CSCI 3104 Spring 2018

Problem Set 8

Merola, Michael

06/04/1998

Problem Set 8

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pylab as plt
%matplotlib inline
```

Problem 1

(10 pts) Ginerva Weasley is playing with the network given below. Help her calculate the number of paths from node 1 to node 14.

Hint: assume a “path” must have at least one edge in it to be well defined, and use dynamic programming to fill in a table that counts number of paths from each node j to 14, starting from 14 down to 1.

I solved this problem by hand using a recurrence starting from node 6:

a1 : $6 \rightarrow 14 = 2paths$

a2 : $5 \rightarrow 14 = a1 + 5 = 7$ ...paths from the last node plus new paths

a3 : $4 \rightarrow 14 = a2 + 7 = 14$

a4 : $3 \rightarrow 14 = a3 + 4 = 18$

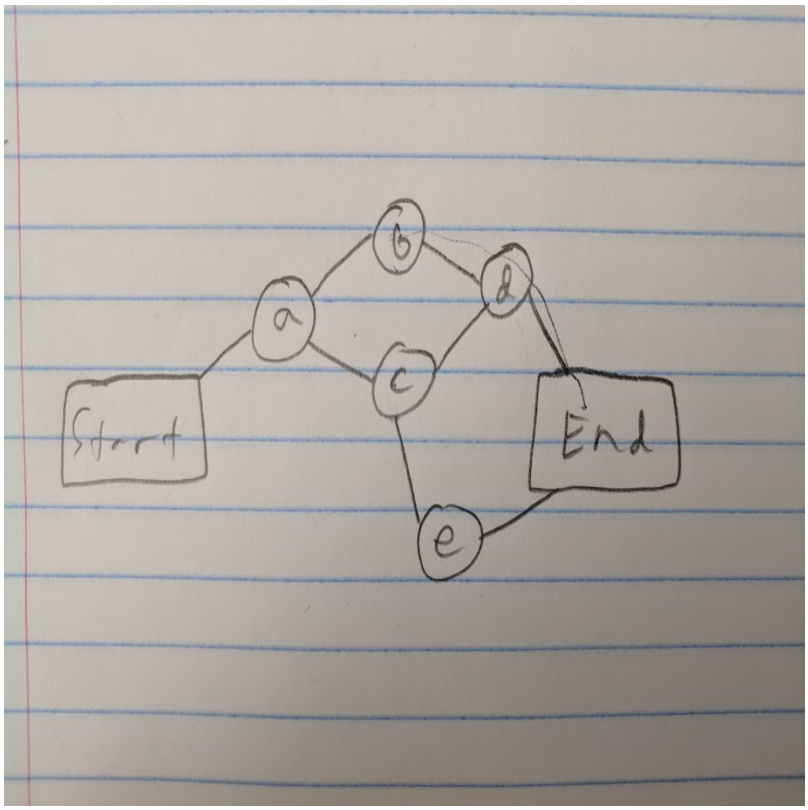
a5 : $2 \rightarrow 14 = a4 + 8 = 26$

a6 : $1 \rightarrow 14 = a5 + 4 = 30$

so there are **30** paths from node 1 to node 14

Problem 2

(10 pts) Ginny Weasley needs your help with her wizardly homework. She's trying to come up with an example of a directed graph $G = (V, E)$, a start vertex $v \in V$ and a set of tree edges $E_T \subseteq E$ such that for each vertex $v \in V$, the unique path in the graph (V, E_T) from s to v is a shortest path in G, yet the set of edges E_T cannot be produced by running a depth-first search on G, no matter how the vertices are ordered in each adjacency list. Include an explanation of why your example satisfies the requirements.



When using DFS to produce a set of edges for this graph, the algorithm would find the path 'START,a,b,d,END' and mark those nodes as 'visited.' On subsequent runs, the algorithm would get to node c and see that node d was visited; then continue to node e without tracking the edge between c and d.

Problem 3

(15 pts) Prof. Dumbledore needs your help to compute the in- and out-degrees of all vertices in a directed multigraph G. However, he is not sure how to represent the graph so that the calculation is most efficient. For each of the three possible representations, express your answers in asymptotic notation (the only notation Dumbledore understands), in terms of V and E, and justify your claim.

(a) An adjacency matrix representation. Assume the size of the matrix is known.

For an adjacency matrix representation, the graph should be represented by a V x V matrix where you can represent an edge between vertices with either a 1 or 0.

In this case, the representation would be $O(V^2)$

(b) An edge list representation. Assume vertices have arbitrary labels.

For an edge list representation, the graph is represented by a list of tuples in the form $[(V_1, V_2), (V_2, V_3), \dots]$ where each tuple marks an edge between two vertices.

A search tree is made for each edge to represent the vertices that are connected to it.

In this case, the representation would be $O(Elog(V))$

(c) An adjacency list representation. Assume the vector's length is known.

For an adjacency list representation, the graph is represented by a list where each V in the list has an attached linked list that represents which other vertices are connected to it by an edge.

A search algo would have to run through V vertices and E edges in two distinct arrays.

In this case, the representation would be $O(V + E)$

Problem 4

(30 pts) Deep in the heart of the Hogwarts School of Witchcraft and Wizardry, there lies a magical grey parrot that demands that any challenger efficiently convert directed multigraphs into directed simple graphs. If the wizard can correctly solve a series of arbitrary instances of this problem, the parrot will unlock a secret passageway.

Let $G = (E, V)$ denote a directed multigraph. A directed simple graph is a $G' = (V, E')$, such that E' is derived from the edges in E so that (i) every directed multiedge, e.g., (u, v), (u, v) or even simply (u, v), has been replaced by a single directed edge (u, v) and (ii) all self-loops (u, u) have been removed.

Describe and analyze an algorithm (explain how it works, give pseudocode if necessary, derive its running time and space usage, and prove its correctness) that takes $O(V + E)$ time and space to convert G into G', and thereby will solve any of the Sphinx's questions. Assume both G and G' are stored as adjacency lists.

Describe Algorithm:

1) Loop thru the length 1st dimension of the adj list by *i*, where variable *head* represents the adjList[i]

2) For each *head* in the list, traverse thru the linked list of edges where *value* represents the current edge.

3) if the *head* does not equal *value* and *value* has not been seen yet,

```
Mark the value as visited
Add the value to the appropriate position in a new adjacency list
```

4) repeat the process until every *head* has been evaluated and the new adjList has been built

Running Time

Since we run thru V vertices and evaluate E edges during the algorithm, its running time would be $O(V + E)$

The Space Complexity is also $O(V + E)$ because it accounts for the size of the new adjacency list created.

Correctness

Because this algorithm is merely transposing the original correct graph into a more condensed graph, the correctness for the algorithm relies on the code that eliminates self-loops and repeats from the original graph.

In step 3 of my algo, only edges that are not equal to the head and have not been repeated can be added to the new adjacency list. Because these are the only operations performed to change the graph, the new graph retains its validity as a structure and therefore my algorithm is correct.

Collaborators

Krish Dholakiya

Gustav Solis

Eric Weng
