EXERCISE REPORT

TEAM 15

GREEDY

Lecturer: Son Nguyen Thanh MCS

Class: CS112.N21.KHTN

Member 1: Quan Vo Minh – 21520093

Member 2: Nhat Nguyen Viet – 21520378

Excercise: Team 3's Excercise

Date: 21/05/2023

EXERCISE 1: Design and conduct an experiment to epirically compare the efficiencies of Prim's and Kruskal's algorithm on random graphs of different sizes and densities

We generated 2 tests:

- Dense connected graph: |V| = 10000, |E| = 5001831
- Sparse connected graph: |V| = 100000, |E| = 1001170

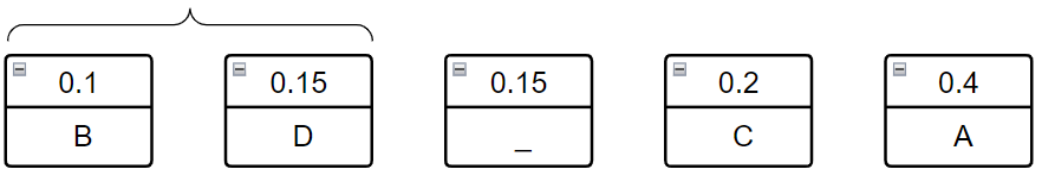
```
Sparse connected graph:
Result(Kruskal's algorithm): 60261505
Kruskal's algorithm run time: 311021.00 microseconds

Result(Prim's algorithm with adjacency list): 60261505
Prim's algorithm run time: 469296.50 microseconds

Dense connected graph:
Result(Kruskal's algorithm): 7684
Kruskal's algorithm run time: 1584080.70 microseconds

Result(Prim's algorithm with adjacency list): 7684
Prim's algorithm run time: 744903.50 microseconds


Process returned 0 (0x0)   execution time : 5.754 s
Press any key to continue.
```

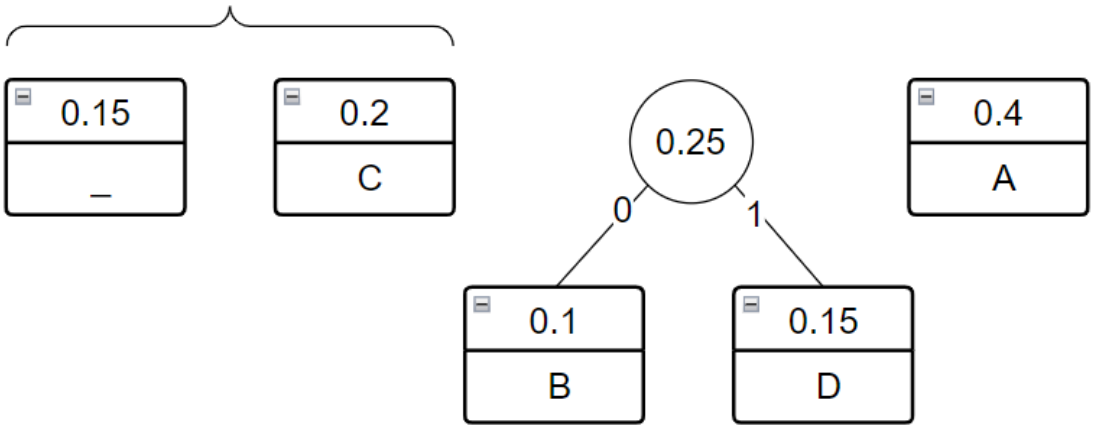
The experiment showed that Kruskal's algorithm is better on sparse graph but Prim's algorithm is better on dense graph.

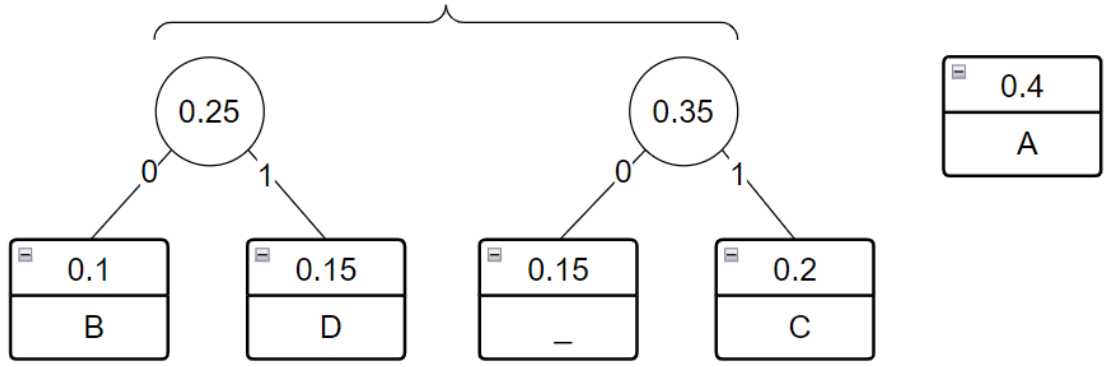
EXERCISE 2:

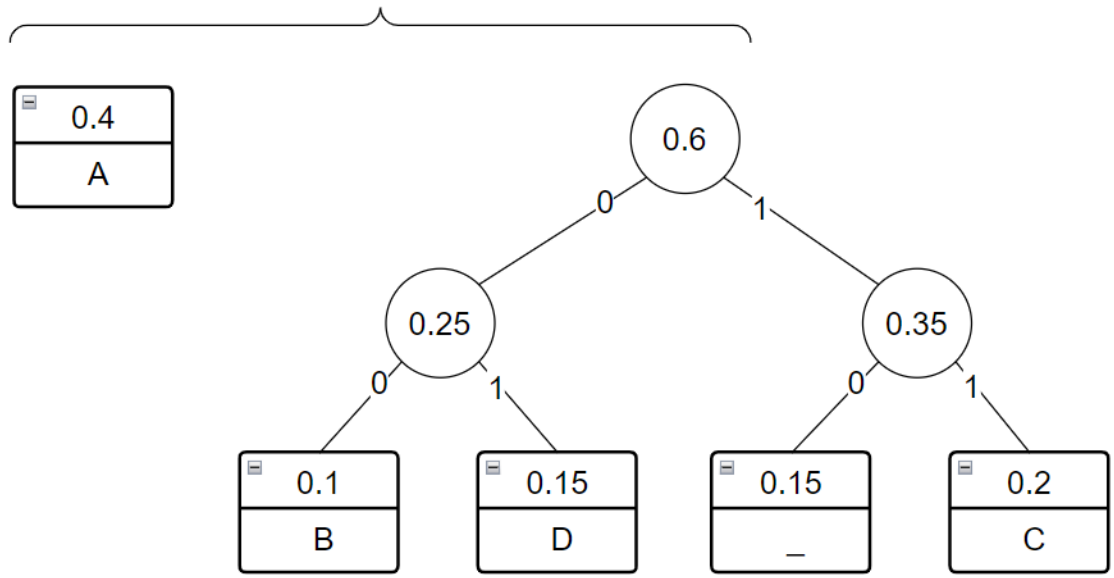
a. Construct a Huffman code for the following data

Construction steps:

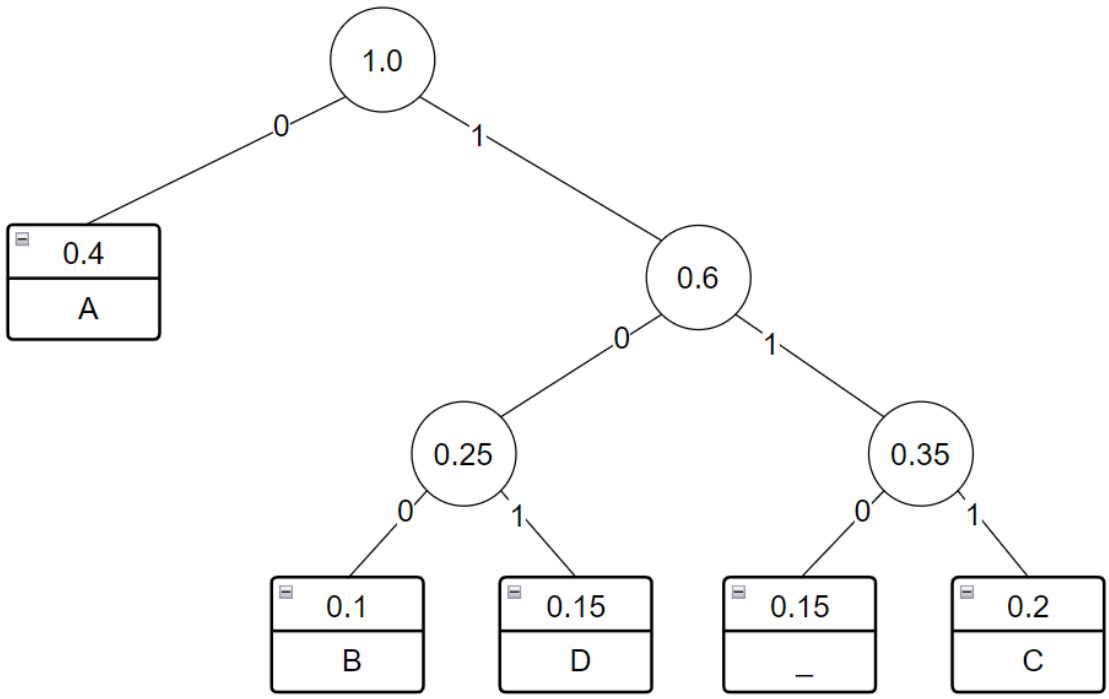








Final Huffman code:



b. Encode ABACABAD using the code of question (a).

Huffman code of following characters are:

“A” = 0

“B” = 100

“D” = 101

“_” = 110

“C” = 111

We can encode the given string by replacing each character with its Huffman code:

“ABACABAD” = 0100011101000101

c. Decode 100010111001010 using the code of question (a).

To decode the encoded data we require the Huffman tree. We iterate through the binary encoded data. To find character corresponding to current bits, we use the following simple steps:

- We start from the root and do the following until a leaf is found.
- If the current bit is 0, we move to the left node of the tree.
- If the bit is 1, we move to right node of the tree.
- If during the traversal, we encounter a leaf node, we print the character of that particular leaf node and then again continue the iteration of the encoded data starting from step 1.

Decoded message: BAD_ADA

Implementation

Result of implementation:

```
A 0
B 100
D 101
_ 110
C 111
Encoded string of 'ABACABAD' is: 0100011101000101
Decoded string of '100010111001010' is: BAD_ADA

Process returned 0 (0x0)   execution time : 0.017 s
Press any key to continue.
```