34 (100 PTS.) Multiple Choice Practice

The following problems are multiple choice. Each question has a *single* answer. Make the best possible choice if multiple options seem correct to you — for algorithms, faster is always better.

Note that this homework will NOT be collected or graded, however we will release the solutions. More examples of multiple choice questions like these can be found in the study section materials for the final exam.

34.A. (3 points)
Given a DFA N and an NFA M with n and m states, respectively. Then there is a DFA M' that accepts the language $L(N) \setminus L(M)$.

(A) True, and the number of states of M' is at most $n2^m$.
(B) True, and the number of states of M' is at most nm.
(C) True, and the number of states of M' is at most 2^n2^m, and no other answer applies.
(D) False.
(E) True, and the number of states of M' is at most $(m+n)2^{(m+n)/2}$.

Solution:

A.

Applying the subset construction to the NFA M gives us a DFA $M^\dagger$ that recognizes the same language $L(M)$ and uses at most 2^m states. Applying the complement construction to $M^\dagger$ gives us a DFA $M^{\dagger\dagger}$ with the same number of states accepting the complement language. Applying the product construction to N and $M^{\dagger\dagger}$ gives us a DFA M' of size $n2^m$ that recognizes the intersection of $L(N)$ and $\neg L(M)$, which is $L(N)\setminus L(M)$.

34.B. (3 points)
Let $L_1, L_2 \subseteq \Sigma^*$ be context-free languages. Then the language $L_1 \cap L_2$ is always context-free.

(A) None of the other answers.
(B) False if the languages L_1 and L_2 are decidable, , and no other answer is correct.
(C) True only if the languages L_1 and L_2 are decidable, and no other answer is correct.
(D) True.
(E) False.

Solution:

E.

Consider the counterexample $L_1 = \{a^n b^n c^*\}$ and $L_2 = \{a^* b^n c^n\}$, such that the intersection $L \cap \{a^n b^n c^n\}$ which we used as an example of a non-context free language.

In answer B, the qualifications about decidability are vacuous. Every context free language is decidable, for example using the CYK parsing algorithm based on dynamic programming.

34.C. (3 points)
Given an undirected graph G with n vertices and m edges, and a number k, deciding if G has a spanning tree with maximum degree k is

(A) Can be done in polynomial time.
(B) Can be done in $O((n+m)\log n)$ time, and there is no faster algorithm.
(C) Can be done in $O(n \log n + m)$ time, and there is no faster algorithm.
(D) NP-Complete.
(E) Can be done in $O(n+m)$ time.

Solution:

D.

Deciding if a graph G has a spanning tree of maximum degree-k is NP-complete. To show this, we can construct a reduction from Hamiltonian Path to this problem. The basic insight is that a simple path corresponds to a tree of maximum degree $k = 2$. Specifically, if $(u_1, u_2, ..., u_k, u_1)$ is a Hamiltonian path, then the path $(u_1, u_2, ..., u_k)$ is a Spanning Tree with maximum degree 2.

Let G be an undirected graph for which we want to determine if a Hamiltonian path exists. Suppose $M(G', k)$ decides whether G' has a spanning tree of maximum degree k. The reduction does:

For each edge $e = (u, v)$ in G, construct a graph G_e where e is removed,
and new nodes u' and v' are added, connected to u and v respectively.
$G_e := G \backslash \{e\} \cup \{(u', u), (v, v')\}$
If $M(G_e, 2)$ returns True, return True (G has a hamiltonian path).
If we have tried each edge, return False (G does not have a hamiltonian path).

The u' and v' values here are added to ensure that if G_e has a spanning tree of maximum degree 2, it must correspond to a path $u', u, ..., v, v'$, since this is the only way to connect u' and v'. Hence $u, ..., v, u$ is a Hamiltonian cycle in G. Hence this problem is NP-Hard.

It is also easy to show this problem is NP, since if the answer is yes, then the spanning tree itself would serve as a certificate.

34.D. (2 points)
You are given a set $\mathcal{I} = \{I_1, I_2, \ldots, I_n\}$ of n weighted intervals on the real line. Consider the problem of computing a value $x \in \mathbb{R}$, that maximizes the total weight of the intervals of $\mathcal{I}$ containing x. This problem:

(A) Can be done in polynomial time.
(B) Undecidable.
(C) NP-Complete.
(D) Can be done in linear time.
(E) NP-Hard.

Solution:

A.

This can be solved in polynomial time, but it requires sorting, which requires at least $O(nlogn)$ time.

A solution is as follows: suppose all of the intervals are initially presented as (s_i, e_i, w_i), start end and weight respectively. Let A be the array containing all the edges, whether they are starts or ends, all together in one list. Each element is of the form $(s_j, w_j, \texttt{Start})$ or $(e_j, w_j, \texttt{End})$. Use a sorting algorithm like mergesort to sort all of these edges.

Initialize a cursor $c := A[0]$, the left most starting edge
Initialize a counter $acc := 0$
Initialize $best := 0$
Consider each edge (d, w, x) in sorted order in A, where $x \in \{\texttt{Start}, \texttt{End}\}$.
If $x = \texttt{Start}$, then any point $r \in [c, x]$ intersects intervals with total weight $acc + w$, including the interval that starts at d.
  $acc := acc + w$
If $x = \texttt{End}$, then any point $r \in [c, x]$ intersects intervals with total weight $acc - w$, since it no longer includes the interval ending at d.
  $acc := acc - w$
$c := d$
$best := max(best, acc)$
return $best$

34.E. (3 points)
Consider the problem of checking if a graph has k vertices that are all adjacent to each other. This problem can be solved in

(A) It is NP-Complete, so it can not be solved efficiently.
(B) Polynomial time.
(C) None of the other answers are correct.
(D) Maybe polynomial time – we do not know. Currently fastest algorithm known takes exponential time.

Solution:

D.

This problem is NP-complete. This problem describes looking for a k-clique. Every k-clique in graph G corresponds to a size-k independent set in the inverted graph $\overline{G}$. We saw a reduction from Independent Set to 3SAT, which we know from Cook-Levin theorem is NP-complete. The answer D better describes our state of knowledge about NP-complete problems than answer (A), since we do not yet know conclusively whether P=NP or not, hene cannot assert there is no polynomial time algorithm for this.

34.F. (2 points)

Consider a Turing machine (i.e., program) M that accepts an input $w \in \Sigma^*$ if and only if there is a CFG G such that $w \in L(G)$. Then the language of $L(M)$ is

(A) context-free.

(B) Σ^*.

(C) finite.

(D) undecidable.

(E) not well defined.

Solution:

B.

For any string w, we know that w is at least included in the context-free language Σ^*. Hence this property holds for all strings w, so therefore M must accept every input w. $L(M)$ is therefore equal to Σ^*.

A is also true, since Σ^* is also context free. B is a better answer since it more precisely describes $L(M)$.