Modern Algorithm Design : Quiz 2
Full Marks : 25
Time : 1 hour

27/9/2024

Problem 1. (5 points) Prove that given a complete weighted graph on n vertices, where the edge-weights form a metric, the MST of the graph is a $(n-1)$-stretch spanning tree.

Proof. Since the graph is a complete graph and the edge weights form a metric, the shortest path for any pair of vertices (u, v) is the edge uv itself. Let the MST of the graph be T. For any pair of vertices (u, v), let $P_T(u, v)$ denote the unique path between u and v in T and $|P_T(u, v)|$ denotes the number of edges in the path.

For any pair of vertices (u, v) such that $|P_T(u, v)| \geq 2$, observe the following: the edge $uv \notin T$ and for any edge $e \in P_T(u, v)$, weight of edge e is at most the weight of edge uv that is $w(e) \leq w(uv)$. This is because $P_T(u, v)$ along with edge uv creates a cycle and by the construction of MST, edge uv is not in T because it is heaviest edge of the cycle. Clearly, the cost of path $P_T(u, v)$ is at most $|P_T(u, v)| \cdot w(uv)$. Since, $|P_T(u, v) \leq (n-1)$, T is a $(n-1)$-stretch spanning three. □

Problem 2. (5 points) In lectures, you saw an algorithm for finding the minimum cost perfect matching in a bipartite graph assuming that a perfect matching exists. Show how to use this algorithm to solve the maximum weight matching in bipartite graphs without the assumption that a perfect matching exists in the graph. You may use only one call to the minimum cost perfect matching algorithm apart from other modifications to the graph.

Proof. Consider the given weighted bipartite graph to be $G = (A \cup B, E)$. We solve the problem of maximum weight matching in the following three steps.

1. **Create a modified graph :** We create a modified graph $G^* = (A^* \cup B^*, E^*)$ from G with the following two three steps.
   (a) Vertex set of G^* is same as the vertex set of G that is $A^* \cup B^* = A \cup B$.
   (b) If one side of the graph has more vertices than the other side, add dummy vertices to $A^* \cup B^*$ to make both side equal.
   (c) For each pair of vertices (u, v) such that $uv \in E$, put an edge uv in E^* with edge weight $-w(uv)$. Here, $w(uv)$ is the weight of uv in G.
   (d) For each pair of vertices (u, v) such that $uv \notin E$, put an edge uv in E^* with edge weight ∞.
2. **Run algorithm on modified graph :** We run the minimum cost perfect matching algorithm on the modified graph G^*. Let M^* be the optimal matching on G^* returned by the algorithm.

3. **Get solution to the original graph from the solution of modified graph :** In this step, we remove all the edges from M^* such that the cost of the edge is ∞ in G^*. Let $\mathcal{M}$ be the set of edges after the removal. Report $\mathcal{M}$ be the optimal matching of maximum weight for graph G.

$\square$

Problem 3. (10 points) Suppose you have access to an algorithm $\mathcal{A}$ that tells you, given a graph (not necessarily bipartite), whether it contains a perfect matching or not but it does not return the perfect matching. Show how you can use $\mathcal{A}$ to *find a perfect matching* if one exists or return that not such matching exists. Your algorithm should use only polynomially many calls to $\mathcal{A}$.

Proof. Consider the given graph to be $G = (V, E)$. $\mathcal{M}$ is the set of edges which will be reported by our algorithm as a perfect matching of the graph. Initially, $\mathcal{M}$ is empty. For simplicity, we consider algorithm $\mathcal{A}$ returns *YES* is there is perfect matching, otherwise returns *NO*. Now, our algorithm is as follows.

Algorithm 1: Find-Perfect-Matching(G)

```
if A returns NO then
    return no perfect matching exists
else
    for each u ∈ V do
        for each uv ∈ E do
            if A returns YES on the graph (G \ {uv}) then
                remove edge uv from G
            else
                M ← M ∪ {uv}
                remove all edges adjacent to u or v form G, except edge uv
            end
        end
    end
end
```

The above algorithm makes $O(|E|)$ many calls to the algorithm $\mathcal{A}$. $\square$

Problem 4. (5 points) In the last lecture, we saw that the bipartite matching polytope - that is the convex hull of the characteristic vectors of perfect matchings - is exactly same as the polytope defined by a natural linear program for perfect matching. Show by giving an example that this property *does not hold* in case of non-bipartite graphs. In particular, show that there can be a vertex solution to the natural LP which has fractional assignments.

Proof. Given a graph $G = (V, E)$, recall the LP constraints for perfect matching is given by,

$$\forall u \in V; \quad \sum_{v : uv \in E} x_{uv} = 1$$

$$\forall e \in E; \quad x_e \geq 0$$

For the example, consider any cycle C of odd length. One may observe that there is no perfect matching for C. However, for each edge $e \in C$, we assign $x_e = \frac{1}{2}$ in the LP and this satisfies all the constraints of the LP. $\square$