CS369B: Problem Set #4

Due in class on Thursday, March 6, 2008

Instructions: Same as the first three problem sets.

Problem 16

Tutte's Theorem. Let $G = (V, E)$ be an undirected graph with $V = \{1, 2, \ldots, n\}$. Recall that the *Tutte matrix* T of G is a $V \times V$ matrix where entry t_{ij} equals the indeterminate x_{ij} if $(i, j) \in E$ with $i < j$, equals $-x_{ij}$ if $(i, j) \in E$ with $i > j$, and equals 0 otherwise. 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 (pairs (i, j) such that $i < j$ but $\sigma(i) > \sigma(j)$). For $\sigma \in S_n$, write $P_\sigma = \prod_{i=1}^{n} t_{i\sigma(i)}$, so that

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

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

$$\sum_{\sigma \in O_n} \text{sgn}(\sigma) P_\sigma = 0.$$

[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 $\det B$ is not the zero polynomial.

(d) Prove more generally that if G has a perfect matching, then $\det B$ includes a term with coefficient ± 1.

[This is useful because it means that $\det B$ is still zero modulo a prime p. Thus randomized algorithms based on the Tutte matrix need only use $\bmod p$ operations, which saves on bit complexity.]

Problem 17

Maximum Matchings via the Tutte Matrix.

(a) Prove that an $n \times n$ skew-symmetric matrix with n odd is singular.

(b) Recall that one way to define the rank of a matrix A is as the largest k for which A has a $k \times k$ non-singular submatrix. (The index sets for the rows and columns of this submatrix must have the same cardinality, but they need not be the same set.) Recall that a submatrix of a square matrix is *principal* if the index sets of the rows and columns are the same. Prove that a skew-symmetric matrix of rank k has a $k \times k$ non-singular principal submatrix.

[Hint: skew-symmetry gives a correspondence between maximal sets of linearly independent rows and those of columns. The result then follows from basic linear algebra.]

(c) Using (a) and (b), prove that a skew-symmetric matrix has even rank.

[Recall that we used this fact in justifying the correctness of the Rabin-Vazirani algorithm.]

(d) Using (b) and Tutte's Theorem, prove the following generalization of Tutte's Theorem: the rank of the Tutte matrix of a graph G is exactly twice the size of a maximum matching of G.

(e) [Do not hand in.] Convince yourself that (d) implies a Monte Carlo $\tilde{O}(M(n))$-time algorithm for computing the size of a maximum matching of a graph, where $M(n)$ is the time required to multiply two $n \times n$ matrices.

(f) Using (e), reduce the problem of constructing a maximum matching of a graph to the problem of constructing a perfect matching of a graph (with $\tilde{O}(M(n))$ overhead).

Problem 18

Gallai-Edmonds Partition. Let $G = (V, E)$ be a graph. Let B be the nodes of G left exposed in some maximum matching. (B could range from the empty set, as in a graph with a perfect matching, to all of V, as in an odd cycle.) Let C denote the nodes not in B but adjacent to a node in B. Let D denote the rest of the nodes.

(a) Prove that every maximum matching of G has the following structure: it includes a perfect matching of the subgraph induced by D, and it matches each vertex of C to some vertex of B.

[Hint: start by proving that the set C achieves equality in the Tutte-Berge formula.]

(b) Prove that the partition (B, C, D) can be immediately deduced after the termination of Edmonds's blossom algorithm (i.e., the point at which the algorithm finally gets stuck).

[Hint: start by identifying a relationship between the even nodes of the alternating trees and one of the three sets.]

Problem 19

Applications of Matchings to Other Combinatorial Problems.

(a) Let $G = (V, E)$ be an undirected graph and $T \subseteq V$ an even number of vertices. A *T-join* is a subset $F \subseteq E$ of edges such that the odd-degree vertices of (V, F) are precisely T. Suppose the edges of G have nonnegative costs. Prove that the problem of computing a minimum-cost T-join reduces to min-cost matching.

(b) Reduce the min-cost T-join problem with arbitrary costs to that with nonnegative costs.

[Hint: Consider writing the cost $c(F)$ of a set F of edges as $|c|(F \triangle N) + c(N)$, where N denotes the negative-cost edges and $\triangle$ denotes symmetric difference. Note that the first term is nonnegative while the second is a constant, independent of F.]

(c) An *edge cover* of a graph (that has no isolated vertices) is a subset of edges that covers every vertex at least once. Prove that the min-cardinality version of edge cover reduces to maximum matching.

(d) Prove that the min-cost version of edge cover (where edges have general real-valued costs) reduces to maximum-weight matching.

[Hint: Consider transforming the edge costs before invoking a matching algorithm.]

Problem 20

Applications of T-Joins.

(a) Consider the single-source shortest-path problem. Explain why the undirected version of the problem easily reduces to the directed one with nonnegative edge costs, and why this is not the case for undirected graphs with negative-cost edges (but no negative undirected cycles).

(b) Consider the single-source shortest-path problem in undirected graphs with negative-cost edges but no negative-cost cycles. Prove that this problem reduces to the minimum-cost T-join problem.

(c) Consider the problem of computing a max-weight cut of an undirected graph (edges can have positive or negative weight). This is, of course, NP-hard in general graphs. Prove that it is poly-time solvable in planar graphs.

[Hint: Consider the dual graph.]

(d) Consider an undirected graph with nonnegative edge costs. Give a polynomial-time algorithm that computes the minimum-cost closed walk that traverses every edge at least once.