McGill University COMP360 Winter 2011 **Instructor: Phuong Nguyen**

Assignment 4 Solution

Question 1 (10pt) Consider the following variant of the problem Interval Scheduling, which we call MIS (M for multiple). In this problem, each request consists of a *set* of intervals (instead of just one interval as in Interval Scheduling). There is a single processor that can process only one request at a time. Thus, if we accept a request R then we cannot accept any other request that contains some interval which overlaps one of the intervals in R. For example, suppose that there are 4 requests:

$$R_1 = \{(1,3),(5,6),(7,8)\}$$
$$R_2 = \{(2,4)\}$$
$$R_3 = \{(4,6),(8,9)\}$$
$$R_4 = \{(3,4),(8,9)\}$$

Then the maximum number of requests that can be scheduled together is 2, e.g.: R_1 and R_4, or R_2 and R_3. The MIS problem is as follows.

Input:

- A set of n requests $R_1, R_2, \dots, R_n$, each request R_i is specified by a list of intervals

$$R_i = (s^i_1, f^i_1), (s^i_2, f^i_2), \dots, (s^i_{m_i}, f^i_{m_i})$$

  where all s^i_j, f^i_j are nonnegative integers.

- An integer k

(Representation of numbers are not very important here, for example, they are written in unary.)

Output: Accept if we can schedule at least k requests together.

Show that the MIS problem is NP-complete, by giving a nondeterministic polytime algorithm for it, and many-one reducing CLIQUE to it. Give formal proof of correctness for your reduction.

Solution

Specify the problem: We reduce CLIQUE to MIS.

The reduction: The reduction is as follows. 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 NOT 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 $1,2,\dots,n$. 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.

Correctness of the reduction: Observe that if (u,v) is 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 not 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 cant be scheduled together). Thus K is a clique of G if and only if the set of requests

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

can be scheduled together.

Consequently, G contains a clique of size k if and only if at least k requests can be scheduled together.

Question 2 (4pt) Consider the following greedy algorithm that attempts to solve the Knapsack problem (see Assignment 3). Informally, the idea is to always select elements with highest possible ratio value/weights. Then output Yes if the selected elements have total value at least V.

(a) (2pt) Give pseudo-code for the algorithm.

(b) (2pt) Exhibit an example where the algorithm fails to solve Knapsack. Clearly describe why the algorithm fails on your example.

Solution

(a) The algorithm:

```
1. sort the items so that $v_1/w_1 \geq v_2/w_2 \geq \ldots \geq v_n/w_n$
2. $S = 0$ % total weight so far
3. $T = 0$ % total value so far
4. for $i = 1$ to $n$ do
5.     if $w_i \leq W - S$ do
6.         $T = T + v_i$
7.         $S = S + w_i$
8.     end if
9. end form
10. if $T \geq V$ output YES
11. else output NO
```

(b) Example where the above algorithm fails: The set consists of two items:

$$w_1 = 1, v_1 = 2, \qquad w_2 = 10, v_2 = 10$$

and $W = 10, V = 10$. The answer is YES, because we can pack just item 2.

However the algorithm outputs NO, because $v_1/w_1 > v_2/w_2$, and the algorithm selects the first item, and then there is no room for the second item. Thus when the for-loop terminates $S = 1, T = 2$, and the algorithm outputs NO.

Question 3 (6pt) In class we discussed an greedy algorithm for Interval Scheduling that works by always selecting requests that finish as early as possible. Another greedy algorithm that also produces an optimal solution is to always select requests that start as late as possible.

(a) (2pt) Give pseudo-code for the algorithm.

(b) (4pt) Proof that the algorithm always output an optimal schedule.

Solution

(a) The algorithm

```
1. sort the requests so that $s_1 \geq s_2 \geq s_3 \geq \ldots \geq s_n$
2. $s = f_1$ % current start time
3. for $i = 1$ to $n$ do
4.     if $f_i \leq s$ do
5.         add $i$ to the schedule
6.         $s = s_i$
7.     end if
8. end for
```

(b) Proof of correctness.

Let O be an optimal schedule,

$$O = i_1, i_2, \ldots, i_k$$

and let the output of the algorithm be

$$j_1, j_2, \ldots, j_\ell$$

We need to show that $\ell = k$. We already know by the optimality of O that $k \geq \ell$. We will show that $\ell \geq k$.

Suppose that the requests have been sorted as in the algorithm, that is

$$s_1 \geq s_2 \geq s_3 \geq \ldots \geq s_n$$

We will prove by induction on t that $s_{j_t} \geq s_{i_t}$ for $1 \leq t \leq \ell$. It will follow that $\ell = k$, because otherwise $k \geq \ell + 1$, and by the time the for-loop terminates, $s = s_{j_\ell}$ and there is still at least one more request, namely $i_{\ell+1}$ (where $i_{\ell+1} > j_\ell$), such that

$$f_{i_{\ell+1}} \leq s$$

hence the for-loop must have continued.

Now we prove the claim. The base case is obvious because $j_1 = 1$. For the induction step, suppose that $s_{j_t} \geq s_{i_t}$ for some $t < \ell$. We need to show that $s_{j_{t+1}} \geq s_{i_{t+1}}$. At step $t + 1$, the algorithm chooses j_{t+1} because it is the request with maximal start time among all requests that start before request j_t. Because i_{t+1} also starts before j_t, we have $s_{j_{t+1}} \geq s_{i_{t+1}}$. QED.