University of Waterloo
CS341 - Fall 2024
Assignment 3

Due Date: Tuesday Nov 12, 11:59pm

Please read the following link for guidelines on submission and late policy:

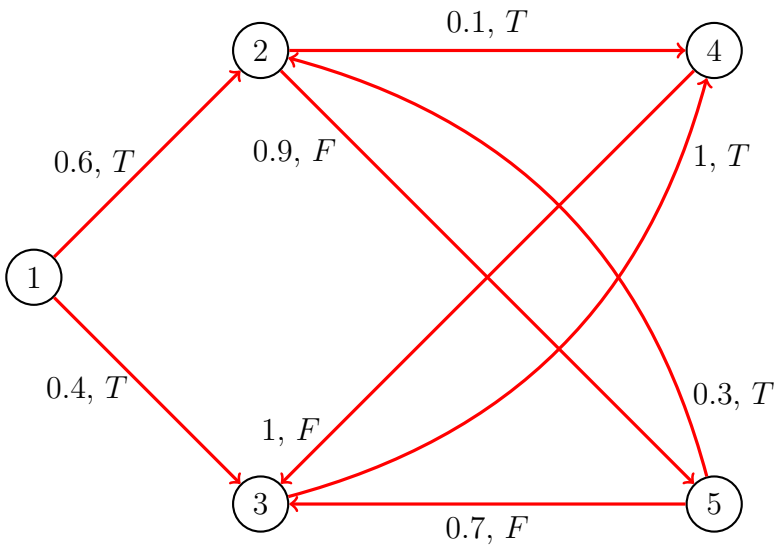
`https://student.cs.uwaterloo.ca/~cs341`

Question 1 Written question + programming: likeliest walk (10 + 10 marks)

We consider a directed graph $G = (V, E)$, where edges e are labelled by both a *probability* $p[e]$ and a character $c[e]$. For any vertex v, the sum of all $p[e]$, for $e = (v, w)$, if equal to 1.

We can build a string S by starting from a given source s, and repeating N times the following: from the current vertex v, choose an edge $e = (v, w)$ at random (using the probability distribution given by the probabilities $p[e]$, for e outgoing edge at v), append its character $c[e]$ to S, and repeat from w.

The goal here is to somehow reverse the process: you are given the string S, and your task is to determine what was the likeliest walk we followed to produce this string: you should return the probability of this walk (which is the product of the probabilities of its edges) and the walk itself. For example, consider the following graph, with edges labelled with their probabilities and characters T or F.



Starting from $s = 1$, there are two ways we can obtain the string $S = TT$, through the walks 124 and 134. The former has probability $0.6 \times 0.1 = 0.06$, the latter $0.4 \times 1 = 0.4$. So in this case, the output of the algorithm should be the walk 134, with its probability 0.4.

1. Describe an algorithm that performs this task and analyze its runtime in terms of n (number of vertices), m (number of edges) and L (length of the string S). You can assume that all operations you do use arguments that fit in a word (this includes floating-point arithmetic, to manipulate the probabilities). You can also assume that the vertex set is $V = \{1, \ldots, n\}$, with source $s = 1$.

2. Implement your algorithm. The input format is:

   - first line: n
   - then, for $i = 1, \ldots, n$, two lines, consisting of
     - the out-degree d_i of vertex i (number of outgoing edges)
     - for $j = 1, \ldots, d_i$, corresponding to an edge $e_{i,j} = (i, v_{i,j})$, three tokens that are: the index $v_{i,j}$ of the destination vertex, the probability $p[e_{i,j}]$ and the character $c[e_{i,j}]$ (this gives a total of $3d_i$ tokens on this line). Tokens are separated by white spaces. If $d_i = 0$, the second line is empty.
   - the string S

   For example, the previous graph and string $S = TT$ could be encoded by the following:

```
5
2
2 0.6 T 3 0.4 T
2
4 0.1 T 5 0.9 F
1
4 1 T
1
3 1 F
2
2 0.3 T 3 0.7 F
TT
```

   The output consists of two lines: on the first one, the probability of the likeliest path that produces S (we assume the path starts at vertex 1), on the second one, the path itself, as a sequence of integers $1 \ldots$ separated by white spaces (leave no trailing white space). If you do not find a path, the probability should be 0, and the second line should be empty. On the example above, the output should be

```
0.4
1 3 4
```

   If we were considering the string $S = F$ instead, we would get

0

(with one empty line after the 0) If you are using C++, use doubles for your floating point operations.

Question 2 Counting walks in a graph (12 marks)

1. Given a directed graph $G = (V, E)$, with $|V| = n$ and $|E| = m$, and two vertices a, b in G, give an algorithm that returns the number of **shortest** walks from a to b in G. Give its runtime in terms of n and m.

2. With the same input as before, give an algorithm that decides if there are infinitely many walks from a to b.

3. Still with the same input as before, assuming that there are finitely many walks from a to b, give an algorithm that returns their number.

Question 3 Updating a minimum spanning tree (10 marks)

You are given a weighted connected graph $G = (V, E)$, with weight function w, n vertices and m edges, and a minimum spanning tree T for G. Now, suppose we suddenly update the weight of an edge e in T, from $w(e)$ to a new value $w'(e)$. Give and analyze an algorithm that computes a minimum spanning tree for the graph with the new weights. For full credit, your algorithm should run in time $O(m)$.

You can assume that all weights involved in this question are distinct. Vertices are indexed $1, \ldots, n$, and you can assume that T is given to you as a rooted tree with root of index 1, by means of an array of parents ($\mathsf{parent}[i]$ is the index of the parent of vertex i, with $\mathsf{parent}[1] = 0$). The output should also be a tree rooted at 1, given by means of an array of parents.

Question 4 Shortest path with small integer distances (10 marks)

Consider a directed graph $G = (V, E)$ with n vertices and m edges, where each edge e has a positive integer weight. Given G and a source s, give a linear time algorithm that computes the distances from the source s to all vertices v, assuming that all these distances are at most equal to m.