Due: Wednesday, March 5, beginning of lecture

NOTE: Each problem set only counts 5% of your mark, but it is important to do your own work. These assignments will be followed by term tests, each worth 15% of your final grade. You may consult with others concerning the general approach for solving problems on assignments, but you must write up all solutions entirely on your own. Anything else is plagiarism, and is subject to the University’s Code of Behavior.

1. Let \( \mathcal{F} = (G, c, s, t) \) be a flow network with \( G = (V, E) \) and \( c \) being integer valued (i.e. all capacities are non-negative integers). Let \( f \) be a maximum flow in \( \mathcal{F} \). Now suppose that the capacity of some edge \( e \in E \) is increased by 1. Using \( f \), give an \( O(|V| + |E|) \) time algorithm for computing a maximum flow \( f' \) in the modified network. Justify why your algorithm correctly produces the updated max flow.

SOLUTION. Given \( \mathcal{F} \), increase the capacity of edge \( e \) by 1 to form the modified flow network \( \mathcal{F}' = (G, c', s, t) \). Using the optimal flow \( f \) for \( \mathcal{F} \), compute the residual graph \( G_f \) with respect to \( c' \). That is, the residual capacity \( c_f'(u, v) = c'(u, v) - f(u, v) \). Now if there is no augmenting path in this residual graph, then the maximum flow value in the modified network is still equal to \( |f| \). If there is an augmenting path \( \pi \) in this residual graph then its residual capacity \( c_f(\pi) = 1 \) and \( |f'| = |f| + 1 \). It should be clear the \( |f| \leq |f'| \leq |f| + 1 \) and a flow \( f \) in the original network is a flow in the modified network. The residual graph with respect to the modified network has an augmenting path iff \( f \) is not optimal for the modified network. If it is not optimal then the residual capacity of any augmenting path can be at most one or else there would have been an augmenting path in the original network contradicting the optimality of \( f \) for the original network.

2. Any string \( x \in \{0, 1\}^* \) can be interpreted as the binary representation (ignoring leading zeros) of a non-negative integer, denoted \( (x)_2 \). (Let the empty string \( \lambda \) represent the integer 0.) Specify a Turing machine which on input \( x \in \{0, 1\}^* \) computes a \( y \in \{0, 1\}^* \) such that \( (y)_2 = (x)_2 + 1 \).

SOLUTION. Informally, the Turing machine head moves to the right of the input (i.e. to its low order bit) and then moves left changing any ‘1’ to a ‘0’ until it hits a ‘0’ which it changes to a ‘1’ and then halts.

3. Define the integer maximum flow decision problem MFLOWD as the language
\( \{< \mathcal{F}, B > | \mathcal{F} \text{ is an integer capacity flow network and } \mathcal{F} \text{ has max flow value } \geq B \} \).

Let MFLOWS be the search/optimization problem (for flow networks with integer capacities) which computes a maximum flow \( f : E \rightarrow N \) where \( N = \{0, 1, 2, \ldots \} \). Say all integer values are represented in binary.

Show that MFLOWS \( \xrightarrow{p} \) MFLOWD.

Note: You should NOT solve the MFLOWS problem by the Ford Fulkerson algorithm.
SOLUTION. As we did for the GKS \(\rightarrow\) GKD reduction in the notes, we use binary search and the algorithm for MFLOWD to compute the optimal flow value \(V = |f|\) for some optimal flow \(f\). Now we want to compute such a flow \(f\); that is, we want to compute a flow \(f(e)\) on each edge so that \(f\) is optimal. For each edge \(e\), we will use a binary search to determine such a flow \(f(e)\). More precisely, using MFLOWS with \(B = V\), we try capacity \(c'(e) = \lceil c(e)/2 \rceil\) and if we can still realize a flow with value \(V\), we try \(c'(e) = \lceil c(e)/4 \rceil\) else we try \(c'(e) = \lceil 3c(e)/4 \rceil\). In this manner we obtain a flow value \(f(e)\) for each edge realizing the optimal flow value \(V\).

4. Prove \(\text{PARTITION} \leq_p \text{JSCHEDD}\) where JSCHEDD is the decision problem corresponding to the unit profit job scheduling problem where each job has a relasase time, a processing time and a deadline.

SOLUTION. The “natural” decision problem would be to determine if a set of jobs has profit (= number of jobs scheduled) \(\geq B\) for some parameter bound \(B\). But simpler yet, for this transformation we only need to determine whether or not all jobs can be scheduled; that is, we can set \(B = \text{number of jobs in the input instance}\).

Now here is the transformation. Let \(A = (a_1, \ldots, a_m)\) be an input instance for the \(\text{PARTITION}\) problem. For each item \(a_j\), we create a job \(J_j = (r_j, p_j, d_j) = (0, a_j, T + 1)\), where \(T = (\sum_{j:1 \leq j \leq m} a_j)\). Note if \(T\) is not even (and hence the input instance to \(\text{PARTITION}\) is not a YES instance) then we can transform the instance to any JSCHEDD instance for which all jobs cannot be scheduled (e.g. take two identical jobs which are intervals and can only be scheduled in one place). We create one additional job \(J' = (T/2, 1, T/2 + 1)\). Let us call this transformation \(f\) where the transformed instance is \(f(A) = \{J_1, \ldots, J_m, J'_m + 1\}\). Note that \(J'\) can only be scheduled at time \(T/2\).

We claim the following:

- The transformation \(f\) is a polynomial time computable transformation. This should be obvious.
- If \(A \in \text{PARTITION}\), then \(f(A) \in \text{JSCHEDD}\). For suppose that \(S \subset \{1, \ldots, m\}\) is such that \(\sum_{j \in S} a_j = T/2\), then all of the items with \(j \in S\) can be scheduled in the time \([0,T/2]\) and the remaining items can be scheduled in the time \([T/2 + 1, T + 1]\) and \(f(A) \in \text{JSCHEDD}\)
- If \(f(A) \in \text{JSCHEDD}\), then \(A \in \text{PARTITION}\). If every job in \(f(A)\) has been scheduled then \(J'\) has been scheduled in slot \([T/2, T/2 + 1]\) and the remaining jobs cannot overlap this slot. The \(m\) jobs \(J_j, 1 \leq j \leq m\), need a total of \(T\) time units to be scheduled and after scheduling \(J'\) that is exactly the time remaining. Hence the jobs \(J_j(1 \leq j \leq m)\) must be scheduled in the time \([0,T/2]\) and \([T/2+1, T+1]\) and there can be no wasted time. Thus some subset \(S\) must exist where all jobs \(J_j\) with \(j \in S\) must be scheduled in \([0,T/2]\) and the remaining jobs must be scheduled in \([T/2 + 1, T + 1]\). This subset \(S\) verifies the fact that \(A \in \text{PARTITION}\).