MCGILL UNIVERSITY – COMP 360

TEST 1 SOLUTION

Question 1 [10pt]

(a)[1pt] True

(b)[2pt] A problem P is polytime many-one reducible to a problem Q if there is a polytime computable function f (the transducer) so that for all x:

$$x \in P \qquad \text{if and only if} \qquad f(x) \in Q$$

(c)[2pt] Let

$$A = (\neg x_1 \vee x_2) \wedge (x_1 \wedge \neg x_2)$$
$$B = \neg x_1$$
$$C = (\neg x_1 \vee x_2) \qquad (i.e., C = B \vee x_2)$$
$$D = \neg x_2$$
$$E = x_1 \wedge \neg x_2 \qquad (i.e., E = x_1 \wedge D)$$

So $A = C \wedge E$. The new variables are:

$$q_A, q_B, q_C, q_D, q_E, \quad \text{and two dummy variables: } u, v$$

First we obtain the following clauses:

$$q_B \vee x_1, \quad \neg q_B \vee \neg x_1$$
$$\neg q_C \vee q_B \vee x_2, \quad \neg q_B \vee q_C, \quad \neg x_2 \vee q_C$$
$$q_D \vee x_2, \quad \neg q_D \vee \neg x_2$$
$$\neg q_E \vee x_1, \quad \neg q_E \vee q_D, \quad \neg x_1 \vee \neg q_D \vee q_E$$
$$\neg q_A \vee q_C, \quad \neg q_Q \vee q_E, \quad \neg q_C \vee \neg q_E \vee q_A$$
$$q_A$$

Some of these clauses have size 1 or 2, so we use the dummy variables to make their size 3. The output clauses are:

$$q_B \vee x_1 \vee u, \quad q_B \vee x_1 \vee \neg u, \quad \neg q_B \vee \neg x_1 \vee u, \quad \neg q_B \vee \neg x_1 \vee \neg u$$
$$\neg q_C \vee q_B \vee x_2, \quad \neg q_B \vee q_C \vee u, \quad \neg q_B \vee q_C \vee \neg u, \quad \neg x_2 \vee q_C \vee u, \quad \neg x_2 \vee q_C \vee \neg u$$
$$q_D \vee x_2 \vee u, \quad q_D \vee x_2 \vee \neg u, \quad \neg q_D \vee \neg x_2 \vee u, \quad \neg q_D \vee \neg x_2 \vee \neg u$$
$$\neg q_E \vee x_1 \vee u, \quad \neg q_E \vee x_1 \vee \neg u, \quad \neg q_E \vee q_D \vee u, \quad \neg q_E \vee q_D \vee \neg u, \quad \neg x_1 \vee \neg q_D \vee q_E$$
$$\neg q_A \vee q_C \vee u, \quad \neg q_A \vee q_C \vee \neg u, \quad \neg q_Q \vee q_E \vee u, \quad \neg q_Q \vee q_E \vee \neg u, \quad \neg q_C \vee \neg q_E \vee q_A$$
$$q_A \vee u \vee v, \quad q_A \vee u \vee \neg v, \quad q_A \vee \neg u \vee v, \quad q_A \vee \neg u \vee \neg v$$

(d)[1pt] Prim's algorithm is for the Minimum Spanning Tree problem.

(e)[2pt] Short the items in non-increasing order of the ration value/weight, i.e., so that

$$v_1/w_1 \geq v_2/w_2 \geq \ldots \geq v_n/w_n$$

Let k be the largest index so that

$$w_1 + w_2 + \ldots + w_k \leq W$$

If $k = n$, or $k < n$ and $v_1 + v_2 + \ldots + v_k > v_{k+1}$, then output the first k items. Otherwise output item $k + 1$.

(f)[2pt] Short the edges in non-decreasing order of cost. Keep adding edges in this order to the solution as long as they don't create cycle.

Question 2 [10pt]

(a)[5pt] On input (G, k), the output is the following set of requests and k: There is one request denoted R_v for each vertex v of G. The request R_v consists of all intervals of the form $[a_{uv}, a_{uv} + 1]$ for all u such that (u, v) is an edge of G. here a_{uv} is an integer such that $a_{uv} = avu$ and $a_{uv} \neq a_{u'v'}$ if the set $\{u', v'\}$ is different from the set $\{u, v\}$. Here we assume that vertices of G are numbered $1, 2, \ldots, n$. Then we can take a_{uv} to be

$$a_{uv} = (u + v)^2 + uv$$

It is clear that if $a_{uv} = a_{u'v'}$ then we must have both

$$u + v = u' + v' \qquad \text{and} \qquad uv = u'v'$$

Hence it must be that $\{u, v\} = \{u', v'\}$.

Formally, define

$$R_v = \{[a_{uv}, a_{uv} + 1] \ : \ \text{for all } u \text{ such that } (u, v) \notin E\}$$

for a_{uv} as above.

(b)[5pt] Observe that if (u, v) is not an edge of G then $R_u \cap R_v = \varnothing$ (so R_v and R_u can be scheduled together), and if (u, v) is an edge of G then $R_u \cap R_v$ is the set of one interval $[a_{uv}, a_{uv} + 1]$ (so R_v and R_u cannot be scheduled together). Thus K is an independent set of G if and only if the set of requests

$$\{R_v \ : \ \text{for } v \in K\}$$

can be scheduled together.

Consequently, G contains an independent set of size k if and only if at least k requests can be scheduled together.

Question 3 [10pt]

(a)[5pt] Sort the tasks in non-decreasing order of the parameters c_i, and schedule them in this order.
(b)[5pt] Suppose the tasks are sorted in the non-decreasing order of c_i, as output by the algorithm:

$$c_1 \leq c_2 \leq \ldots \leq c_n$$

We show that this is an optimal schedule.

An inversion in a schedule S is defined to be a pair of tasks (i, j) so that $c_i > c_j$ but i is schedule after j. Thus the output schedule does not contain any inversion.

First we show that an optimal schedule cannot contain any inversion. To see this, let $\mathcal{O}$ be any optimal schedule, and let C denote its total cost. Suppose for a contradiction that $\mathcal{O}$ contains an inversion (i, j). Let $\mathcal{O}'$ be the schedule obtained from $\mathcal{O}$ by swapping these two tasks. Suppose that in $\mathcal{O}$ i is executed at time t and j is executed at time s, where $t > s$. Then the total cost of $\mathcal{O}'$ is

$$C - (tc_i + sc_j) + (tc_j + sc_i) = C - (t - s)(c_i - c_j)$$

The total cost of $\mathcal{O}'$ is less than C because $t - s > 0$ and $c_i - c_j > 0$. This contradicts the fact that $\mathcal{O}$ is an optimal schedule.

Now to show that our schedule S is optimal it remains to show that any two schedules with no inversion have the same total cost. Note that any such two schedules differ only in the order of tasks that have the same parameter c_i. The same calculation as above showing that swapping two tasks that have the same parameters does not change the total cost of the schedule. Thus S has the same total cost as any optimal schedule; therefore it is an optimal schedule.