Data Structures
BITL3

Worksheet 10

Fall 2022

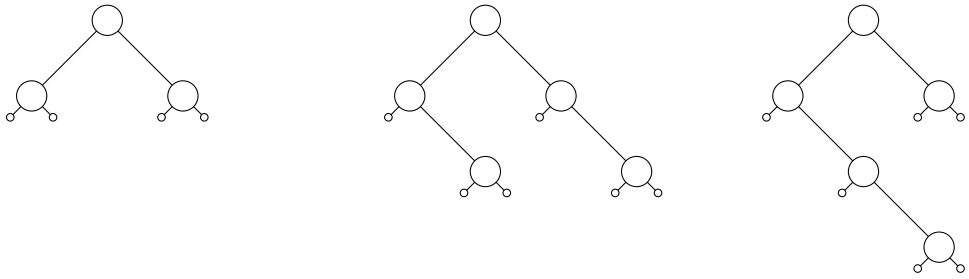
17 November 2022

1. **Warm up:** Answer the following questions.

   (a) What is the difference between hashing by chaining and hashing by probing?

   (b) Why is the size of the "table" (the collection of all possible keys) important to know for hashing?

   (c) How many different ways are there to color the nodes of the following tree, so that it is a red-black tree? The conditions are listed on the next page.



2. This question is about *hash functions*. Let $A = \{\texttt{a}, \texttt{b}, \ldots, \texttt{z}\}$ be the alphabet, and let $h\colon A \to \{0, 1, \ldots, 25\}$ be the funtion that sends each letter to its index position in A. For this question, strings only contain the 26 lowercase letters. Use the file `hash.cpp`.

   (a) For each of the following keys and hashes, find two keys that give the same hash.

      i. keys: all strings S of length 5
      hash: $\sum_{\texttt{s} \in S} h(\texttt{s})$ modulo 10

      ii. keys: all English words W of length 5
      hash: $\sum_{\texttt{w} \in W} h(\texttt{w})$ modulo 10

   (b) For each of the following keys and hashes, find two keys that give a hash of 0.

      i. keys: all strings $S = \texttt{s}_0\texttt{s}_1\texttt{s}_2\texttt{s}_3\texttt{s}_4$ of length 5
      hash: $\sum_{i=0}^{4} 2^i(\texttt{s}_i)$ modulo 128

      ii. keys: all strings S of length 5 containing at least one character `r`
      hash: $\sum_{i=0}^{4} 2^i(\texttt{s}_i)$ modulo 256

   (c) For each of the four key-hash pairs given, are there hash values that are never comuted for any key?

3. This question is about *red-black trees*, and uses a modified `struct` of a `nodeType` used in a previous lab.

```
1   template <class elemType>
2   struct nodeType {
3       elemType info;
4       string color;
5       nodeType<elemType> *leftLink;
6       nodeType<elemType> *rightLink;
7   };
```

Recall the three conditions to be a red-black tree:

- All leaves are black
- Both children of a red node are black
- For every node, any path from down to a leaf must go through the same number of black nodes

(a) Write a function `bool areLeavesBlack(nodeType<int> root)` that, given the root node of a binary tree, returns `true` if the first condition above is satisfied, and `false` otherwise.

(b) Write a function `bool areRedChildrenBlack(nodeType<int> root)` that, given the root node of a binary tree, returns `true` if the second condition above is satisfied, and `false` otherwise.

(c) Write a function `bool arePathsSameBlackLength(nodeType<int> start)` that, given a node in a binary tree, returns `true` if every path from it down to a leaf goes through the same number of black nodes, and `false` otherwise.

Use the file `redblacktree.cpp` for this question.