Submission instructions as in previous homeworks.

1 (100 PTS.) Where to park?

Urbana high school has (only) n students, and every student has a car (sadly, only one car). The parking lot S has $m \geq n$ spots where one can park their car. The ith car c_i, has exactly two distinct spots $s_i, s'_i \in S$ where it is allowed to park, for $i = 1, \ldots, n$. Given these spots, design an efficient algorithm that decides if there is a way to park all the cars (no two cars park in the same spot).

1.A. (10 PTS.) Consider a graph G with $2n$ nodes, where for every car c_i there are two nodes $\langle i/s_i \rangle$ and $\langle i/s'_i \rangle$. For $\gamma \in \{s_i, s'_i\}$ and $\delta \in \{s_j, s'_j\}$, add a directed edge from $\langle i/\gamma \rangle$ to $\langle j/\delta \rangle$, if parking the ith car at γ implies that the jth car must be parked in the slot δ because the other parking spot of the jth car is γ. Let m denote the number of edges of G. What is the maximum value of m (in the worst case)? What is the running time of your algorithm to compute this graph?

1.B. (10 PTS.) If there is a path in G from $\langle i/\gamma \rangle$ to $\langle j/\delta \rangle$, then $c_i = \gamma$ ***forces*** $c_j = \delta$. Prove that if $c_i = s_i$ forces $c_j = s_j$ then $c_j = s'_j$ forces $c_i = s'_i$.

1.C. (20 PTS.) Prove that if $\langle i/s_i \rangle$ and $\langle i/s'_i \rangle$ are in the same strong connected component of G, then there is no legal way to park the cars.

1.D. (20 PTS.) Assume that there is a legal solution, and consider a SCC Y of G involving cars, say, $c_1, \ldots, c_t$ in G; that is, Y is a set of vertices of the form $\langle 1/x_1 \rangle, \ldots, \langle t/x_t \rangle$. Prove that $\langle 1/x'_1 \rangle, \ldots, \langle t/x'_t \rangle$ form their own strong connected component $\overline{Y}$ in G ($\overline{Y}$ is the ***reflection*** of Y).

1.E. (10 PTS.) Prove that if X is a SCC of G that is a sink in the meta graph $\mathsf{G}^{\mathrm{SCC}}$, then $\overline{X}$ is a source in the meta graph $\mathsf{G}^{\mathrm{SCC}}$.

1.F. (30 PTS.) Consider the algorithm that takes the sink X of the meta-graph $\mathsf{G}^{\mathrm{SCC}}$, use the associated slots as specified by the nodes in X, remove the vertices of X from G and the vertices of $\overline{X}$ from G, and repeating this process on the remaining graph. Prove that this algorithm generates a legal parking of the cars if it exits (or otherwise outputs that no such parking exists [describe how to modify the algorithm to check for this]). Describe how to implement this algorithm efficiently.
What is the running time of your algorithm in the worst case as a function of n and m.

2 (100 PTS.) Revisit.

2.A. (20 PTS.) Consider a DAG G with n vertices and m edges. Assume that s is a source in G (a ***source*** is a vertex that has only outgoing edges). Describe how to compute in linear time a set of new edges such that s is the only source in the resulting graph (which still has to be a DAG). How many edges does your algorithm add (the fewer, the better)?

2.B. (20 PTS.) Assume G is a DAG with a source vertex s. Some of the vertices of G are marked as being ***important***. Show an algorithm that in linear time computes all the vertices that can be reached from s via a path that goes through at least τ important vertices, where τ is a prespecified parameter.

2.C. (30 PTS.) An edge e of G has the length $\ell(\mathsf{e})$ assigned to it (it can be potentially a negative number, not that it matters). Show an algorithm (faster is better) that computes for all the vertices v in G the length of the ***longest*** path from s to v.

2.D. (30 PTS.) Using the above, describe how to compute, in linear time, a path that visits the maximum number of vertices of the DAG G (the path is allowed to start at any vertex and end at any vertex of G).