**CS 430 – FALL 2017**

**INTRODUCTION TO ALGORITHMS**

**HOMEWORK #7 KEY**

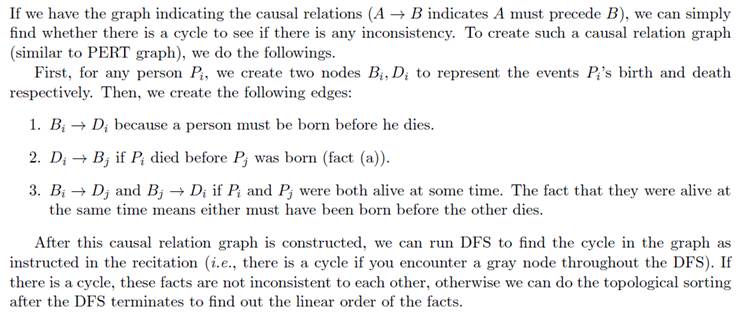
**DUE Fri Dec 1, 11:25am**

1. (3 points) You are helping a group of ethnographers analyze some oral history data they have collected by interviewing members of a village to learn about the lives of people who lived there over the last two hundred years. From the interviews, you have learned about a set of people, all now deceased, whom we will denote P1, P2, P3,…, Pn. The ethnographers have collected several facts about the lifespans of these people:

a)     Pi died before Pj was born

b)     Pi and Pj were both alive at the same time

However, the ethnographers are not sure that there facts are correct. So they’d like you to determine whether the data they have collected is at least internally consistent, in the sense that there could have existed a set of people for which all the facts simultaneously hold. Describe an algorithm to answer the ethnographers’ problem. Your algorithm should output whether all the facts are consistent, or output the subset of facts that may be inconsistent. Hint: construct a bipartite directed graph in which there are 2n nodes representing, respectively, dates of birth and death of the n people. Draw edges from one node to another when we have a fact that states that the former precedes the latter. Clearly a person’s birth precedes his/her death, so that would be an edge. What does it mean if two people are claimed to have been alive at the same time?

****

2. (3 points) The **clustering coefficient** *C(G)* of a simple graph *G* is the probability that if *u* and *v* are neighbors and *v* and *w* are neighbors, then *u* and *w* are neighbors, where *u*, *v*, and *w* are distinct vertices of *G*.

7a. We say that three vertices *u*, *v*, and *w* of a simple graph *G* form a triangle if there are edges connecting all three pairs of these vertices. Find a formula for *C(G)* in terms of the number of triangles in *G* and the number of paths of length two in the graph. [*Hint:* Count each triangle in the graph once for each order of three vertices that form it.]



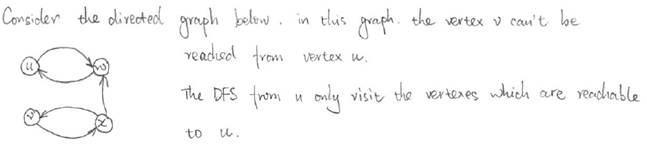
7b. Explain what the clustering coefficient measures in each of these graphs.

- the Hollywood graph (used for the six degrees of Kevin Bacon problem)   **probability that IF actor A in movie with actor B, and actor B in movie with actor C, THEN actor A also in a movie with actor C**

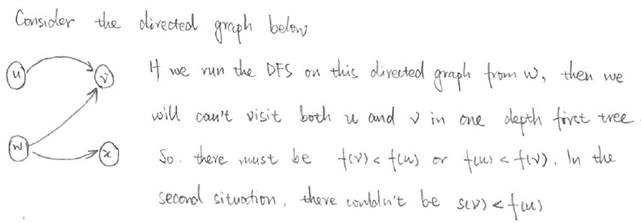
**-** the graph of Facebook friends **probability that IF person A is friend with person B, and person B is friend with person C, THEN person A also is friend with person C**

- the graph representing the routers and communications links that make up the worldwide Internet **probability that IF router A is connected with router B, and router B is connected with router C, THEN router A also is connected with router C**

3. (2 points)  3a. Show how on a directed graph that depth first search starting at vertex u can result in vertex v not being reachable from u even though both u and v have both incoming and outgoing edges.



3b. If a directed graph contains a path from u to v, show that it is not necessary that the s(v) < f(u). s() is the depth first search start time and f() is the depth first search finish time



4. (3 points) Bob loves foreign languages and wants to plan his course schedule to take the following nine language courses: LA15, LA16, LA22, LA31, LA32, LA126, LA127, LA141, and LA169. The course prerequisites are:

LA15: (none)                                                           LA16: LAl5 is prerequisite

LA22: (none)                                                           LA31: LAl5 is prerequisite

LA32: LA16 and LA31 are prerequisites                         LA126: LA22 and LA32 are prerequisites

LA127: LAl6 is prerequisite                                 LA141: LA22 and LAl6 are prerequisites

LA169: LA32 is prerequisite

Find a sequence of courses that allows Bob to satisfy all the prerequisites.

**The topological sorting algorithm can help us solve this problem.**

* **Build a digraph to represent the course prerequisite requirements. The nine courses are vertices in the digraph. If a course A is a prerequisite for another course B, the digraph has a directed edge from A to B.**
* **Apply the topological sorting algorithm on this digraph. The result is one possible sequence of courses for Bob.**

**For the given set of prerequisites, the solution is not unique. For example, one possible solution is LA15, LA16, LA22, LA31, LA32, LA126, LA127, LA141, LA169. Another solution is LA15, LA16, LA127, LA31, LA32, LA169, LA22, LA126, LA141.**

5. (3 points) Prim’s and Kruskal’s algorithms both “grow” a minimum spanning tree of a graph by selecting edges to add to the tree in a specified, greedy order. Design an efficient algorithm to “prune” a graph and yield a minimum spanning tree by removing edges from the graph in a specified, greedy order. Prove optimal substructure and the Greedy Choice Property.

**Optimal Substructure - Same as proof of Optimal Substruture for Minimum Spanniung Tree that we did in class. The algorithm to solve the problem has nothing to do with proving Optimal Substructure. Proving Optimal Substructure only considers the problem, find a minimum spanning tree.**

**Draw any tree T with edge (u,v) in the middle.  Let T be the MST of graph G (other edges of G not shown). Removing edge (u,v) must partition tree T into 2 trees T1 and T2.**

**Claim: T1 is the MST of G1=(V1, E1) , the subgraph of G using the vertices of T1**

**T2 is the MST of G2=(V2, E2) , the subgraph of G using the vertices of T2**

**Proof: weight(T) = weight(u,v) + weight(T1) + weight(T2) is valid (edge weights are additive)**

**Therefore that can't be better subtrees than T1 and T2 or else T would not have been optimal.**

**Greedy Choice Property - We want to prove that the greedy choice of removing the maximum weight edge from the graph (and maintaining the spanning and tree properties) will yield a MST.  Another way of saying this is that a MST minimizes the weight of the maximum-weight edge.**

**ssume we have an optimal solution (we have a MST) T for graph G. We don't care how we got this optimal solution. Some else tells us that T' is a better (more minimal) MST of G. Let e be the maximum weight edge in T and e' be the maximum weight edge in T'. Since someone told us that T' was better than T, then**

**weight(e') < weight(e) . Removing e from T breaks T into 2 components (subtrees). Some edge e'' in T' must connect these 2 components or else T' would be disconnected. Replacing e in T with e'' forms a new spanning tree T'' = T - e + e''**

**weight(e'') <= weight(e') because e' was the maximum weight edge in T' and hence**

**weight(e'') <= weight(e') < weight(e)**

**Thus T'' has a lower weight than T:   weight(T'') = weight(T)-weight(e)+weight(e'') < weight(T)**

**But there cannot be a spanning tree T'' with a lower weight than T because T was assumed to be a MST, therefore there is no better (smaller) choice for edge e. edge e is the minimum maximum weight edge. Thus any MST minimizes the weight of the maximum weight edge.**

6. (3 points) Shortest Path Verification - Your roommate has written a program to implement Dijkstra’s shortest path algorithm. Design and analyze a linear time algorithm to check your roommate’s algorithm’s results. That is, given a graph G = (V,E), a source vertex s, and your roommate’s values of v.d and v.pi for every vertex v ∈ V , your algorithm must verify their correctness or find a value that is wrong.

**The verification consists of three steps requiring a total of O(|V|+ |E|) time.**

**(i) Verify that s.d = 0 and s.pi = nil. This takes O(1) time.**

**(ii) For all v ∈ V , v != s, check that v.d = v.pi.d + w(v.pi, v) and that v.d = ∞ if and only if**

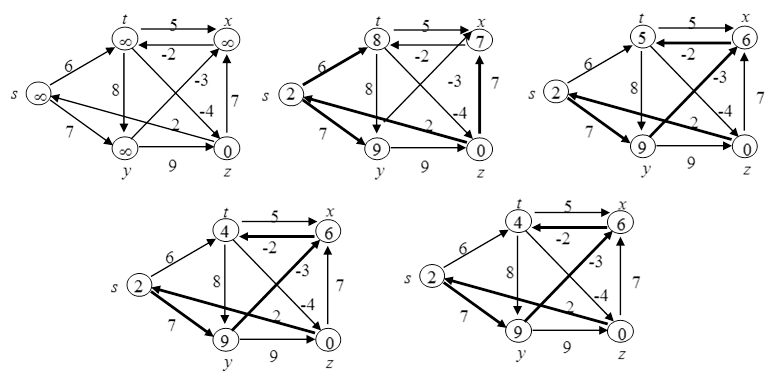
**v.pi = nil. This takes O(|V|) time.**

**(iii) Finally, relax along each edge; none of these relaxations should change a distance value.**

**This takes O(|E|) time.**

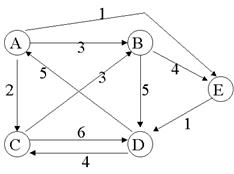
**Failure of any step means the results are wrong; the vertices or edges where steps fail indicate places where the algorithm has produced nonsense.**

7. (3 points) Exercise 24.1-1 (only first question, NOT "Now, change the weight of edge (z, x) to 4 and run the algorithm again, using s as the source.") Run the Bellman-Ford algorithm on the directed graph of Figure 24.4, using vertex *z* as the source. In each pass, relax edges in the same order as in the figure, and show the *d* and π values after each pass.

****

**Using vertex “z” as source. Each pass relaxes the edges in the order (t, x), (t, y), (t, z), (x, t), (y, x), (y, z), (z, x), (z, s), (s, t), (s, y).**

8. (5 points) Compute D(1), D(2), D(3), D(4) and D(5) using Floyd-Warshall algorithm to compute minimum weight paths between all vertices in the below graph. Also keep track of the predecessor vertices (last vertex in the shortest paths).



**CHANGES ARE UNDERLINED AT EACH STEP**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| D0 | **A** | **B** | **C** | **D** | **E** |
| **A** | 0 | 3,A | 2,A | inf | 1,A |
| **B** | inf | 0 | inf | 5,B | 4,B |
| **C** | inf | 3,C | 0 | 6,C | inf |
| **D** | 5,D | inf | 4,D | 0 | inf |
| **E** | inf | inf | inf | 1,E | 0 |
| D1 | **A** | **B** | **C** | **D** | **E** |
| A | 0 | 3,A | 2,A | inf | 1,A |
| **B** | inf | 0 | inf | 5,B | 4,B |
| **C** | inf | 3,C | 0 | 6,C | inf |
| **D** | 5,D | 8,A | 4,D | 0 | 6,A |
| **E** | inf | inf | inf | 1,E | 0 |
| D2 | **A** | **B** | **C** | **D** | **E** |
| **A** | 0 | 3,A | 2,A | 8,B | 1,A |
| **B** | inf | 0 | inf | 5,B | 4,B |
| **C** | inf | 3,C | 0 | 6,C | 7,B |
| **D** | 5,D | 8,A | 4,D | 0 | 6,A |
| **E** | inf | inf | inf | 1,E | 0 |
| D3 | **A** | **B** | **C** | **D** | **E** |
| **A** | 0 | 3,A | 2,A | 8,B | 1,A |
| **B** | inf | 0 | inf | 5,B | 4,B |
| **C** | inf | 3,C | 0 | 6,C | 7,B |
| **D** | 5,D | 7,C | 4,D | 0 | 6,A |
| **E** | inf | Inf | inf | 1,E | 0 |
| D4 | **A** | **B** | **C** | **D** | **E** |
| **A** | 0 | 3,A | 2,A | 8,B | 1,A |
| **B** | 10,D | 0 | 9,D | 5,B | 4,B |
| **C** | 11,D | 3,C | 0 | 6,C | 7,B |
| **D** | 5,D | 7,C | 4,D | 0 | 6,A |
| **E** | 6,D | 8,C | 5,D | 1,E | 0 |
| D5 | **A** | **B** | **C** | **D** | **E** |
| **A** | 0 | 3,A | 2,A | 2,E | 1,A |
| **B** | 10,D | 0 | 9,D | 5,B | 4,B |
| **C** | 11,D | 3,C | 0 | 6,C | 7,B |
| **D** | 5,D | 7,C | 4,D | 0 | 6,A |
| **E** | 6,D | 8,C | 5,D | 1,E | 0 |

