Discrete Mathematics Quiz 2
2008.05.21

Name:_____ Student ID:_____

Instructions. This is a 2-hr close book quiz. Please manage your time well. No dictionary, calculator, PDA, or any other electronic device. Any dishonorable cheating behavior will give you a miserable future. (Totally 120 points)

1. [45%] Let $G = (V, E)$ be a undirected public bus transit network in a metropolitan area, where each vertex $i \in V$ represents a bus stop and each undirected edge $(i, j) \in E$ denotes a route segment between bus stop i and j. Suppose there are totally R bus routes and each route is a simple path in G. Suppose X_{ij} is the set of bus routes that passes both bus stops i and j for each undirected edge $(i, j) \in E$. Let c_{ij} and t_{ij} denote the traveling distance and time between bus stop i and j, respectively, for each edge $(i, j) \in E$. Suppose Tom plans to travel from bus stop s to bus stop t in G.

   (a) [10%] Give a linear-time method to check whether Tom's travel plan is feasible or not (i.e. whether Tom can really travels from s to t via buses in G). What is the complexity of your method? Explain your answers.
   (b) [10%] Suppose Tome needs to find the shortest distance route whose travel time can not exceed T. Formulate this problem as an Integer Programming (IP) problem. (hint: try to use the 1-1 shortest path formulation, add new constraint for time bound)
   (c) [5%] Suppose Tom makes the travel $v_1 \to v_2 \to v_3 \to ... \to v_q$, which passes nodes v_i for $i = 1, ..., q$ in order.
   A bus transfer happens when one takes off at a bus stop v_i from a bus stop v_{i-1} via a bus route $r_{i-1,i} \in X_{i-1,i}$, and then take a different bus route $r_{i,i+1} \in X_{i,i+1}$ from v_i heading for v_{i+1}. Give an upper bound for the number of different routes that Tom may take. (hint: any route segment can form a complete bipartite graph; your answer should be some big-O notation in terms of $\left|X_{v_i v_j}\right|$, the size of $X_{v_i v_j}$, for some route segment $v_i \to v_j$).
   (d) [20%] Following from (c), give a method to calculate the least number of bus transfers required in the travel $v_1 \to v_2 \to v_3 \to ... \to v_q$. Explain why your method works and give its complexity. (hint: following the hint of (c), assign suitable edge weights)

   Ans:
   (a) Just conduct a BFS or DFS on G which takes $O(|E|)$ time to check whether s can connect to t on G or not. If yes, it means someone can take a bus from s to t.

(b)

$$\min \sum_{(i,j)\in E} c_{ij}x_{ij}$$

$$\text{s.t.} \sum_{(i,j)\in E} x_{ij} - \sum_{(j,i)\in E} x_{ji} = \begin{cases} +1, \text{ if } i = s \\ -1, \text{ if } i = t \\ 0, \text{ otherwise} \end{cases} \quad \forall i \in V$$

$$\sum_{(i,j)\in E} t_{ij}x_{ij} \leq T$$

$$x_{ij} \in \{0,1\} \forall (i,j) \in E$$

(c) Since there are $\left|X_{v_iv_j}\right|$ routes connecting stop v_i to stop v_j, totally there are $\prod_{i=1,2,...,q-1} \left|X_{v_iv_{i+1}}\right|$ ways to connect from v_1 to v_q

(d) Let V_0 be a dummy origin vertex, V_q be a dummy destination vertex,.and $V_i = \{v_i^w : w = 1, 2, ..., \left|X_{v_iv_{i+1}}\right|\}$ be a group of $\left|X_{v_iv_{i+1}}\right|$ vertices denoting the set of routes connecting from stop v_i to stop v_{i+1} for $i = 1, 2, ..., q-1$. For each vertex v_1^e in V_1, we construct an edge (V_0, v_1^e) with zero weight. For each vertex v_i^e in V_i $(i = 1, ..., q-2)$, we construct an edge (v_i^e, v_{i+1}^f) that connects to each vertex v_{i+1}^f in V_{i+1}, and assign its weight as 1 (or 0) if they represent the same (or different) route, respectively. For each vertex v_{q-1}^e in V_{q-1}, we construct an edge (v_{q-1}^e, V_q) with zero weight. Therefore, the ways to connect two consecutive bus stops v_i and v_{i+1} can be represented as a compete bipartite graph. Finally a graph G' representing ways to connect from v_1 to v_q can be constructed with $\sum_{i=1}^{q-1} \left|X_{v_iv_{i+1}}\right| + 2$ vertices and $\sum_{i=1}^{q-2} \left|X_{v_iv_{i+1}}\right| \cdot \left|X_{v_{i+1}v_{i+2}}\right| + |X_{v_1v_2}| + \left|X_{v_{q-1}v_q}\right|$ edges. Then, any path connecting from V_0 to V_q would represent a possible way to take buses from v_1 to v_q and its total length equal to the total number of bus transfers required in that way.

Therefore, to find the least number of bustransfers, one can conduct a shortest path algorithm in G' from V_0 to V_q. Note that G' is acyclic and thus we can apply the topological ordering algorithm to calculate a shortest path from V_0 to V_q in $O(\sum_{i=1}^{q-2} \left|X_{v_iv_{i+1}}\right| \cdot \left|X_{v_{i+1}v_{i+2}}\right| + |X_{v_1v_2}| + \left|X_{v_{q-1}v_q}\right|)$ time.

2. [30%] Answer the following questions about a tournament T of n vertices.
(a) [10%] Show that T contains a directed Hamiltonian Path.
(b) [10%] Explain how you can find a directed Hamiltonian Path in a transitive T, and also explain the complexity of your method.
(c) [10%] Explain how do you know whether T is acyclic, and also explain the complexity of your method.

Ans:

(a) The result is trivial for $n = 1$ and $n = 2$. Suppose the result still holds for $n = 3, ..., k$. Now we check the case when $n = k + 1$.

Remove a vertex u from T. Then the remaining subgraph T' is still a tournament (since it still satisfies each vertex connects with every other vertex). Suppose T' has a directed HP:

$v_1 - v_2 - ... - v_k$. If (u, v_1) or (v_k, u) exists, then we can easily find a directed HP in T by connecting u with the HP in T'. Otherwise, let $i > 1$ be a minimal integer such that edge (u, v_i) exists and then (v_{i-1}, u) exists Thus, there exists a HP $v_1 - v_2 - ... - v_{i-1} - u - v_i - v_{i+1} - ... - v_k$.
(b) A transitive T contains a unique directed HP, where each vertex has a different score(=outdegree).. Calculate outdegree for each vertex (takes $O(m) = O(n^2)$ time). Sort vertices by the descending order of their outdegrees (takes $O(n \log n)$ time). For those vertices of the same outdegrees, trace the path by the order, which will give a directed HP (takes $O(n)$). Thus totally it will take $O(n^2)$ time.
(c) T is acyclic iff T has no directed cycle iff T is transitive.
There are many methods to check whether T is transitive. For example, an $O(n^2 + m)$ or $O(n \lg n + m)$ time method to calculate outdegrees (this takes $O(m)$ time), sort the outdegrees (takes $O(n^2)$ or $O(n \lg n)$ time), and then check whether there exists common outdegree (takes $O(n)$ time).

3. [20%] Given a connected graph $G = (V, E)$ where $|V| = n$ and $|E| = m$.
(a) [10%] Give a method withe complexity better than $O(n^4)$ to calculate the number of paths of length $\lfloor \frac{n}{2} \rfloor$ from any vertex $i \in V$ to any vertex $j \in V$. Explain the complexity of your method.
(b) [6%] Suppose G is a undirected $w \times l$ grid graph with $n = (w+1) \times (l+1) \geq 4$, where $w \geq 1$ and $l \geq 1$ are the number of edges in each of the horizontal and verticle line, respectively. What is $\chi(G)$, the chromatic number of G? Explain your answer.
(c) [4%] Following (b), please express m as a function of w and l.

Ans:
(a) This is equivalent to calculating $A^{\lfloor \frac{n}{2} \rfloor}$ where A is the $n \times n$ adjacency matrix. One can recursively calculate $A^{\lfloor \frac{n}{2} \rfloor}$ by $A \to A^2 \to A^4 \to ... \to A^{2^{\lg \lfloor \frac{n}{2} \rfloor}} \to A^{2^{\lg \lfloor \frac{n}{2} \rfloor}+1} \to ... \to A^{\lfloor \frac{n}{2} \rfloor}$ which takes $O(n^3 \lg n)$ time since each multiplication takes $O(n^3)$ time and there are $O(\lg n)$ times of self-multiplication and $O(1)$ times of multiplication from $A^{2^{\lg \lfloor \frac{n}{2} \rfloor}+1}$ to $A^{\lfloor \frac{n}{2} \rfloor}$.
(b) $\chi(G) = 2$, since any cycle in G must be even.
(c) $m = w(l + 1) + l(w + 1)$

4. [15%] Answer the following questions about MST algorithms for a given graph $G = (V, E)$ where $|V| = n$ and $|E| = m$.
(a) [10%] Given a spanning tree T, how do you check whether T is an MST without conducting any MST algorithm from the scratch (i.e. you are NOT ALLOWED to apply Prim's, Kruskal's, or any other MST building algorithm)? Explain your method and its complexity.
(b) [5%] Explain why Kruskal's algorithm takes $O(n \log n)$ time to merge components. Is this operation the bottleneck for Kruskal's algorithm?

Ans:
(a) For each nontree edge (k, l), conduct a DFS/BFS to find the unique path from k to l on T, check whether any edge (i, j) on that path has smaller weight than (k, l). If all staisfied,

then T is an MST. Totally there are $m-n+1$ nontree edges, and each DFS/BFS takes $O(n)$ time on T since it has $n-1$ edges. Thus totally it takes $O((m-n+1)n) = O(mn)$ time.
Or, for each tree edge (i, j) in T, conduct DFS/BFS from i or j separately to mark the vertices on each side (i.e., i side or j side) of the cut with different colors. Say, i side with black color, and j side with red color. Then for each edge (k, l) with different colored end vertices, check whether its weight is larger than the weight of (i, j). If all satisified, then T is an MST. Totally there are n tree edges, each DFS/BFS takes $O(n)$ time on T since it has $n-1$ edges, and one may spend $O(m-n+1)$ time to check the colors of end vertices for each cut edges as well as its weight. Thus totally it takes $O(n(n+m-n+1)) = O(mn)$ time.
(b) When we merge the smaller component to the larger component, we have to relabel the index and the number of vertices of the smaller component for each vertex in the smaller component. It is obvious that the newly merged component will contain at least twice the number of vertices of the original smaller component. In the worst case, such a merge operation will at most be conducted $O(\log n)$ times for each vertex. Thus overall the merge operation takes $O(n \log n)$ time.
This operation is NOT the bottleneck since Kruskal's algorithm has to sort the weights for m edges which will take $O(m \log m)$ time that is larger than $O(n \log n)$ time.