**CSC 1500 – Exam 3**

**(1)** Create a SINGLE, binary search tree using the following three groups of input. For EACH of the three groups of input you add to the tree, say whether the tree is full, balanced, and/or complete. (*15 pts.*)

**[50,25,30,24,70]**

A black text on a white background

Description automatically generated

**Figure 1-1 Binary Tree With First Set of Input**

**FULL – every node has the maximum or zero number of nodes.**

**BALANCED – difference in height of nodes is not greater than 1 node.**

**COMPLETE – all internal nodes are full, and as left as possible.**

**[72,71,26,32,22]**

A diagram of numbers and a triangle

Description automatically generated

**Figure 1-2 – Binary tree with first and second set of input**

**NOT FULL – SOME INTERNAL NODES DO NOT HAVE MAXIMUM NUMBER OF NODES (70)**

**BALANCED – DIFFERENCE IN HEIGHT BETWEEN NODES DOES NOT EXCEED 1**

**NOT COMPLETE – ALL NODES ARE NOT AS LEFT AS POSSIBLE (70, 72)**

**[56,60,77,54]**

A diagram of numbers and a triangle

Description automatically generated

**Figure 1-3 Binary Tree with first, second, and third set of input.**

**NOT FULL – NOT ALL INTERNAL NODES HAVE MAXIMUM NUMBER OF CHILDREN (24)**

**BALANCED – HEIGHTS DO NOT DIFFER BY MORE THAN 1**

**COMPLETE – ALL LEVELS FULL EXCEPT FOR LOWEST LEVEL WHICH IS SHIFTED TO THE LEFT.**

**(2)** For the following list of letter frequencies, create a Huffman tree, and use it to determine the encoding for each of the letters. After you’ve written down the encoding for each letter, determine the average number of bits needed to encode ANY letter using this encoding. (*15 pts.*)

**A: .33 B: .10 C: .08 D: .12 E: .37**

**IN ORDER FOR LAZINESS:**

**E: .37, A: .33, D: .12, B: .10 C: .08**

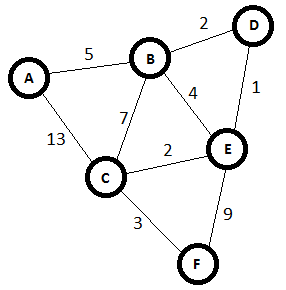
**A paper with text and numbers

Description automatically generated**

**Figure 2 -1 – World’s most colorful Huffman tree.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **LETTER** | **REPRESENTATION** | **BITS** | **FREQUENCY** | **BITS\*FREQ** |
| **A** | **11** | **2** | **.33** | **.66** |
| **B** | **1011** | **4** | **.1** | **.4** |
| **C** | **1010** | **4** | **.08** | **.32** |
| **D** | **100** | **3** | **.12** | **.36** |
| **E** | **0** | **1** | **.37** | **.37** |
|  |  |  | **AVG:** | **2.11** |

**(3)** Using the following graph, draw the Minimum Spanning Tree, **AND** by using Dijkstra’s Algorithm give the shortest path from A to F (do not confuse F with E). You may either draw the shortest path, or write out the steps needed to follow the route you find. (*15 pts. each)*



A diagram of a triangle with circles and lines

Description automatically generated

Figure 3-1 – The World’s Jankiest Looking Minimum Spanning Tree

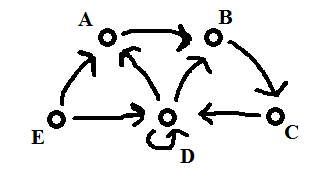
A diagram of a tree with numbers and letters

Description automatically generated

Figure 3-2- Djikstra strikes again.

The shorted path is F-C-E-D-B-A at 13 long.

**(4)** For the following Directed Graph, create an adjacency matrix. Then, via matrix multiplication, show me the CUBED Matrix (M x M x M). (*20 pts.*)



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E |
| A | 0 | 0 | 0 | 1 | 1 |
| B | 1 | 0 | 0 | 1 | 0 |
| C | 0 | 1 | 0 | 0 | 0 |
| D | 0 | 0 | 1 | 1 | 1 |
| E | 0 | 0 | 0 | 0 | 0 |

Table 1 – Adjacency Matrix.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| SQUARED | A | B | C | D | E |
| A | 0 | 0 | 1 | 1 | 1 |
| B | 0 | 0 | 1 | 2 | 2 |
| C | 1 | 0 | 0 | 1 | 0 |
| D | 0 | 1 | 1 | 1 | 1 |
| E | 0 | 0 | 0 | 0 | 0 |

Table 2 – Adjacency Matrix Squared

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CUBED | A | B | C | D | E |
| A | 0 | 1 | 1 | 1 | 1 |
| B | 0 | 1 | 2 | 2 | 2 |
| C | 0 | 0 | 1 | 2 | 2 |
| D | 1 | 1 | 1 | 2 | 1 |
| E | 0 | 0 | 0 | 0 | 0 |

Table 3 – Adjacency Matrix Cubed

**(5)** You’ve got two lottery tickets available to you. One of them offers 30 different REPEATING numbers for you to pick FOUR from. The other offers 34 different NON-REPEATING numbers for you to pick FIVE from. Calculate the odds for winning on each of these lottery tickets (assume there’s only ONE winning combination for each of them), write them down, then pick the one which gives you a better chance of winning. (*20 pts.)*

REPEATING LOTTERY TICKET

30 NUMBERS = n

4 SELECTED = r

COMBINATION WITH REPETITION: (n+r-1)!/ ( r! \* (n-1)!)

= (30+4-1) / (4! \* (30-1)!) )

=(33!)/(24 \* (29!)

= 40920

= 1/ 40920 odds of drawing matching numbers

NON-REPEATING LOTTERY TICKET

34 NUMBERS = n

5 SELECTED = r

COMBINATION WITHOUT REPETITION: (n!) / (r! \* (n-r)!)

= (34!) / (4!\*(34-5)!

=(34!) / (24\* (29!))

= 1391280

1/1391280 odds of drawing matching numbers

The combination with repetition gives better odds.