College of Computer & Information Science
Northeastern University
Fall 2017

CS5800
Algorithms
December 11 2017

CS5800 - Algorithms - Final exam

- **Duration, Problems, and Points:** This is a 3 hours exam. There are 6 problems and 1 bonus problem, for a total of 30+2 points. Read all the problems through first and solve them in the order that allows you to make the most progress.
- **A note on grading:** Your grade on any problem will be determined on the basis of the correctness of your solution and the clarity of the description. In case you are not able to present an algorithm that has the desired properties, give the best algorithm you have designed. Show your work, as partial credit may be given. If you need extra space, use the blank facing page.
- **Closed-book and closed-notes exam:** This exam is closed-book. No sources of any kind, electronic or physical, may be consulted during the course of this exam. No collaboration of any kind is permitted.
- **Policy on cheating:** Students who violate the above rules on scholastic honesty are subject to disciplinary penalties. Any student caught cheating will receive an **F** (failing grade) for the course, and the case will be forwarded to the Office of Student Conduct and Conflict Resolution. Sign below and confirm that you are strictly abiding by the rules of this test.

Question	Score	Maximum
1		5
2		5
3		5
4		5
5		5
6		5
7		2
Total		30+2

Signature: _____

- Good luck!

Name: _____

1. A sequence of letters is a palindrome if it is the same whether it is read backward or forward. For example:

   - abccba - is a palindrome.
   - madam - is a palindrome.
   - was it a car or a cat i saw - is a palindrome if we ignore spaces. (Taken from Wikipedia).

   The following are **NOT** palindromes:

   - abab
   - abdooyume
   - abdca

   Given a sequence $X = x_1, x_2, ..., x_n$, denote by $L(i, j)$ (for $i \leq j$) the length of a longest subsequence of $x_i, x_{i+1}, ..., x_j$ that is a palindrome.
   (For example if X = x,a,b,t,r,h,b,a then the subsequence a,b,t,b,a is a longest possible subsequence that is a palindrome, so $L(1, 8) = 5$. The subsequence a,t,a is also a palindrome but not the longest possible. We also have $L(3, 7) = 3$ since b,r,b is a longest palindrome that is a subsequence of b,t,r,h,b)

   Write a recurrence relation that allows to calculate $L(i, j)$ using values $L(i', j')$ that correspond to subsequences of $x_i, x_{i+1}, ..., x_j$.

   Answer: If $x_i = x_j$ than we can add x_i and x_j to any palindrome found in $x_{i+1}, ..., x_{j-1}$. If $x_i \neq x_j$ then a longest subsequence of $x_i, x_{i+1}, ..., x_j$ that is a palindrome can't include both x_i and x_j. Therefore we have:

$$L(i,j) = \begin{cases} 1, & i = j; \\ L(i+1, j-1) + 2, & x_i = x_j; \\ \max\{L(i+1, j), L(i, j-1)\}, & x_i \neq x_j. \end{cases}$$

2. Given a sequence of 0's and 1's we want to organize them such that all the 0's appear before all the 1's. For example if we are given the sequence 11001010, we would like to reorder it as 00001111.

   The operation "reverse(i,j)" reverses the order of the bits in positions i to j. For example performing "reverse(2,6)" on the sequence 0110101 results in 0010111:

$$0\underbrace{11010}1 \overset{\text{reverse(2,6)}}{\longrightarrow} 0\underbrace{01011}_{\text{reversed}}1$$

   Performing "reverse(i,j)" takes $j - i + 1$ time units (reversing the order of k elements takes k time units). We want to reorganize the sequence using only "reverse(i,j)" operations.
   For example, to reorganize the sequence 11001010 we could reverse the order of the elements in positions 3 to 5 by "reverse(3,5)":

$$11\underbrace{001}010 \overset{\text{reverse(3,5)}}{\longrightarrow} 11\ \underbrace{100}_{\text{reversed}}\ 010$$

   Then we can reverse the order of the elements in positions 4 to 7 by "reverse(4,7)":

$$111\underbrace{0001}0 \overset{\text{reverse(4,7)}}{\longrightarrow} 111\ \underbrace{1000}_{\text{reversed}}\ 0$$

   and finally, we reverse the order of positions 1 to 8 by "reverse(1,8)":

$$\underbrace{11110000} \overset{\text{reverse(1,8)}}{\longrightarrow} \underbrace{00001111}_{\text{reversed}}$$

   Design an algorithm that gets a sequence of 0's and 1's and reorders it such that the 0's appear first, using only "reverse(i,j)" operations. Your algorithm should run in $O(n \log n)$ time where n is the length of the sequence.

   The operations your algorithm can do on a sequence of 0's and 1's are:

   - Reverse the order of all the bits from position i to position j.
   - Check whether the ith element of the sequence is 0 or 1 in $O(1)$ time.

   (Hint: A divide and conquer approach might work well for this problem.)

   Answer:

   - Divide the array to 2 "halves".
   - Organize the 2 halves (with zeros before ones) recursively.
   - Find the index i of the first 1 on the "left half".
   - Find the index j of the last 0 on the "right half".
   - Perform "reverse(i,j)"

The recursive formula that describes the running time of this algorithm is:

$$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$$

It follows that the running time of the algorithm is $O(n \log n)$.

3. Let $G = (V, E)$ be a flow network with capacity $c : E \to \mathbb{R}_{\geq 0}$.
For 2 s-t cuts $(S, \overline{S})$ and $(S', \overline{S'})$, we define their intersection to be $(S \cap S', \overline{S \cap S'})$.
Show that if $(S, \overline{S})$ and $(S', \overline{S'})$ are both minimal cuts, then their intersection $(S \cap S', \overline{S \cap S'})$ is also a minimal cut.

Answer: Let $f_0 : E \to \mathbb{R}$ be a maximal flow for G.
Since S is a minimal cut and f_0 is a maximal flow, we have:

$$\text{val}(f_0) = \text{cap}(S, \overline{S}) = \sum_{(u,v) \in ((S \times \overline{S}) \cap E)} c(u, v)$$

On the other hand we have:

$$\text{val}(f_0) = f_0(S, \overline{S}) - f_0(\overline{S}, S) \leq f_0(S, \overline{S}) \leq \sum_{(u,v) \in ((S \times \overline{S}) \cap E)} c(u, v)$$

It follows that:

$$f_0(S, \overline{S}) = \sum_{(u,v) \in ((S \times \overline{S}) \cap E)} c(u, v)$$

and:

$$f_0(\overline{S}, S) = 0$$

We see that we must have $f_0(u, v) = \text{cap}(u, v)$ for any edge of the form $(u, v) \in ((S \times \overline{S}) \cap E)$, and $f_0(u, v) = 0$ for any edge of the form $(u, v) \in ((\overline{S} \times S) \cap E)$.
Let (u, v) be an edge from $S \cap S'$ to $\overline{S \cap S'}$ (that is $u \in S \cap S'$, $v \in \overline{S \cap S'}$). It follows that $v \notin S$ or $v \notin S'$. Assume $v \notin S$. The edge (u, v) goes from S to $\overline{S}$ and hence by what we saw $f_0(u, v) = c(u, v)$.
Let (v, u) be an edge from $\overline{S \cap S'}$ to $S \cap S'$. It follows that $v \notin S$ or $v \notin S'$. Assume $v \notin S$. The edge (v, u) goes from $\overline{S}$ to S and hence by what we saw $f_0(v, u) = 0$.
We see that for any edge e from $S \cap S'$ to $\overline{S \cap S'}$ we have:

$$f_0(e) = c(e)$$

and for any edge e from $\overline{S \cap S'}$ to $S \cap S'$ we have:

$$f_0(e) = 0$$

It follows that:

$$\text{val}(f_0) = f(S \cap S', \overline{S \cap S'}) - f(\overline{S \cap S'}, S \cap S') = \text{cap}(S \cap S') - 0 = \text{cap}(S \cap S')$$

and thus by the min cut max flow theorem $S \cap S'$ is a minimal cut.

4. Let $\{x_1, x_2, ..., x_n\}$ represent n students. In each of the next m days, exactly k students go out for dinner in a restaurant. (Clearly $1 \leq k \leq n$)
In the jth day the set of students that go to the restaurant is $S_j \subseteq \{x_1, x_2, ..., x_n\}$ ($|S_j| = k$). Let d_i denote the total number of times the student x_i goes to the restaurant in days 1 to m. That is:

$$d_i = |\{j \mid i \in S_j, 1 \leq j \leq m\}|$$

For any day $1 \leq j \leq m$ the students must choose one of the students that went to the restaurant, to be the one that will pay the bill.

Design an algorithm that finds a student that will pay the bill in each of the following m days such that no student pays more than $\lceil \frac{d_i}{k} \rceil$ times. (Constructing a corresponding flow network can help with this problem.)

Answer: Define a flow network $G = (V, E)$ by:

$$V = \{s\} \cup \{t\} \cup \{x_1, ..., x_n\} \cup \{S_1, ..., S_m\}$$

$$E = \{(s, x_i)\}_{i=1}^{n} \cup \{(x_i, S_j) \mid x_i \in S_j\} \cup \{(S_j, t)\}_{j=1}^{n}$$

with capacities:

$$c(s, x_i) = \left\lceil \frac{d_i}{k} \right\rceil \ , \ c(x_i, S_j) = 1 \ \ (\text{if } (x_i, S_j) \in E) \ , \ c(S_j, t) = 1$$

The flow that is defined by $f(s, x_i) = \frac{d_i}{k}$, $f(x_i, S_j) = \frac{1}{k}$ (if $(x_i, S_j) \in E$), $f(S_j, t) = 1$ satisfies $\text{val}(f) = m$. A corresponding integer flow solves the problem.

5. Design an algorithm that gets an undirected weighted graph $G = (V, E)$ and determines whether G has a unique minimum spanning tree or whether there exist more then one minimum spanning tree for G. What is the complexity of your algorithm?
(Hint: Try to think how you can cause an algorithm that finds a MST to "prefer" one MST over another MST.)

   Solution: Any version of the following can work. If the edges of G have colors (red and white) we know how to find a MST with maximum number of red edges. (See quiz 3 solution for the algorithm)
   Apply some MST algorithm once. Set all the edges of the minimum spanning tree we found to be white and the rest of the edges to be red. Apply some MST algorithm again in order to find a MST with as many red edges as possible.

6. For an undirected graph $G = (V, E)$ and $v_1, v_2 \in V$, we define $d_2(v_1, v_2)$ in the following way:

   $$d_2(v_1, v_2) = \min_p\{|p| \ : \ p \text{ is a path connecting } v_1 \text{ and } v_2 \text{ with } 2k \text{ edges, } k \in \mathbb{N} \cup \{0\}\}$$

   where $|p|$ is the number of edges in a path p.
   ($d_2(v_1, v_2)$ is the number of edges in a path from v_1 to v_2 that has the smallest number of edges among all paths from v_1 to v_2 that have an even number of edges.)

   Given an undirected graph $G = (V, E)$ and a vertex $v_0 \in V$, describe an efficient algorithm that computes $d_2(v_0, v)$ for all the vertices $v \in V$. What is the complexity of your algorithm?

   Answer: We construct a new graph (undirected) $G' = (V', E')$. The set of vertices V' is a union of two copies of V:

   $$V^1 = \{v^1 \mid v \in V\} \quad , \quad V^2 = \{v^2 \mid v \in V\}$$

   and

   $$V' = V^1 \cup V^2$$

   (If one wishes to be formal then $v^1 = (v, 1)$ and $v^2 = (v, 2)$.)
   The edges of G' are defined by:

   $$E' = \{(u^1, v^2), (u^2, v^1) \mid (u, v) \in E\}$$

   Since G' is bipartite, a path from u^1 to v^1 in G' must have an even number of edges. On the other direction, a path from u to v in G with an even number of edges, corresponds to a path from u^1 to v^1 in G'. For example, if $u, v_1, v_2, ..., v_m, v$ is a path with an even number of edges in G, then $u^1, v_1^2, v_2^1, ..., v_m^2, v^1$ is a path from u^1 to v^1 in G'.

   It follows that to calculate $d_2(u, v)$ in G, it is enough to calculate the shortest path from u^1 to v^1 in G' (which can be done using BFS). Indeed any path with an even number of edges from u to v in G, yields a path from u^1 to v^1 in G' with the same number of edges, and any path from u^1 to v^1 in G' yields a path from u to v in G with the same number of edges (which is an even number of edges). (Note that we allow paths to use the same edge more then once.)

7. (Bonus question)
A Hamiltonian path in an undirected graph is a path that visits each vertex exactly once.
Let L_1 and L_2 be the following languages:
L_1 is the language of all the undirected graphs that have a hamiltonian path:

$$L_1 = \{G \mid \text{The graph } G \text{ has a Hamiltonian path}\}$$

L_2 is the language of all the undirected graphs that have a spanning tree with maximum degree smaller or equal to 10:

$$L_2 = \{G \mid \text{There exists a spanning tree } T \text{ for } G \text{ such that } \forall v \in V,\ \deg_T(v) \leq 10\}$$

Show that $L_1 \leq_p L_2$.

Answer: Given an undirected graph G, we construct a new graph G' by connecting each vertex to 8 new vertices. That is, for each vertex $v \in V$ we add 8 vertices $v_1, v_2, ..., v_8$ and connect them with an edge to v.

Clearly if the new graph G' has a spanning tree T then T must contain all the "new edges". Therefore if T is a spanning tree such that $\deg_T v \leq 10$ for each $v \in V$ then T corresponds to a spanning tree T_0 in G that has degree at most 2 for all vertices. It mean that T_0 is a Hamiltonian path in G.

Clearly if G has a Hamiltonian path then G' has a spanning tree with maximum degree ≤ 10.

This page is intentionally left blank. You may use it for your answers.

This page is intentionally left blank. You may use it for your answers.