UCSC Silicon Valley Extension

C Programming, Advanced

Assignment 2 : Data structures and graphs
Instructor : Radhika Grover

1. Write a program to store a directed or undirected graph using an adjacency matrix.

Huffman code

2. Given a table of frequencies for items, create a Huffman encoding for the items. To create a Huffman tree, the two items with the lowest frequencies are merged as follows: the items form the leaves of the tree and a new parent node is added with a frequency that is the sum of the frequencies of the leaf nodes.

For example, suppose that we have a list with the following letter frequencies:

Items	Frequencies
a	1
b	2
c	3

Then, we remove a and b from the list to create a tree whose root node has the frequency 3 (sum of the frequencies of a and b). We put the root node back in the list and sort the list again by frequency:

Items	Frequencies
*	3
c	3

Now we can merge the root node with c and create a new root node. If there are more elements in the list, this process of creating the tree continues until all the items have been added to it.. To generate a Huffman code for a given item, we traverse the generated tree to the item and output a 0 for a left branch, and a 1 for a right branch.

```
// Pseudocode listing 1
NodeList list;
// Add an item to a list. The second argument to addNode is the frequency of the itemand the third argument is the item.
addNode(list, 1, ‘a');
addNode(list, 2, 'b');
addNode(list, 5, 'c');
```

```
addNode(list, 3, 'd');
HuffmanTree tree;
// create the huffman tree
construct(tree, &list);
// find the huffman codes
huffmanCode(tree, tree.root, 'a');
huffmanCode(tree, tree.root, 'b');
huffmanCode(tree, tree.root, 'c');
huffmanCode(tree, tree, tree.root, ‘d’);

//Outputs may differ from those shown below
a: 100
b: 101
c: 0
d: 11

// Pseudocode listing 2
NodeList list;
addNode(list, 5, 'a');
addNode(list, 9, 'b');
addNode(list, 12, 'c');
addNode(list, 13, 'd');
addNode(list, 16, 'e');
addNode(list, 45, 'f');
HuffmanTree tree;
construct(tree, &list);
huffmanCode(tree, tree.root, 'a');
huffmanCode(tree, tree.root, 'b');
huffmanCode(tree, tree.root, 'c');
huffmanCode(tree, tree.root, 'd');
huffmanCode(tree, tree.root, 'e');
huffmanCode(tree, tree.root, 'f');
//Outputs may differ from those shown below
a: 1100
b: 1101
c: 100
d: 101
e: 111
f: 0


// Pseudocode listing 3
NodeList list;
addNode(list, 50, 'a');
addNode(list, 9, 'b');
addNode(list, 10, 'c');
addNode(list, 10, 'd');
addNode(list, 6, 'e');
addNode(list, 5, 'f');
HuffmanTree tree;
construct(tree, &list);
huffmanCode(tree, tree.root, 'a');
huffmanCode(tree, tree.root, 'b');
huffmanCode(tree, tree.root, 'c');
```

```
huffmanCode(tree, tree.root, 'd');
huffmanCode(tree, tree.root, 'e');
huffmanCode(tree, tree.root, 'f');
//Outputs may differ from those shown below
a: 1
b: 000
c: 001
d: 010
e: 0111
f: 0110

// Pseudocode listing 4
addNode(list, 10, 'a');
addNode(list, 10, 'b');
addNode(list, 10, 'c');
addNode(list, 10, 'd');
HuffmanTree tree;
construct(tree, &list);
huffmanCode(tree, tree.root, 'a');
huffmanCode(tree, tree.root, 'b');
huffmanCode(tree, tree.root, 'c');
huffmanCode(tree, tree.root, ‘d');
//Outputs may differ from those shown below
a: 10
b: 11
c: 00
d: 01
```