm with $O(n)$ is expected (otherwise you could get a maximum score of 15).

A fast algorithm to build the binary heap:

```
HEAPIFY(n)
    for i = floor(n / 2) downto 1
        PERCOLATE-DOWN(i)
```

It returns a min-heap given any n elements, not necessarily given a max-heap. The time complexity is $O(n)$.

Remark. The details of the implementation of the method PERCOLATE-DOWN and time complexity analysis are omitted here. You could check the lecture slides for reference.

Problem 3

> We usually define a modulo function as a hash function, for example, $h(x) = x \mod p$. Is there any benefit to using a prime number for p? If yes, explain why and in which scenario prime numbers are better.

For nonrandom data, defining a modulo function that uses a prime number for p as a hash function will produce the widest distribution of integers to the index. Therefore, it will greatly reduce the occurrence of collisions.

Problem 4

> Please draw the final hash tables when inserting a sequence of numbers $[6, 12, 29, 28, 34, 11, 23, 7, 0, 33, 30, 45, 10001]$ using the two following ways to resolve collision separately. In other words, you have to give two hash tables, one using linear probe and the other one using double hashing.
>
> (1) Linear probe, $h(k) = k \mod 17$
>
> (2) Double hashing $h(k) = k \mod 17$ and $h'(k) = 1 + k \mod 5$
>
> Which one do you prefer? Please explain the reason (no standard answers here).

(1) Linear probe gives the final hash table like: $[34, 0, 45, \emptyset, \emptyset, 10001, 6, 23, 7, \emptyset, \emptyset, 28, 12, 29, 11, 30, 33]$

More specifically, the hash value $h(k)$ is $[6, 12, 12, 11, 0, 11, 6, 7, 0, 16, 13, 11, 5]$

$h(6,0) = 6 \rightarrow \checkmark$
$h(12,0) = 12 \rightarrow \checkmark$
$h(29,0) = 12 \rightarrow \text{✗}$
  $h(29,1) = 13 \rightarrow \checkmark$
$h(28,0) = 11 \rightarrow \checkmark$
$h(34,0) = 0 \rightarrow \checkmark$
$h(11,0) = 11 \rightarrow \text{✗}$
  $h(11,1) = 12 \rightarrow \text{✗}$
  $h(11,2) = 13 \rightarrow \text{✗}$
  $h(11,3) = 14 \rightarrow \checkmark$
$h(23,0) = 6 \rightarrow \text{✗}$
  $h(23,1) = 7 \rightarrow \checkmark$
$h(7,0) = 7 \rightarrow \text{✗}$
  $h(7,1) = 8 \rightarrow \checkmark$
$h(0,0) = 0 \rightarrow \text{✗}$
  $h(0,1) = 1 \rightarrow \checkmark$
$h(33,0) = 16 \rightarrow \checkmark$
$h(30,0) = 13 \rightarrow \text{✗}$
  $h(30,1) = 14 \rightarrow \text{✗}$
  $h(30,2) = 15 \rightarrow \checkmark$
$h(45,0) = 11 \rightarrow \text{✗}$
  $h(45,1) = 12 \rightarrow \text{✗}$
  $h(45,2) = 13 \rightarrow \text{✗}$
  $h(45,3) = 14 \rightarrow \text{✗}$
  $h(45,4) = 15 \rightarrow \text{✗}$
  $h(45,5) = 16 \rightarrow \text{✗}$
  $h(45,6) = 0 \rightarrow \text{✗}$
  $h(45,7) = 1 \rightarrow \text{✗}$

$h(45, 8) = 2 \to ✓$
$h(10001, 0) = 5 \to ✓$

(2) Double hashing gives the final hash table like:
$[29, 0, \emptyset, \emptyset, \emptyset, 34, 6, 7, \emptyset, 10001, 23, 28, 12, 11, 30, 45, 33]$

More specifically, the hash value $h'(k)$ is $[2, 3, 5, 4, 5, 2, 4, 3, 1, 4, 1, 1, 2]$

$h(6, 0) = 6 \to ✓$
$h(12, 0) = 12 \to ✓$
$h(29, 0) = 12 \to ✗$
$h(29, 1) = 0 \to ✓$
$h(28, 0) = 11 \to ✓$
$h(34, 0) = 0 \to ✗$
$h(34, 1) = 5 \to ✓$
$h(11, 0) = 11 \to ✗$
$h(11, 1) = 13 \to ✓$
$h(23, 0) = 6 \to ✗$
$h(23, 1) = 10 \to ✓$
$h(7, 0) = 7 \to ✓$
$h(0, 0) = 0 \to ✗$
$h(0, 1) = 1 \to ✓$
$h(33, 0) = 16 \to ✓$
$h(30, 0) = 13 \to ✗$
$h(30, 1) = 14 \to ✓$
$h(45, 0) = 11 \to ✗$
$h(45, 1) = 12 \to ✗$
$h(45, 2) = 13 \to ✗$
$h(45, 3) = 14 \to ✗$
$h(45, 4) = 15 \to ✓$
$h(10001, 0) = 5 \to ✗$
$h(10001, 1) = 7 \to ✗$
$h(10001, 2) = 9 \to ✓$

Problem 5

> How to delete an item in a hashing table with linear probe and double hashing respectively? You do not need to write the code.

To delete an item in a hashing table with linear probe or double hashing, we could use the same manner, which is put an empty item where to be deleted.

Remark. (i) We cannot just empty the slot since we need to keep the chain structure.

For example, if we use linear probe with $h(k) = k \mod 17$, or, use double hashing with $h(k) = k \mod 17$ and $h'(k) = 1 + k \mod 5$, to insert 0 and 85, then we get the hash tables like: $[0, 85, \emptyset, \ldots, \emptyset]$

Next we want to delete 0 and 85, then we get the hash tables like: $[\emptyset, 85, \emptyset, \ldots, \emptyset]$

We fail to delete 85 because there is nothing at the index 0. So we encounter an error. To fix it, we just need to put an empty item at the index 0. Then the program will search for 85 as search for an empty slot when inserting.

(ii) We cannot just consider the items following the item to be deleted when using double hashing due to the same reason.

For example, if we use double hashing with $h(k) = k \mod 17$ and $h'(k) = 1 + k \mod 5$, to insert 0 and 51, then we get the hash tables like: $[0, \emptyset, 51, \emptyset, \ldots, \emptyset]$

Next we want to delete 0 and 51, then we get the hash tables like: $[\emptyset, \emptyset, 51, \emptyset, \ldots, \emptyset]$

We fail to delete 51 because there is nothing at the index 0. So we encounter an error. To fix it, we just need to put an empty item at the index 0. Then the program will search for 51 as search for an empty slot when inserting.

(iii) We cannot just use one of the items following the item to be deleted to fill in the gap when using linear probe due to the same reason.

For example, if we use linear probe with $h(k) = k \mod 17$, to insert 0, 17, and 34, then we get the hash tables like: $[0, 17, 34, \emptyset, \ldots, \emptyset]$

Next we want to delete 0 and 34, then we get the hash tables like: $[17, \emptyset, 34, \emptyset, \ldots, \emptyset]$

We fail to delete 34 because there is nothing at the index 1. So we encounter an error. To fix it, we need to use 34 to fill in the gap made by 17 when deleting 0. More generally, we need to repeat filling in the gap until we finish considering a consecutive sequence of items. This method is indeed correct but not recommended since it is a little bit complicated.