CS 61B Fall 2023

Graphs II, Tries

Exam-Level 11: October 30, 2023

1 Multiple MSTs

Recall a graph can have multiple MSTs if there are multiple spanning trees of minimum weight.

(a) For each subpart below, select the correct option and justify your answer. If you select "never" or "always," provide a short explanation. If you select "sometimes", provide two graphs that fulfill the given properties — one with multiple MSTs and one without. Assume G is an undirected, connected graph with at least 3 vertices.

1. If **some** of the edge weights are **identical**, there will

   ◯ never be multiple MSTs in G.

   ◯ sometimes be multiple MSTs in G.

   ◯ always be multiple MSTs in G.

   Justification:

2. If **all** of the edge weights are **identical**, there will

   ◯ never be multiple MSTs in G.

   ◯ sometimes be multiple MSTs in G.

   ◯ always be multiple MSTs in G.

   Justification:

(b) Suppose we have a connected, undirected graph G with N vertices and N edges, where all the **edge weights are identical**. Find the maximum and minimum number of MSTs in G and explain your reasoning.

Minimum: _____
Maximum: _____

Justification:

(c) It is possible that Prim's and Kruskal's find **different** MSTs on the same graph `G` (as an added exercise, construct a graph where this is the case!). Given any graph `G` with integer edge weights, modify G to **ensure** that Prim's and Kruskal's will always find the same MST. You may not modify Prim's or Kruskal's, and you may not add or remove any nodes/edges.
Hint: Look at subpart 1 of part a.

2 Topological Sorting for Cats

The big brain cat, Duncan, is currently studying topological sorts! However, he has a variety of **curiosities** that he wishes to satisfy.

(a) Describe at a high level in plain English how to perform a topological sort using an algorithm we already know (hint: it involves DFS), and provide the time complexity.

(b) Duncan came up with another way to possibly do topological sorts, and he wants you to check him on its correctness and tell him if it is more efficient than our current way! Let's derive the algorithm.

1. First, provide a logical reasoning for the following claim (or a proof!): Every DAG has at least one source node, and at least one sink node.

2. Next, describe an algorithm (in English or in pseudocode) for finding all of the source nodes in a graph.

3. Now, make the following observation: If we remove all of the source nodes from a DAG, we are guaranteed to have at least one new source node. Inspired by this fact, and using the previous parts, come up with an algorithm to topological sort. Describe it in words or using pseudocode. Is it more efficient than what we already have? *Hint: If it's easier for you, first consider one with quadratic runtime, then think about how you might save some computations to make it faster.*

3 A Wordsearch

Given an N by N wordsearch and N words, devise an algorithm (using pseudocode or describe it in plain English) to solve the wordsearch in $O(N^3)$. For simplicity, assume no word is contained within another, i.e. if the word "bear" is given, "be" wouldn't also be given.

If you are unfamiliar with wordsearches or want to gain some wordsearch solving intuition, see below for an example wordsearch. Note that the below wordsearch doesn't follow the precise specification of an N by N wordsearch with N words, but your algorithm should work on this wordsearch regardless.

Example Wordsearch:

C	M	U	H	O	S	A	E	D
T	R	A	T	H	A	N	K	A
O	C	Y	E	S	R	T	U	T
N	I	R	S	A	I	O	L	S
Y	R	R	M	T	N	N	H	R
Y	E	A	E	V	A	R	U	E
A	A	A	I	M	E	L	C	R
N	H	D	J	Y	U	A	C	I
T	Y	S	A	A	R	S	U	C
A	R	S	I	G	Y	E	S	A

ajay
crystal
grace
luke
rica
sherry
sohum
tony
anton
eric
isha
naama
sarina
shreyas
sumer
vidya

Hint: Add the words to a Trie, and you may find the `longestPrefixOf` operation helpful. Recall that `longestPrefixOf` accepts a `String key` and returns the longest prefix of `key` that exists in the `Trie`, or **`null`** if no prefix exists.