G22.3520: Honors Analysis of Algorithms

Problem Set 6

Problem 1

Show that P is closed under the star operation (*Hint: Use dynamic programming.*) Recall that for a language L,

$$L^* = \{x_1 x_2 \ldots x_k \mid k \geq 0,\ x_i \in L\ \forall 1 \leq i \leq k\}$$

Solution: We show that **P** is closed under the star operation by dynamic programming. Let $L \in \mathbf{P}$, and let A be a polynomial-time algorithm that decides L. We construct a polynomial-time algorithm B that decides L^*, as follows. On input $w = w_1 \cdots w_n$, algorithm B constructs a table T such that $T[i,j] = 1$ if $w_i \cdots w_j \in L^*$, and $T[i,j] = 0$ otherwise. Details follow.

Algorithm B:

On input $w = w_1 \cdots w_n$,

```
1. If $w = \varepsilon$, output ACCEPT and halt.
2. Initialize $T[i,j] = 0$ for each $0 \leq i \leq j \leq n$.
3. For $i = 1$ to $n$: [1]
   (a) Run $M$ on $w_i$ to decide whether $w_i \in L$.
   (b) If $w_i \in L$, then set $T[i,i] = 1$.
4. For $\ell = 2$ to $n$: [2]
   For $i = 1$ to $n - \ell + 1$: [3]
   (a) Let $j = i + \ell - 1$. [4]
   (b) Run $M$ on $w_i \cdots w_j$ to decide whether $w_i \cdots w_j \in L$.
   (c) If $w_i \cdots w_j \in L$, then set $T[i,j] = 1$.
   (d) For $k = i$ to $j - 1$: [5]
         If $T[i,k] = 1$ and $T[k,j] = 1$, then set $T[i,j] = 1$.
5. Output ACCEPT if $T[1,n] = 1$; else output REJECT. [6]
```

[1] Test each substring of length 1

[2] ℓ is the length of the substring

[3] i is the start position of the substring

[4] j is the end position of the substring

[5] k is the split position

[6] $T[1,n] = 1$ if and only if $w = w_1 \cdots w_n \in L^*$.

It is not hard to see that Algorithm B correctly decides L^* provided that Algorithm A correctly decides L. Moreover, Algorithm B runs in $O(n^3)$ stages, and each stage takes polynomial-time as A runs in polynomial-time. Therefore, Algorithm B is a polynomial-time algorithm that decides L^*.

Problem 2

Show that these problems are NP-complete (you can assume that SAT and CLIQUE are NP-complete).

1. DOUBLE-SAT $= \{\langle\phi\rangle \mid \phi$ is a boolean formula that has at least two satisfying assignments$\}$.
2. HALF-CLIQUE $= \{\langle G\rangle \mid G$ is a graph with a clique of size at least $|G|/2\}$.

Solution: Clearly Double-SAT $\in$ **NP**, since a NTM can decide Double-SAT as follows: On input a Boolean formula $\varphi(x_1,\ldots,x_n)$, nondeterministically guess 2 assignments and verfify whether both satisfy φ. To show that Double-SAT is **NP-Complete**, we give a reduction from SAT to Double-SAT, as follows:

On input $\varphi(x_1,\ldots,x_n)$:

1. Introduce a new variable y.
2. Output formula $\varphi'(x_1,\ldots,x_n,y) = \varphi(x_1,\ldots,x_n) \wedge (y \vee \overline{y})$.

If $\varphi(x_1,\ldots,x_n) \in SAT$, then φ has at least 1 satisfying assignment, and therefore $\varphi'(x_1,\ldots,x_n,y)$ has at least 2 satisfying assignments as we can satisfy the new clause $(y \vee \overline{y})$ by assigning either $y = 1$ or $y = 0$ to the new variable y, so $\varphi'(x_1,\ldots,x_n,y) \in$ Double-SAT. On the other hand, if $\varphi(x_1,\ldots,x_n) \notin SAT$, then clearly $\varphi'(x_1,\ldots,x_n,y) = \varphi(x_1,\ldots,x_n) \wedge (y \vee \overline{y})$ has no satisfying assignment either, so $\varphi'(x_1,\ldots,x_n,y) \notin$ Double-SAT. Therefore, SAT $\leq_P$ Double-SAT, and hence Double-SAT is **NP-Complete**.

Solution: Clearly Half-CLIQUE $\in$ **NP**, since a NTM can decide Half-CLIQUE as follows: On input a graph G, nondeterministically choose at least $n/2$ nodes and verfiy whether they form a clique. To show that Half-CLIQUE is **NP-Complete**, we give a reduction from CLIQUE to Half-CLIQUE, as follows:

On input $\langle G, k\rangle$, where G is a graph on n vertices and k is an integer:

1. If $k = n/2$, then output $\langle G\rangle$.
2. If $k < n/2$, then construct a new graph G' by adding a complete graph with $n - 2k$ vertices and connecting them to all vertices in G, and output $\langle G'\rangle$.
3. If $k > n/2$, then construct a new graph G'' by adding $2k - n$ isolated vertices[7] to G, and output $\langle G''\rangle$.

[7]An isolated vertice is one which is not adjacent to any other vertex.

When $k = n/2$, it is clear that $\langle G, n/2\rangle \in$ CLIQUE if and only if $\langle G\rangle \in$ Half-CLIQUE.

When $k < n/2$, if G has a k-clique, then G' has a clique of size $k + (n - 2k) = (2n - 2k)/2$, and therefore $\langle G'\rangle \in$ Half-CLIQUE as G' is a graph with $2n - 2k$ vertices. Conversely, if $\langle G'\rangle \in$ Half-CLIQUE, that is, if G' has a clique of size $n - k = k + (n - 2k)$, then at most $n - 2k$ of the clique come from the $n - 2k$ new vertices. Therefore the remaining at least k vertices form a clique in G, and hence $\langle G, k\rangle \in$ CLIQUE.

When $k > n/2$, if G has a k-clique, then G'' has a clique size $k = 2k/2$, and therefore $\langle G'\rangle \in$ Half-CLIQUE as G'' is a graph with $n + 2k - n = 2k$ vertices. Conversely, if $\langle G''\rangle \in$ Half-CLIQUE, that is, if G'' has a clique of size k, then the clique does not contain any of the new vertices as they are isolated. Thus the clique is a k-clique of G, and hence $\langle G, k\rangle \in$ CLIQUE.

Problem 3

Let $\{x_1, x_2, \ldots, x_n\}$ be boolean variables. A literal is either a variable or its negation, i.e. x_i or $\overline{x}_i$. A clause is logical OR of one or more distinct literals. The size of a clause is the number of literals in it. A 2CNF formula ϕ is a collection of m clauses (possibly with repetition),

$$\phi = (C_1, C_2, \ldots, C_m)$$

where each C_i is of size at most two. Let MAX-2SAT be the following decision problem: Given a pair (ϕ, k) where ϕ is a 2CNF formula with n variables and m clauses, and k is a positive integer such that $k \leq m$, decide whether there exists an assignment to the n boolean variables that satisfies at least k clauses. Show that MAX-2SAT is NP-complete, by giving a polynomial time reduction from VERTEX COVER.

Recall that VERTEX COVER is a problem where, given an undirected graph $G = (V, E)$, with $|V| = n$, and given a positive integer $\ell \leq n$, one needs to decide whether G has a vertex cover of size at most ℓ. A vertex cover is a subset $V' \subseteq V$ such that for every edge in E, at least one of its endpoints is included in V'.

Hint: To every vertex in the graph, assign a boolean variable which is intended to be TRUE if and only if the vertex is included in the vertex cover. Add clauses of size two corresponding to the edges, and clauses of size one corresponding to the vertices. The clauses corresponding to edges may need to be repeated a number of times.

Solution: It appears as Problem-2 here: `http://www.cs.nyu.edu/web/Academic/Graduate/exams/sample/algs_f07_ans.pdf`

Problem 4

Show that these problems are NP-complete:

1. Problem 5 in `http://cs.nyu.edu/web/Academic/Graduate/exams/sample/algs_f09.pdf`
2. Problem 6 in `http://cs.nyu.edu/web/Academic/Graduate/exams/sample/algs_f08.pdf`

Solution:

1. The NP algorithm for AHC is as follows: we first check that the graph has an even number of vertices n, then nondeterministically select a sequence of n vertices and check whether this sequence forms a Hamiltonian cycle with edges colored alternately, taking linear time.

   For NP-completeness, we reduce DHC to AHC. Given a graph G, we construct the graph G' by replacing every vertex v in G with a pair of vertices v_1, v_2 with an edge going from v_1 to v_2, which we color blue. Additionally, for every edge in G going from u to v, we add in G' an edge going from u_2 to v_1, which we color red. This reduction can be easily computed in time $O(m+n)$. Now, suppose G has a Hamiltonian cycle $v^1, \ldots, v^n$. Then, in G', $v_1^1, v_2^1, v_1^2, v_2^2, \ldots, v_1^n, v_2^n$ is a Hamiltonian cycle with edges $v_1^i \to v_2^i$ colored blue and edges $v_2^i \to v_1^{i+1 \bmod n}$ colored red, proving completeness. As for soundness, suppose G' has a Hamiltonian cycle with alternating edge colors. Since all blue edges are of the form $v_1 \to v_2$, this cycle must be of the form $v_1^{i_1}, v_2^{i_1}, v_1^{i_2}, v_2^{i_2}, \ldots, v_1^{i_n}, v_2^{i_n}$, implying that $v^{i_1}, \ldots, v^{i_n}$ is a Hamiltonian cycle in G. $\square$

2. The NP algorithm for TRIANGLE-FREE-SET (TFS) is as follows: we nondeterministically select a set of vertices and check whether this set is of size $\geq k$ and is triangle-free, taking $O(n^3)$ time to check every triplet.

   For NP-completeness, we reduce INDEPENDENT-SET to TFS. Given an input $\langle G, k\rangle$ with n vertices, we construct the graph G' by adding a set S of $n-k+1$ vertices to G, with edges between all the vertices in S and all the vertices in G (but no edges between vertices in S). We then ask whether G' has a triangle-free set of size $n+1$, reducing to the problem TFS$\langle G', n+1\rangle$. This reduction can be computed in time $O(n^2)$. For completeness, suppose G has an independent set A of size k. We then simply consider $A \cup S$ in G'. Both A and S are independent sets in G', so $A \cup S$ is therefore bipartite and triangle-free of size $k + n - k + 1 = n + 1$. For soundness, if G' has a triangle-free set of size $n+1$, then by Pigeonhole Principle one of the vertices in T must also be in S. Since the vertices in S are connected to all vertices in G, for T to be triangle-free, the set $T \setminus S \subseteq G$ must be independent of size $\geq n + 1 - (n - k + 1) = k$. $\square$

Problem 5

Solve Problem 2 in `http://cs.nyu.edu/web/Academic/Graduate/exams/sample/algs_f08.pdf`

Solution:

1. We use the probabilistic method. Specifically, we pick an assignment uniformly at random, setting each variable to 0 or 1 with probability $1/2$. Let M be the random variable representing the number of clauses satisfied by our assignment, and let M_i be the indicator variable for the event that clause C_i is satisfied. We clearly have $M = \sum_{i=1}^m M_i$ and $\mathbb{E}[M] = \sum_{i=1}^m \mathbb{E}[M_i]$ by linearity of expectation. The probability that all literals in C_i are set to 0 is $1/4$ if C_i has two literals, and $1/8$ if C_i has three literals, so the probability that clause C_i is satisfied is either $3/4$ or $7/8 \geq 3/4$. We thus have $\mathbb{E}[M] \geq 3m/4$, and since we have $\mathbb{E}[M] \leq \max M$, we get $\max M \geq 3m/4$, showing that an assignment satisfying $\geq 3m/4$ clauses must exist. $\square$

2. We claim that $c = 5/6$—that is, for all instances of the specified form, there exists an assignment satisfying $5m/6$ of the clauses (assume for simplicity that m is divisible by 6). By the analysis done in 1, we have that

$$\max M \geq \mathbb{E}[M] \geq \left(\frac{2m}{3}\right)\left(\frac{7}{8}\right) + \left(\frac{m}{3}\right)\left(\frac{3}{4}\right) = \frac{7m}{12} + \frac{m}{4} = \frac{5m}{6}$$

as desired. $\square$