Mid Semester Test, September 18, 2018

CS345: Algorithms II

Max Marks 53

Instructions: Please try to be brief and to the point. Start each question from a new page and clearly mark the question numbers.

Question 1. [Marks 8].

Let G be a graph and M be a matching in it. Let C be an augmenting cycle in (G,M). Recall that one of the theorems discussed in the class states that if (G,M) has an augmenting path, then $(G/C,M/C)$ also has an augmenting path.

Prove or disprove that if M' is a maximum matching for G/C, then M'^C is a maximum matching for G. If the claim is true, then prove it otherwise give a counter example.

Solution

The claim is false. Consider the graph $(\{1,2,\ldots,11\},\{\{1,2\},\{2,3\},\{3,4\},\{4,5\},\{5,6\},\{6,7\},$ $\{7,3\},\{4,8\},\{5,9\},\{6,10\},\{7,11\}\})$. Let $M = \{\{1,2\},\{4,5\},\{6,7\}\}$. So $C = \{\{3,4\},\{4,5\},\{5,6\},\{6,7\},$ $\{7,3\}\}$ is an augmenting cycle.

The $G/C = (\{1,2,8,9,10,11,v_C\},\{\{1,2\},\{2,v_c\},\{8,v_C\},\{9,v_C\},\{10,v_C\},\{11,v_C\}\})$. Then a maximum matching for G/C is $M' = \{\{1,2\},\{8,v_c\}\}$. So $M'^C = \{\{1,2\},\{8,4\},\{5,6\},\{7,3\}\}$. It has 4 edges. But the maximum matching in G, for example $M_1 = \{\{3,2\},\{4,8\},\{5,9\},\{6,10\},\{7,11\}\}$, has 5 edges.

Question 2. [Marks 5+2].

Given two sequences $A : a_1, \ldots, a_n$ and $B : b_1, \ldots, b_m$. Design an algorithm to find the number of subsequences (not substrings) of B which are equal to A. For example if $A = apple$ and $B = appple$, then the answer is 3. Design a dynamic programming based algorithm to solve the problem.

(a) Write the recurrence relation.

(b) Write the pseudocode for the algorithm.

Hint: Suitably modify the longest common subsequence algorithm.

Solution

(a) Let $S(i, j)$ denote the number of subsequences of $a_1, \ldots, a_i$ which are equal to $b_1, \ldots, b_j$.

Base Case: $S(0, j) = 0$ for all $j > 0$ and $S(0,0) = 1$.

Recurrence Relation:

$S(i, j) = S(i-1, j)$ if $a_i \neq b_j$

$\quad\quad S(i-1, j) + S(i-1, j-1)$ if $a_i = b_j$.

(b) Complete yourself.

Question 3. [Marks 8].

A machine converts one currency note of denomination k into notes of denominations $\lfloor k/2 \rfloor, \lfloor k/3 \rfloor, \lfloor k/4 \rfloor$ and $\lfloor k/5 \rfloor$ in one run. For example a 12 denomination note will get converted to 4 notes of face value $6,4,3,2$ respectively. Thus the total value adds up to 15. One can use the machine any number of times to maximize the value of their currency notes. Let $max(k)$ denote the largest value a currency note of face value k can be converted to. In our example the 6 denomination note can be converted to the notes of values $3,2,1,1$. Thus the total value becomes 16. Verify that no further conversion helps increase the value. So $max(12) = 16$.

Determine an upperbound for the number of times the machine needs to be run to convert a note of value k to the notes of total value $max(k)$. Show all the steps of the analysis. Hint: Show the number to be $O(k^{\alpha})$ for a suitable values of α.

Solution

Consider the reduction tree where the root node corresponds to the original note of face value k. Each node corresponds to a note that was generated in the process. Each internal node correspond to that note which was entered into the machine and its 4 child nodes correspond to the 4 notes output by the machine. So the number of internal nodes is the number of times the machine was used.

Suppose a subtree rooted at a node corresponding a note of face value j has at most $j^{\alpha} - 1$ nodes. The smallest note which will be converted has face value 6. So we require that $6^{\alpha} - 1 \geq 1$. This holds trivially true for any $\alpha \geq 1$.

So if the root node corresponds to value k, then we want $k^{\alpha} - 1 \geq (k/2)^{\alpha} - 1 + (k/3)^{\alpha} - 1 + (k/4)^{\alpha} - 1 + (k/5)^{\alpha} - 1 + 1$. Therefore it will suffice if $k^{\alpha} \geq k^{\alpha}(1/2^{\alpha} + 1/3^{\alpha} + 1/4^{\alpha} + 1/5^{\alpha}))$ or $1 \geq 1/2^{\alpha} + 1/3^{\alpha} + 1/4^{\alpha} + 1/5^{\alpha}$. The smallest value of α which satisfies this condition is appximately 1.24.

Question 4. [Marks 5+10].

(a) Consider a set of time intervals $X = \{(s_1,t_1),(s_2,t_2),\ldots,(s_n,t_n)\}$. Let I be the collection of all subsets of X in which intervals do not overlap. Prove that (X, I) is not a matroid. Hint: You can prove it by giving a suitable example which violates a theorem on matroids.

(b) Let $G = (A,B;E)$ be a bipartite graph. A subset $S \subseteq A$ is said to be *matchable* if there exists a matching of G in which all the vertices of S are matched. Prove that $(A,\{S|S \text{ is matchable}\})$ is a matroid. For clarity use an example to explain the proof of exchange property.

Solution

(a) Consider an instance with $X = \{(1,2),(1.5,2.5),(2,3)\}$. It has two maximal non-overlapping sets $I_1 = \{(1,2),(2,3)\}$ and $I_2 = \{(1.5,2.5)\}$. They have different cardinalities so (X, I) cannot be a matroid because in a matroid all maximal independent sest have same cardinality.

(b) (i) Emptyset is matchable because any matching matched "all of its vertices" which is an emptyset.

(ii) Let S_1 be a matchable set and $S_2 \subseteq S_1$. So there exists amatching M in which all the vertices of S_1 are matched. Since S_2 is a subset of S_1, all of S_1 vertices are also matched under M. Hence S_2 is also a matchable set.

(iii) Let S_1 and S_2 be two matchable sets such that $S_1 \setminus S_2$ and $S_2 \setminus S_1$ are non-empty. Let M'_1 and M'_2 be the matchings with respect to which S_1 and S_2 are respectively matched. Delete those edges from M'_1 which match vertices from $A \setminus S_1$. Let resulting matching be M_1. Similarly define M_2. So $U_1 = A \setminus S_1$ is the set of unmatched vertices in M_1 and $U_2 = A \setminus S_2$ is the set of unmatched vertices in M_2.

Let $x \in S_1 \setminus S_2$. So $x \in U_2 \setminus U_1$. Consider the maximal M_1-M_2 alternating path, say, $P : x = x_1, y_1, x_2, y_2, \ldots$. There are two cases to consider: the path length is odd and even.

If it is even, then P is $x = x_1, y_1, x_2, y_2, \ldots, y_{k-1}, x_k$. Then define the matching $M''_2 = (M_2 \setminus \{\{y_i,x_{i+1}\}|1 \leq i \leq k-1\}) \cup \{\{x_i,y_i\}|1 \leq i \leq k-1\}$. In this matching $(S_2 \cup \{x\}) \setminus \{x_k\}$ is matchable. Since $\{y_{k-1},x_k\} \in M_2$, maximality of P implies that x_k is not matched in M_1. So $x_k \in S_2 \setminus S_1$.

If it is odd then P is $x = x_1, y_1, x_2, y_2, \ldots, y_{k-1}, x_k, y_k$. Define $M''_2 = (M_2 \setminus \{\{y_i,x_{i+1}\}|1 \leq i \leq k-1\}) \cup \{\{x_i,y_i\}|1 \leq i \leq k\}$. In this matching $S_2 \cup \{x\}$ is matched. So for any $y \in S_2 \setminus S_1$, $(S_2 \cup \{x\}) \setminus \{y\}$ is also matchable.

This proves that exchange property holds.

Question 5. [Marks 6+9].

Let $G = (V, E, w)$ be an edges weighted undirected graph with non-negative weights. Let us assume that $w(x, y) = \infty$ if $\{x, y\}$ is NOT an edge. Consider the following algorithm to compute weighted distance (weight of the minimum weight path) between all pairs of vertices. Note that $w(x, y) = w(y, x)$.

```
for x, y ∈ V do
 | d[x, y] = w(x, y);
end
for k := 1 to ⌊log |V|⌋ do
 | for x ∈ V do
 | | for y ∈ V \ {x} do
 | | | for z ∈ V \ {x, y} do
 | | | | if d[y, z] > d[y, x] + d[x, z] then
 | | | | | d[y, z] := d[y, x] + d[x, z];
 | | | | end
 | | | end
 | | end
 | end
end
return Matrix d;
```

Let $\delta_w(x, y)$ be the weight of the minimum weight path between x to y. Also let $\delta_u(x, y)$ denote the number of edges in that path. If the minimum weight path is not unique, then $\delta_u(x, y)$ is minimum among them.

(a) State an invariant for the outermost For-loop (i.e., for each k) of the second statement in this algorithm, which can be used to prove the correctness of the algorithm. You may use notations $\delta_w()$ and $\delta_u()$ in your invariant.

(b) Prove your invariant.

Solution

(a) Let $d_k[x, y]$ denote the value of $d[x, y]$ after k iterations. Then the invariant is:
If $\delta_u(x, y) \leq 2^k$ then $d_k[x, y] = \delta_w(x, y)$.

Observe that after $k = \lceil \log n \rceil$ passes every pair of vertices, x, y, satisfies $\delta_u(x, y) \leq n - 1 \leq 2^{\log n} \leq 2^k$. Hence the invariant implies that after $k = \lceil \log n \rceil$ passes, $d_k[x, y] = \delta_w(x, y)$ for all pairs x, y.

(b) To prove that the above statement holds after each iteration first consider the case $k = 0$ (before the first iteration). If for some pair x, y, $\delta_u(x, y) = 1 = 2^0$, then the edge $\{x, y\}$ must be the minimum weight path between x and y. Since initially $d_0[x, y] = w(x, y)$, $d_0[x, y] = \delta_w(x, y)$ for such pairs.

The induction step: Suppose the statement holds for $k = j - 1$. Consider $k = j$. Let the minimum weight path between a vertex pair x and y is $P : x = z_0, z_1, \ldots, x_r = y$ where $r \leq 2^j$. Split it into $P_1 : z_0, \ldots, z_{2^{j-1}}$ and $P_2 : z_{2^{j-1}+1}, \ldots, z_r$. Since each weight is non-negative, the sub-paths of P must be optimal. Hence P_1 and P_2 are also optimal. Let $z_{2^{j-1}} = u$. So $\delta_w(x, u) = w(P_1)$ and $\delta_u(x, u) \leq 2^{j-1}$. Similarly $\delta_w(u, y) = w(P_2)$ and $\delta_u(u, y) \leq 2^{j-1}$.

From induction hypothesis $w(P_1) = \delta_w(x, u) = d_{j-1}[x, u]$ and $w(P_2) = \delta_w(u, y) = d_{j-1}[u, y]$. During j-th round we will have $d_j[x, y] \leq d_{j-1}[x, u] + d_{j-1}[u, y] = w(P_1) + w(P_2) = w(P) = \delta_w(x, y)$. Later on we will show that $d[p, q] \geq \delta_w(p, q)$ for all p, q. So that will imply that $d_k[x, y] = \delta_w(x, y)$. This completes the proof of the correctness of the invariant.

Claim 1 *$d_i[p, q] \geq \delta_w(p, q)$ for all vertices p, q.*

Proof. $d_i[p, q]$ will be finite if and only if either (i) $d_i[p, q] = w(p, q)$, or (ii) there exists r such that $d_i[p, q] = d_j[p, r] + d_j[r, q]$ for some $j < i$.

We will prove using induction that $d_i[p,q]$ is the weight of a walk from p to q.

If $d_i[p,q] = w(p,q)$ and the corresponding walk is the edge $\{p,q\}$.

If $d_j[p,r] + d_j[r,q]$ for some $j < i$, then from the induction hypothesis $d_j[p,r]$ is the weight of a walk from p to r, $d_j[r,q]$ is the weight of a walk from r to q. So $d_i[p,q] = d_j[p,r] + d_j[r,q]$ is the weight of the adjoined walk going from p to q.

This means $d_i[p,q]$ is at least $\delta_w(p,q)$ because in presence of non-negative weights the weight of the minimum-weight walk is $\delta_w()$. $\square$