Ricardo Baeza-Yates & Anurag Bhardwaj December 10, 2018

CS5800 Algorithms – Section 4

Exam Fall 2018 – 2 Hours

1. Mysterious Hashing (25 points)

A smart programmer decided to improve hashing with linear open addressing using the following insertion code:

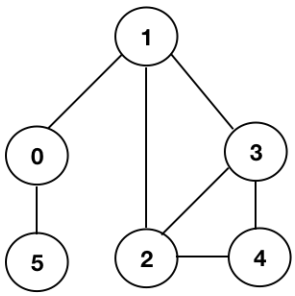
```
Insert( T, m, x)  // Hashing table T[0..m-1] of size m, new element x
int dist, i, j;
{
  j = hash(x); i = j;
  while( T[i] != EMPTY ) {
     dist = abs(j-i) - abs(hash(T[i])-i)
     if( i > j ) dist = abs(hash(T[i])-i) - abs(j-i)
     if( dist < 0 ) {
        temp = T[i]; T[i] = x; x = temp;
     }
     i = i-1; if( i < 0 ) i = m-1;
     if( i == j ) Error( "Full table" );
  }
  T[i] = x;
}
```

What the programmer is improving? Justify.

2. Single points of failure (25 points)

While designing a robust cellular network, Alice was worried that a single point of failure may completely break the network. A **single point of failure** is defined as a node in the network graph, which if removed (along with its edges), will cause the number of connected components in the graph to increase. Let us see this with an example:

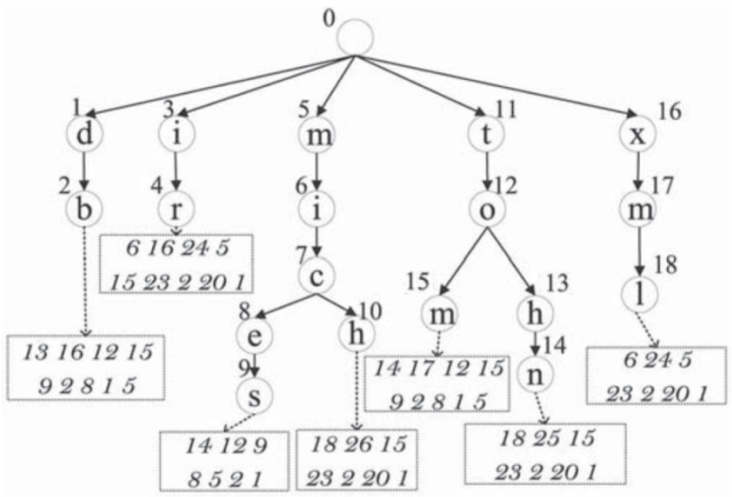
The graph besides is a single connected component. If you remove node 2 and edges (2-1, 2-3, 2-4), the resulting graph will still be a single connected component. However, if you remove node 1 and edges (1-0, 1-2, 1-3), you will have 2 connected components: one with nodes (0,5) and other with nodes (2,3,4). Hence, node 1 is a single point of failure. Write an algorithm to find out **all nodes** that are a single point of failure in a given graph, as there can be more than one. In the example above there are 2: 0 & 1.



3. Searching Palindromes (25 points)

A group of friends, Ada, Anna, Ava, Bob, Elle, Eve, Hannah, and Otto, want to know **how many occurrences of palindromic words of at least two letters** are in a set of documents as well as **how many documents contain palindromes**. They know that the documents are already indexed using an **inverted index** where the vocabulary is stored in a **trie** and each leaf of the trie points to a linked list (inverted list) that contains all documents that contain a word. Every node in the linked list contains a document id and the number of occurrences in that document, that can be retrieved using the fields *docid* and *count*. It also contains the field *next* that gives the next node in the list.

The tree besides shows an example where the letter labels of the branches are in the trie nodes and every rectangle represents a linked list where all the document ids are shown. Notice also that, for layout purposes, the nodes 13 & 15 are in swapped positions. In this example, the words that appear in more documents (9) are “db”, “ir”, and “tom”, while all other words appear in 7 documents. For this example, the solution to the questions above is {0, 0} as there are no palindromes at all.



Write the pseudocode of an algorithm that solves the friends’ questions and indicate the temporal complexity of your solution, considering that the following operations are supported by the index:

- Search(w) returns a pointer to the inverted list of word w (*e.g.*, Search(w)->docid is the docid of the first document in the list that contains w).
- Follows(w) returns the word that follows w in the trie vocabulary (*e.g.*, Follows(“db”) returns “ir”). The word w may not exist and still works.

What you would change to this data structure to make your algorithm faster? Justify if your change implies more space.

4. <u>Vertex Cover</u> (25 points)

Vertex cover is a graph problem defined as follows. Given a graph G(V,E) and a positive integer K, you have to find if there is a subset of vertices V’ of size at most K such that every edge in the graph G is connected to some vertex in V’ (that is, all edges are covered).

Using NP reductions, prove that the Vertex Cover problem is NP Complete. For this problem, you can assume that Clique, 3-CNF-SAT, SAT and CIRCUIT-SAT are given as NP complete problems.

<u>Bonus Question</u> (10 points)

In class we have seen several data structures or techniques that are faster because one of their parameters is over a finite range of values. Name two such examples, indicating which parameter is finite and what improvement is obtained.