CS261: Problem Set #2

Due by 11:59 PM on Tuesday, May 5, 2015

Instructions:

(1) Form a group of 1-3 students. You should turn in only one write-up for your entire group.

(2) Turn in your solutions by email to `cs261submissions@gmail.com`. Please type your solutions if possible and feel free to use the LaTeX template provided on the course home page.

(3) Write convincingly but not excessively.

(4) Some of these problems are difficult, so your group may not solve them all to completion. In this case, you can write up what you've got (subject to (3), above): partial proofs, lemmas, high-level ideas, counterexamples, and so on.

(5) Except where otherwise noted, you may refer to your course notes and the specific supplementary readings listed on the course Web page *only*. You can also review any relevant materials from your undergraduate algorithms course.

(6) You can discuss the problems verbally at a high level with other groups. And of course, you are encouraged to contact the course staff (via Piazza or office hours) for additional help.

(7) If you discuss solution approaches with anyone outside of your group, you must list their names on the front page of your write-up.

(8) Refer to the course Web page for the late day policy.

Problem 7

[ESTIMATED DIFFICULTY LEVEL: Not too bad.] A *vertex cover* of an undirected graph (V, E) is a subset $S \subseteq V$ such that, for every edge $e \in E$, at least one of e's endpoints lies in S.[1]

(a) Prove that in every graph, the minimum size of a vertex cover is at least the size of a maximum matching.

(b) Give a non-bipartite graph in which the minimum size of a vertex cover is strictly bigger than the size of a maximum matching.

(c) Prove that in every bipartite graph, the minimum size of a vertex cover equals the size of a maximum matching.

[Hint: Use either the max-flow/min-cut theorem (and see Exercise 9) or directly apply the matching arguments from lecture. Also, it might be easier to solve part (d) first.]

(d) Prove that the problem of computing a minimum-cardinality vertex cover can be solved in polynomial time in bipartite graphs.[2]

[1] Yes, the problem is confusingly named.

[2] In general graphs, the problem turns out to be NP-hard (you don't need to prove this).

Problem 8

[ESTIMATED DIFFICULTY LEVEL: Somewhat difficult.] This problem considers the special case of maximum flow instances where edges have integral capacities and also

(*) for every vertex v other than s and t, the capacity of v (as defined in Problem 3) is at most 1.

Your tasks:

(a) Prove that the maximum flow problem can be solved in $O(m\sqrt{n})$ time in networks that satisfy (*). (As always, m is the number of edges and n is the number of vertices.)

[Hint: proceed as in Problem 5, but prove a stronger version of part (a) of that problem.]

(b) Prove that the maximum bipartite matching problem can be solved in $O(m\sqrt{n})$ time.

[Hint: examine your reduction in Exercise 9.]

Problem 9

[ESTIMATED DIFFICULTY LEVEL: A bit difficult.] Consider the maximum weighted matching problem in general (not necessarily bipartite) graphs. The input is an undirected graph with a non-negative weight w_e for each edge $e \in E$, and the goal is to compute a matching M that maximizes the total weight $\sum_{e \in M} w_e$. This problem can be solved in polynomial time, using generalizations of the techniques we covered in lecture (augmenting paths, blossom contractions, and suitable prices). All known polynomial-time algorithms for the problem are fairly slow, though, and in some situations we're willing (or forced) to give up correctness in exchange for speed.

In this problem, we'll consider the following greedy algorithm, which is easy to implement in $O(m \log n)$ time:

1. $M = \emptyset$.
2. Relabel the edges $E = \{1, 2, \ldots, m\}$ so that $w_1 \geq w_2 \geq \cdots \geq w_m \geq 0$.
3. For $i = 1$ to m:
   (a) If $M \cup \{i\}$ is a matching, then $M := M \cup \{i\}$.
4. Return M.

For the analysis:

(a) Give an example in which this algorithm outputs a matching with total weight only 50% times that of the maximum possible. (Feel free to break ties in an arbitrary way.)

(b) Prove that the algorithm always outputs a matching with total weight at least 50% times that of the maximum possible.

[Hint: if the greedy algorithm adds an edge e to M, how many edges in the optimal matching can this edge "block"? How do the weights of the blocked edges compare to that of e?]

Problem 10

[ESTIMATED DIFFICULTY LEVEL: A bit difficult.] This problem considers the maximum-weight bipartite matching problem. The input is a bipartite graph $G = (A, B, E)$. Assume without loss of generality that $|A| \leq |B|$. Each edge $e \in E$ has a weight w_e, which we assume is an integer in $\{0, 1, 2, \ldots, W\}$. The goal is to compute a matching M that maximizes $\sum_{e \in M} w_e$. We've already seen one algorithm for this problem, in Lecture #7 (see also Exercise #16). This problem presents an alternative algorithm which is simple and intuitive, albeit only approximately correct and pseudo-polynomial-time. Intuitively, the algorithm allows

the vertices on the left-hand side to "compete" for vertices on the right-hand side, using prices to mediate the competition.

Formally, the algorithm is as follows. By an "eligible edge" with respect to the current prices, we mean an edge $e = (i, j)$ with $w_e \geq p(j) + 1$.

1. Initialize $M = \emptyset$.
2. For every $j \in B$, initialize $p(j)$ to 0.
3. While there is a vertex $i \in A$ that is not matched in M and is incident to at least one eligible edge:
   (a) Choose an arbitrary such vertex i and choose an edge (i, j) maximizing $w_{ij} - p(j)$. [Intuitively, i picks its "favorite vertex" given the current prices.]
   (b) If M already contains an edge e incident to j:
      i. Remove e from M.
      ii. Increment the price $p(j)$.
   (c) Add edge (i, j) to M.
4. Return M.

Here's the analysis:

(a) Prove that the algorithm eventually terminates, and that the final set M is a matching.

(b) Prove that the algorithm maintains the following invariant: if $e = (i, j) \in M$, then

$$w_e - p(j) \geq w_{e'} - p(j') - 1 \tag{1}$$

for every edge $e' = (i, j') \in \delta(i)$.

(c) Prove that the algorithm maintains the following invariant: if a vertex $j \in B$ is unmatched in M, then $p(j) = 0$.

(d) Prove that the output M of the algorithm has total weight within $|A|$ of the maximum possible.

[Hint: use (b) and (c). As a warm up, show that if (1) held without the "-1" term, then the final matching would be optimal.]

(e) Give a running-time analysis of the algorithm.

Problem 11

[ESTIMATED DIFFICULTY LEVEL: Pretty difficult.] This problem outlines an algebraic approach to non-bipartite matching that is radically different from any of the algorithms that we've discussed in lecture. We focus only on the problem of checking whether or not a graph (not necessarily bipartite) has a perfect matching; similar ideas apply more generally to computing the size of a maximum matching.

Let $G = (V, E)$ be an undirected graph with $V = \{1, 2, \ldots, n\}$. The *Tutte matrix* T of G is a $V \times V$ matrix where entry t_{ij} is the indeterminate x_{ij} if $i < j$ and i, j are connected by an edge in G; $-x_{ji}$ if $i > j$ and i, j are connected by an edge in G; and 0 otherwise. For example, the Tutte matrix of the triangle is

$$\begin{pmatrix} 0 & x_{12} & x_{13} \\ -x_{12} & 0 & x_{23} \\ -x_{13} & -x_{23} & 0 \end{pmatrix}.$$

Write S_n for the set of permutations of n elements. Recall that the *sign* $\text{sgn}(\sigma)$ of a permutation $\sigma \in S_n$ is -1 raised to the number of its inversions (i.e., the number of pairs (i, j) with $i < j$ but $\sigma(i) > \sigma(j)$). For $\sigma \in S_n$, write $P_\sigma = \prod_{i=1}^{n} t_{i\sigma(i)}$, so that

$$\det T = \sum_{\sigma \in S_n} \text{sgn}(\sigma) P_\sigma. \tag{2}$$

We can regard (2) as a polynomial in the variables $\{x_e\}_{e \in E}$.

(a) Identify permutations with directed cycle covers of V.[3] Let $O_n \subseteq S_n$ denote the permutations that include at least one odd cycle of length at least 3. Prove that

$$\sum_{\sigma \in O_n} \text{sgn}(\sigma) P_\sigma.$$

is the zero polynomial.

[Hint: what happens if you reverse the direction of an odd cycle?]

(b) Prove that G has a perfect matching if and only if there is a permutation $\sigma \in S_n \setminus O_n$ with $P_\sigma \neq 0$.

(c) Prove Tutte's Theorem: G has a perfect matching if and only if $\det T$ is not the zero polynomial.

(d) Use (c) to give a randomized reduction from the problem of checking if a graph has a perfect matching to the problem of computing the determinant of an integer-valued matrix. The resulting randomized matching algorithm should err with probability at most $\frac{1}{n}$.

[Hint: substitute random values for the x_{ij}'s. Look up the Schwartz-Zippel Lemma, which you are free to use without proof. (Though its proof, which is short and sweet, is also worth reading.)]

Problem 12

[ESTIMATED DIFFICULTY LEVEL: Pretty difficult.] The goal of this problem is to explore the beautiful structure of the set of maximum-cardinality matchings of a graph $G = (V, E)$ (not necessarily bipartite). Define

$$B = \{v \in V \,:\, \text{there exists a maximum matching } M \text{ in which } v \text{ is exposed}\}. \tag{3}$$

Define C as the set of vertices of $V \setminus B$ that have at least one neighbor in B. Define $D = V \setminus (B \cup C)$ as the remaining vertices.

(a) Recall from lecture the Tutte-Berge formula: the size of a maximum matching is exactly

$$\min_{S \subseteq V} \frac{1}{2} \left[|V| - (\text{oc}(S) - |S|) \right], \tag{4}$$

where $\text{oc}(S)$ denotes the number of odd-size connected components in the graph induced by $V \setminus S$. (We gave a direct argument that (4) is an upper bound on the maximum matching; the proof of correctness of Edmonds's algorithm shows that equality always holds.)

Prove that the set C defined above achieves the minimum in (4).

(b) Prove that in every maximum matching M and for every vertex $v \in C$, there is an edge $(v, w) \in M$ with $w \in B$.

(c) Prove that every maximum matching M contains a perfect matching of the graph $G[D]$ induced by D.

(d) Consider a graph G that does not have a perfect matching. (So the set B in (3) is non-empty.) Consider Edmonds's algorithm at termination, with its final unsuccessful search for augmenting path yielding a collection of search trees (one rooted at each exposed vertex). Let A denote the vertices labeled "even," and note that these can be supernodes (i.e., can correspond to subsets of vertices of the original graph G). Let A' denote all original vertices of G contained in some vertex of A. Prove that A' is precisely the set B of vertices defined in (3) above.[4] (The original problem statement was to prove that the "odd" vertices at termination correspond to the set C; you can alternatively prove this version, if you prefer.)

[3] In more detail, derive from a permutation σ a directed graph with vertices V and with an arc (i, j) whenever $\sigma(i) = j$. This directed graph has in- and out-degree 1 everywhere, and hence is a collection of directed cycles.

[4] Example: if G is a triangle, then Edmonds's algorithm terminates with A a single (super)node and A' is all three vertices.