CSC 364H **Solutions to Problem Set 3** **Summer 99**

Total marks: 100

[55] 1. It is clear what the guess-and-check algorithm is for the *NP* problems, so we omit it, but you need it for full marks. Each problem is worth 5 marks.

(a) SMOOTH SAT is *NP*-complete. We show that SAT $\leq_p$ SMOOTH SAT. Let $S = \{C_1, \ldots, C_m\}$ be a set of clauses. If C_i contains both negated and unnegated variables, then we replace each negated variable $\bar{P}$ be a *new* variable Q, and add the two clauses $\{P, Q\}, \{\bar{P}, \bar{Q}\}$ which together assert that $Q \equiv \bar{P}$.

(b) LIS is *NP*-complete. We show that IS $\leq_p$ LIS. Let $G = (V, E), B$ be an instant of IS. We define the corresponding instance $G' = (V', E'), B'$ of LIS as follows:

If $|V|$ is even, then let $B' = B$. Otherwise, add a vertex v_0 to V, i.e. $V \longleftarrow V \cup \{v_0\}$. Now, let $x = ||V| - 2B'|$. If $x = 0$, then $B' = |V|/2$, and we already have an instance of LIS. If $x \neq 0$, then we have two cases:

case I. $B' < |V|/2$. Then, add $|V| - 2B'$ new nodes.

case II. $B > |V|/2$. Then, add $2B' - |V|$ new nodes, and add new edges connecting every new node to every other node.

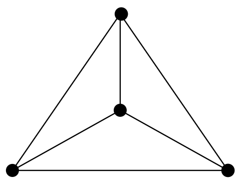
(c) DEGREE 2 IS is in P. Find all connected components — each of them must be a cycle. Each such connected component has an independent set of size $\lfloor n/2 \rfloor$, where n is the number of nodes in the component.

(d) UNARY PARTITION is in P. It has a polytime solution, using dynamic programming. Let

$$R(j,k) = \begin{cases} T & \text{if there } \exists S \subseteq \{1, \ldots, j\} \text{ s.t. } \sum_{i \in S} w_i = k \\ F & \text{otherwise} \end{cases}$$

where $0 \leq j \leq d$, $0 \leq k \leq W/2$, and $W = \sum_{i=1}^{d} w_i$. Note that if W is odd then there is no solution. The answer is "YES" iff $R(d, 1/2W) = T$. The time is $O(dW) = O(n^2)$, where $n =$ total length of input.

(e) DEGREE 3 CLIQUE is in P. In a graph of degree 3 every clique must be of the form:



Since there are $\binom{n}{4}$ subsets of 4 nodes, we can check them all in polytime.

(f) HP2PS is *NP*-complete. We show that PARTITION $\leq_p$ HP2PS. Let $d, w_1, \ldots, w_d$ be an instance of PARTITION. We transform it to the instance $w_1, \ldots, w_d, w'_1, \ldots, w'_d, T$ of HP2PS, where $T = 1/2W$ if W is even, and $T = 0$ otherwise, and $W = \sum_{i=1}^{d} w_i$.

(g) LARGE CYCLE is *NP*-complete. HC can be reduced to it by asking: does G have a cycle with $|V|$ (or more) distinct nodes.

(h) SMALL CYCLE in in P. For each pair of nodes (and there are $\binom{n}{2}$ pairs) find the shortest path between them (using Floyd's algorithm, for example), and then take the min.

(i) 4-COLOUR is *NP*-complete. 3-COLOUR can be reduced to 4-COLOUR by adding a new vertex, and connecting all other vertices to it by an edge.

(j) TRIPARTITE is *NP*-complete. Show that 3-COLOUR $\leq_p$ TRIPARTITE. Given a graph $G = (V, E)$, G is three colourable iff V can be partitioned into three disjoint subsets, where the vertices in each subset get the same colour, and hence are not connected by an edge in E. Thus, G is three colourable iff V can be partitioned into three independent subsets.

(k) s-tHP in *NP*-complete. Show that HP $\leq_p$ s-tHP. Given a graph G add two new vertices ("handles") s and t, and connect each of them to every vertex in G.

[10] 2. Here is the idea for the reduction:

Input: $I = \langle d, w_1, \dots, w_d \rangle$.

If $I \notin$ PARTITION, output "NO".

Otherwise, we want to find an S so that $\sum_{i \in S} w_i = \sum_{i \notin S} w_i$.

Initialize: $S \longleftarrow \{1\}$

Ask if $\langle d-1, w_1 + w_2, \dots, w_d \rangle \in$ PARTITION. If so, then w_1 and w_2 go together, and hence

$S \longleftarrow S \cup \{2\}$
$I \longleftarrow \langle d-1, w_1 + w_2, w_3, \dots, w_d \rangle$ and repeat with the new S and I.

If not, then w_1 and w_2 must be separated, and hence

S remains unchanged
$I \longleftarrow \langle d-1, |w_1 - w_2|, w_3, \dots, w_d \rangle$ and repeat with the new S and I.

[5] 3. (a) We cannot use the simpler algorithm because the TM M might never halt on some s_i.

[5] (b) This exercise simply emphasizes that *Turing-recognizbale* is what we call *semi-decidable* in class.

[15] (c) **union:** suppose that L_1 and L_2 are deciable. Then, $L_1 = L(M_1)$ and $L_2 = L(M_2)$ where M_1 and M_2 halt on all inputs. Design M as follows: on input x, M simulates M_1, and accepts if M_1 accepts. If M_1 rejects, then M simulates M_2, and accepts if M_2 accepts, and rejects otherwise.

complementation: suppose that L is decidable. Then $L = L(M)$ where M halts on all inputs. Design M' as follows: M' is the same as M except when M is about to enter q_{accept}, then M' enter q_{reject} and viceversa. Thus, $\overline{L} = L(M')$, where M' always halts, and so $\overline{L}$ is decidable.

intersection: since $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$, we are done by the above two cases.

[10] 4. Let Σ be a finite alphabet, and let $\{s_1, s_2, s_3, \dots\}$ be the list of all strings over Σ in lexicographic order. Let $L \subseteq \Sigma^*$ be a language. If $L = \emptyset$, then L is decidable and we are done. Otherwise:

[$\Longleftarrow$] Suppse E enumerates L in lexicographic order. We design a TM M such that $L = L(M)$, and M halts on all inputs. On input $x = s_i$, M simulates E, until one of the following happens: E outputs x (in which case M accepts), or E outputs some s_j such that $j > k$, in which case M rejects.

[$\Longrightarrow$] If L is decidable, then $L = L(M)$ for some TM M which halts on every input. We design E as follows: E simulates M on each s_i in order (i.e. for $i = 1, 2, 3, \dots$), and outputs s_i only if M accepts s_i.