CMPSC 465 Assignment - Greedy Method

Name: Anthony Vallin PSU Username: aav5195

Format Requirement

- Algorithms in pseudo code **MUST** be placed in code blocks/fences (5%), and use either `cpp` or `java` as the syntax highlight.
- Algorithms should follow the pseudo code standard described in handout 1. (2%)
- Do NOT change the template except the answer portion. (5%)
- Formulas and equations should be in math mode using Latex math symbols. (5%)
  - Markdown math tutorial: http://csrgxtu.github.io/2015/03/20/Writing-Mathematic-Fomulars-in-Markdown/
  - Two ways to enter math mode:
    - Insert a pair of dollar signs: $ your equations go here $. This is the inline math mode.
    - Insert a pair of double-dollar signs: $$ your equations go here $$, which produces a standalone equation/formula set.

Problem Set

Problem 1

Does Prim’s algorithm always work correctly on graphs with **negative edge weights**? Justify your answer

Answer:

Prim's algorithm works correctly on graphs with negative edge weights. This is because the algorithm uses a single minimum weight edge.

Problem 2

How can one use Prim’s algorithm to find a spanning tree of a connected graph **with no weights** on its edges? Is it a good algorithm for this problem?

Answer:

An arbitrary weight constant would need to be applied to all the edges in order to use Prim's algorithm. Prim's algorithm is not a good choice for this problem. For example, a sparse graph would be slow using Prim's algorithm with an arbitrary weight.

Problem 3

Outline an efficient algorithm in **pseudo code** for changing an element's value in a max-heap. What is the time efficiency of your algorithm?

Answer:

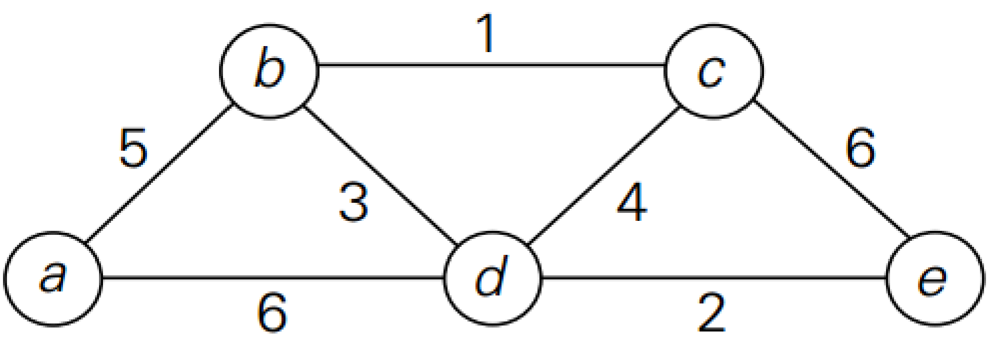
```
MaxHeapChange(H, k)
//Changes an element's value in a max-heap.
//Input: Max-heap and new k element value.
//Output: Modified Max-Heap with element value changed to k

H.heapSize <- H.heapSize + 1
H[H.heapSize] <- k
i <- H.heapSize
while i < 1 && H[i] > H[i/2] do
    swap(H[i], H[i/2])
    i <- i/2
MaxHeapify(i)
```

Time efficiency: $\theta(logn)$

Problem 4

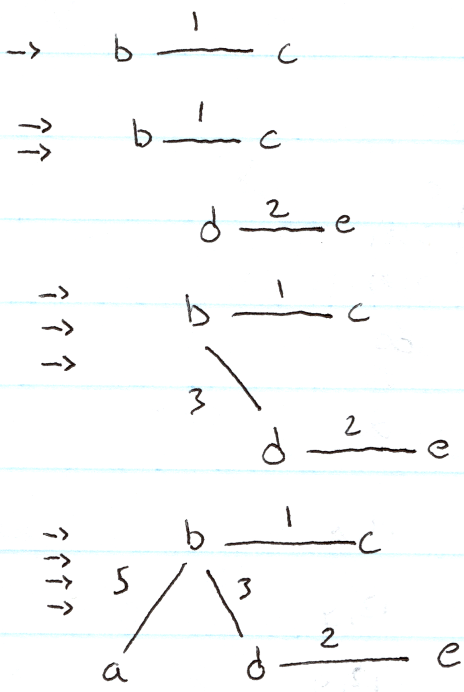
Apply Kruskal's algorithm to find a minimum spanning tree of the following graphs. You need to do it step by step.



Answer:

4

bc	de	bd	cd	ab	ad	ce
1	2	3	4	5	6	6



Problem 5

Does Kruskal’s algorithm work correctly on graphs that have negative edge weights? Justify your answer.

Answer:

Kruskal's algorithm works correctly on graphs that have negative edge weights. The reason is similar to Primm's algorithm. The algorithm uses a single edge weight to determine the next step.

Problem 6

Indicate whether the following statements are true or false:

1. If e is a minimum-weight edge in a connected weighted graph, it must be among edges of at least one minimum spanning tree of the graph.
2. If e is a minimum-weight edge in a connected weighted graph, it must be among edges of each minimum spanning tree of the graph.
3. If edge weights of a connected weighted graph are all distinct, the graph must have exactly one minimum spanning tree.
4. If edge weights of a connected weighted graph are not all distinct, the graph must have more than one minimum spanning tree.

Answer:

1. true
2. false
3. true
4. false

Problem 7

Construct a Huffman code for the following data:

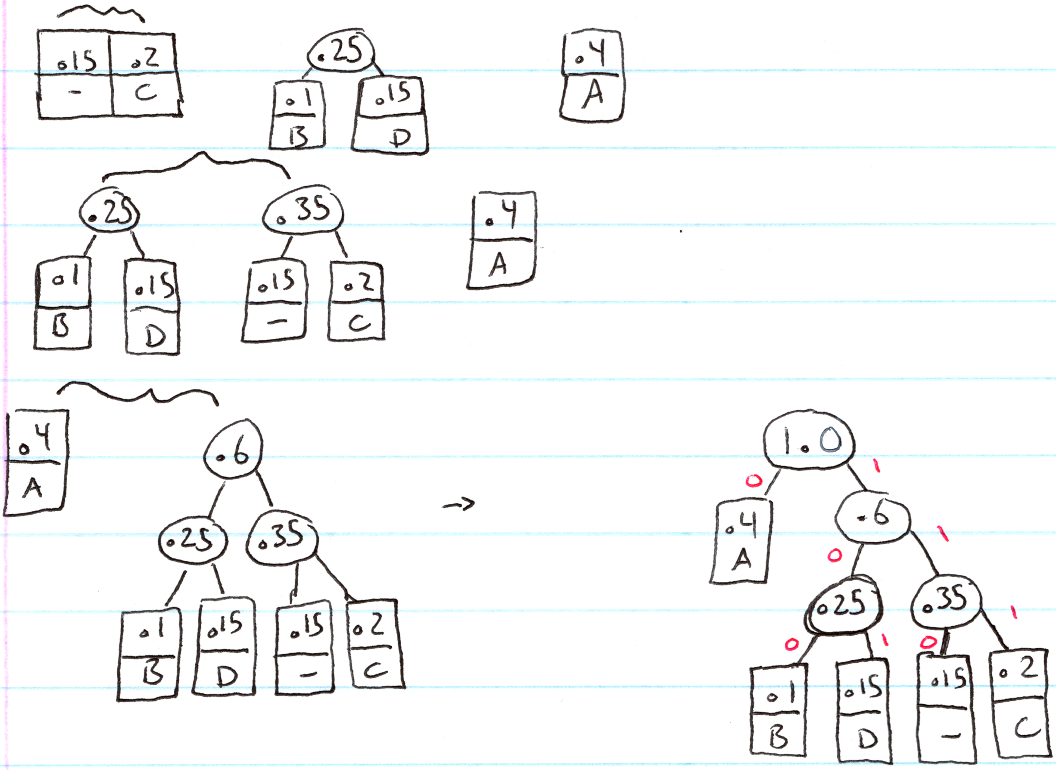
symbol	A	B	C	D	_
frequency	0.4	0.1	0.2	0.15	0.15

- Encode `ABACABAD` using the code.
- Decode 100010111001010 using the code.

Answer:

(7) B D - C A

.1 .15 .15 .2 .4



ENCODE: ABACABAD → 0 100 0 111 0 100 0 101

DECODE: ~~1000 10 111 001010~~

BAD-ADA