CMSC 420: Homework 1: Hashing & Vanilla Trees

- Handed out: 2/12 Due: 2/19 before class starts. Late homeworks will not be accepted.
- When in doubt, make VALID assumptions (if necessary), state your assumptions, and proceed!
- In general, blank spaces given below indicate the space I used to solve the answers. You are not discouraged in writing answers elsewhere.

1. (10 points) Complete, correct, or answer TRUE or FALSE as necessary. Explain why in all cases.
   (a) An open addressed hashing scheme with hash function \( h(k) = (ak + b) \mod m \) is employed on a table of size \( m \). The following statements are TRUE in a correct implementation:
      i. \( m \) is not a prime number.
      ii. If \( m \) is a prime number, it is possible that all keys hash to the same location.
      iii. If \( m \) is a prime number, insert\( (k) \) will succeed.
      iv. If \( m \) is a prime number, and the load factor is 0.5, insert\( (k) \) will succeed.

2. (20 points) You are given the hashing function \( h(k) = k \mod 11 \), and an array \( A \) of size 11 (start index is 0) to implement a dictionary \( S \). Accessing a location of \( A \), whether it contains an item in \( S \) or not, counts as a probe.
   (a) Give a linear probing ordered hashing scheme with the property that if FIND\( (k) \) is issued, and the probe sequence is \( h(k) = p_1, p_1, \ldots, p_m \) the condition \( A[p_i] < k \) will enable you to terminate the search without making additional probes at locations \( p_{i+1}, \ldots, p_m \).
   \[ \text{INIT()} \]
   \[ \text{INSERT(x)} \]
   \[ \text{DELETE(x)} \]
   \[ \text{FIND(x)} \]
   \[ \text{MAXIMUM(S)} \]
(b) How many probes are made in your scheme for each \textbf{FIND} given \textbf{INSERT}(31), \textbf{INSERT}(16), \textbf{INSERT}(12), \textbf{FIND}(7), \textbf{FIND}(42), \textbf{DELETE}(16), \textbf{FIND}(42), \textbf{INSERT}(7), \textbf{FIND}(7)

(c) How many (in the best/worst case) probes will be needed to compute the maximum in $S$? Why?

Best case |
<table>
<thead>
<tr>
<th>Worst Case</th>
</tr>
</thead>
</table>

3. (20 points) Consider the following algorithm used in searching an open addressing hashing scheme where $h(k)$ maps to an integer in the range $[0..m-1]$ and $m$ is a power of 2.

(a) $i \leftarrow h(k); j \leftarrow 0.$

(b) Probe position $i$ for the desired key $k$. If found, or if position is \textbf{EMPTY} terminate.

(c) $j \leftarrow (j + 1) \mod m; i \leftarrow (i + j) \mod m$. Return to Step (b).

Show that the algorithm is an instance of quadratic probing of the form $h(k, i) = (g(k) + c_1i + c_2i^2) \mod m$ where $c_1 \in Z$, $c_2 \neq 0 \in Z$.

(Optional): Show that every location in the hash table is probed.

4. (20 points) Consider the following situation at the time of inserting Rudy.

<table>
<thead>
<tr>
<th>Rudy</th>
<th>Tim</th>
<th>Tim</th>
<th>Alan</th>
<th>Jay</th>
<th>Katy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tim</td>
<td>Rita</td>
<td>Alan</td>
<td>Tim</td>
<td>Katy</td>
<td>Rita</td>
</tr>
<tr>
<td>Alan</td>
<td>Ron</td>
<td>Rita</td>
<td>Kim</td>
<td>Alex</td>
<td>Kim</td>
</tr>
<tr>
<td>Jay</td>
<td>Ruth</td>
<td>Ruth</td>
<td>Jay</td>
<td>Bob</td>
<td>Kim</td>
</tr>
<tr>
<td>Katy</td>
<td>-</td>
<td>-</td>
<td>Katy</td>
<td>Kim</td>
<td>-</td>
</tr>
</tbody>
</table>

The adjacent data is to be interpreted as follows (for example). Let the current location of Jay be $L$. If $L$ were occupied when we insert Jay, then the double hash function of Jay is such that we would consider the locations currently occupied by Jill and Kathy. A “-” specifies that the next location is available (empty).

(a) Where would the Brent algorithm inserts Rudy?

(b) Where would the Gonnet-Munro algorithm inserts Rudy?

In each case mention what other changes are made.

5. (20 points) A \textit{proper} binary tree with $n$ nodes is represented by an array $A$ of size $N = 2^{n+1}$. Prove that

- A is sufficient (you don’t need more space).

- Such a size is necessary (assume $n > 4$).