Advanced Algorithms - Problem set 5

Whenever you give an algorithm, also argue for its running time and correctness. Always feel free to ask on MS Teams if you're stuck, need a clarification, or suspect a typo.

1. Inclusion-exclusion algorithms:

   (a) Solve the exercise discussed on October 29: Implement the inclusion-exclusion algorithm for counting perfect matchings and compare it against the formula from statistical physics on grids. Check that your algorithm gives the correct output for all grids it can handle.

   (b) Implement the inclusion-exclusion based algorithm for Hamiltonian cycles from the lecture and compare its running time against the DP algorithm: What is the maximum number of vertices that can be handled with each of the algorithms?

   (c) Show how to extend the inclusion-exclusion based algorithm for Hamiltonian cycles to the TSP problem with polynomially bounded integer edge-weights. (Hint: Introduce a variable x and encode an edge-weight W as x^W. Compute a sum over all Hamiltonian cycles, each weighted by the product of edge-weights occurring on it. To this end, rather than counting walks, sum over the walks such that each walk $(v_1, \ldots, v_k, v_1)$ is weighted by the product of the edge-weights occurring in it. Show that this sum can also be computed via matrix multiplication.)

2. On *bipartite* graphs, perfect matchings can be counted faster than in general graphs: Assume we are given a bipartite graph $G = (V, E)$ with bipartition $V = L \cup R$ such that $|L| = |R| = \frac{n}{2}$ for even n.

   (a) Let Ω denote the set of all edge-subsets $S \subseteq E$ such that every vertex $v \in L$ is contained in exactly one edge $e \in S$. (Note that there is no restriction on the vertices in R.) Define a family of bad sets $B_1, \ldots, B_t$ for some $t \in \mathbb{N}$ such that $S \in \Omega$ is a perfect matching iff $S \in \Omega \setminus (B_1 \cup \ldots \cup B_t)$.

   (b) Using the above observation together with the inclusion-exclusion principle, prove *Ryser's formula*:

   $$\#\text{PerfMatch}(G) = \sum_{X \subseteq R} (-1)^{|S|} \prod_{v \in L} d(X, v),$$

   where $\#\text{PerfMatch}(G)$ denotes the number of perfect matchings in G, and $d(X, v)$ denotes the number of edges incident with v in the graph $G - X$. (Hint: For fixed $X \subseteq R$, show that the number $\prod_{v \in L} d(X, v)$ counts those edge-sets S in which (i) every $v \in L$ has exactly one incident edge and (ii) the vertices hit by S in R do not contain any element from X. Then apply the inclusion-exclusion principle.)

   (c) Derive an $O^*(2^{n/2})$ time algorithm for computing $\#\text{PerfMatch}(G)$ for bipartite G from the above formula.

3. Assume you have a SAT solver that, given a formula φ as input, only outputs whether φ is satisfiable or not. The solver does not output a satisfying assignment for φ. Given a formula φ with n variables, show how to compute a satisfying assignment (if it exists) with at most n calls to the SAT solver. (Hint: For every possible assignment $a \in \{0, 1\}$ to the first variable x_1, use the solver to determine whether the formula can still be satisfied after setting x_1 to a. Use this approach inductively.)