The University of Newcastle
School of Information and Physical Sciences

COMP2230/COMP6230 Algorithms
Tutorial Week 8
6^{th} – 10^{th} September 2021

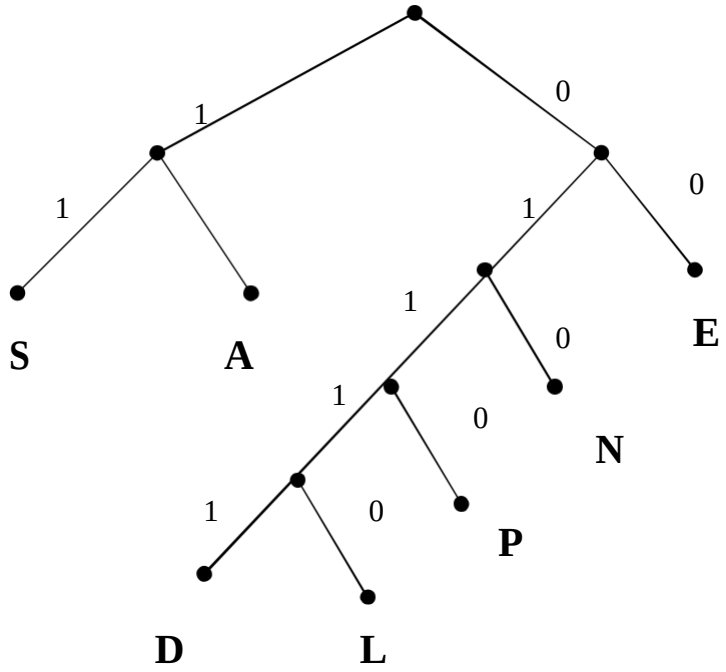
Tutorial

1. Construct Huffman code for the following data and calculate the compression ratio.

character	A	B	C	D	E
probability	0.2	0.3	0.15	0.25	0.1

2. Decode each of the following bit strings using the following Huffman coding tree.

a. 01110100110
b. 01111001001110



3. Consider the coin-changing algorithm and the following denomination: 1, 8, 12. Is the greedy algorithm presented in the lectures optimal for these denominations?

4. Consider a possible divide-and-conquer approach to finding a minimum spanning tree in a connected, weighted graph G. Suppose that we divide the vertices of G into two disjoint subsets V_1 and V_2. We then find a minimum spanning tree T_1 for V_1 and a minimum spanning tree T_2 for V_2. Finally, we find a minimum weight edge e connecting T_1 and T_2. We then construct a minimum spanning tree T of G by taking e and all edges in T_1 and T_2.

   a. Is T always a spanning tree?

   b. If yes, is it always a minimum spanning tree?

5. Let T be a minimum spanning tree for a graph G, let e be an edge in T and let T' be T with e removed. Show that e is a minimum weight edge between components of T'.

6. Trace Kruskal's algorithm for the graph given below.

Initially we have no edges so connected components are {a} {b} {c} {d} {e}
The smallest edge is (b,c) thus we have {a} {b,c} {d} {e}
The smallest edge is (c,e) thus we have {a} {b,c,e} {d}
The smallest edge is (c,d) thus we have {a} {b,c,e,d}
The smallest edge is (a,b) thus we have {a,b,c,e,d}

The edges are: {(b,c), (c,e), (c,d), (a,b)}

7. Trace Prim's algorithm for the same graph (Assume start vertex=a).

Homework

8. Prove that the following Greedy Coin Changing Algorithm is optimal for denominations 1, 5, and 10. (Algorithm 7.1.1 and Theorem 7.1.2 from the text)

   Greedy Coin Changing Algorithm: This algorithm makes change for an amount A using coins of denominations $denom[1] > denom[2] > \cdots > denom[n] = 1$.

```
Input Parameters: denom,A
Output Parameters: None
greedy_coin_change(denom,A) {
    i = 1
    while (A > 0) {
            c = A/denom[i]
            println("use " + c + " coins of denomination " +
                                        denom[i])
            A = A - c * denom[i]
            i = i + 1
    }
}
```

9. (Huffman Codes) Given a text that comprises characters from some n-character alphabet we need to construct a variable-length encoding so as to minimize the total length of the encoded text. This can be done by assigning short codes to frequent characters and longer codes to less frequent characters. Design a greedy algorithm to construct such encoding. Illustrate the algorithm on the following example:

character	a	b	c	d	E
probability	0.35	0.2	0.15	0.2	0.1

What is the compression ratio in this example?

10. What is the time complexity for constructing Huffman code

a. if the frequencies of characters are sorted?
b. if they are not sorted?

Solution: If the characters are unsorted, and kept in a simple array or the like, the algorithm is quadratic; if they are sorted and kept in the same way, then when the new vertices are added back in, we have to re-sort, ultimately ending up with quadratic again. If however as the first step we add everything to a heap, then we reduce the time to T (n log n).

11. Let G be a connected weighted graph and let v be a vertex in G and let e be an edge of minimum weight incident on v. Show that e is contained in some minimum spanning tree.

12. Prove that the Greedy Coin Changing Algorithm from question 8 is optimal for denominations 1, 13, and 25.

13. Trace Kruskal's algorithm for the following graph.

14. Trace Prim's algorithm for the same graph (Assume **a** starting vertex)

More exercises

15. Euro coin denominations are 1,2,5,10,20,50,100 and 200. Tell whether Greedy Coin Changing Algorithm from question 5 is optimal for this system. If it is optimal, give an argument to support your answer. If algorithm is not optimal give a counterexample.

16. Trace Kruskal algorithm for the following figure-3.

17. Trace Prim's algorithm for the same graph.