CSE 202 Homework 4: Solutions
Flow and NP

1. **Bin Packing: CLR 37-1, p. 983**

   Solution

   a) We do the reduction in two parts. First we show that the subset-sum problem is NP-hard even when restricted to the case where $k = \sum_{s \in S} w(s)/2 = W/2$. Then we show that bin-packing can solve these instances easily. Given an instance of subset-sum: (case 1) if $k = W/2$ we require no modification. (case 2) If $k < W/2$ then add an element, x, of weight $W - 2k$ to the set. If there is a subset, S, that sums to k in the old set, then there is a set $S \cup x$ that sums to $k + W - 2k = W - k$ in the new set. Since the new set has total weight $W + W - 2k = 2W - 2k$, this is a problem of subset-half-sum. (case 3) If $k > W/2$ add an element, x, of weight $2k - W$. Then, if a subset in the original set exists that sums to k, that same subset exists in the new set, and its weight satisfies $W'/2 = (W + 2k - W)/2 = 2k/2 = k$. In both cases this does not introduce a false solution, since if a set, A, exists in the new problem that does not use x, then it's complement, $S - A$, does use x and has the same sum.

   The reduction from subset-half-sum to bin-packing is now trivial.[1] Keep the same instance set and ask how many bins of size $W/2$ are required. If the number of bins required is exactly 2, then the subset-half-sum answer is yes. It is no otherwise.

   Proof: If the minimal number of bins is 2 and each bin has capacity $W/2$ then each must contain a subset of weight exactly $W/2$ and each is a solution to subset-half-sum. If the minimal number of bins is greater than 2 (it cannot be 1 since that is smaller than W), then no subset with weight $W/2$ exists. Otherwise we would use that subset in the first bin and the remaining $W/2$ weighted items in the second and have a 2-bin solution.

   b) Since the bins are of size 1, the total weight carried in x bins is at most x, so to pack S weight requires $\lceil S \rceil$ bins.

   c) If more than one bin is half full, then the first item of some less than half full bin, y, started after some other less than half full bin x, would fit into x at the time it was put into y and therefore the first-fit heuristic was not used.

   d) Assume more than $\lceil 2S \rceil$ bins were used. Since the first-fit heuristic fills all but the last bin with weight at least 1/2, the first $\lceil 2S \rceil$ bins have at least S weight in them and the $\lceil 2S \rceil + 1$ bin has some non-zero weight. This exceeds the total weight of the set packed.

   e) For any problem, at least $\lceil S \rceil$ bins are required and the first-fit heuristic uses at most $\lceil 2S \rceil$ bins. The approximation is therefore at most $\lceil 2S \rceil / \lceil S \rceil \leq 2$.

[1] We must first convert bin packing in 1-bins to bin-packing in k-bins simply by growing or shrinking the element weights by k

f) If we maintain a binary tree of bins keyed on how much room they have, we can look to see if the most empty bin has room for the next object. If it has enough room, put the object in it, remove it from the tree, and reinsert based on its reduced vacant volume. If not, then insert a new empty bin and retry. These operations take a constant number of $O(lgn)$ operations for each item and therefore the algorithm is $O(nlgn)$.

2. **Jobs and Resources**

   Solution

   Construct a flow network with souce s, sink t, verticies for each resource x_i's, and each project y_i's. Connect the source to each resource with an edge equal to the cost of that resource and each project to the sink with an edge of its reward. Also, for all projects, connect forward edges of infinite weight from each resource in its requirement list. We wish to prove that a minimal cut through this network represents an optimal selection of jobs to undertake, where the preformed jobs and resources used are those on the sink side of the cut.

   First note that in a minimal cut, no edge from a resource not used to a job performed will be included (since its weight is higher than a cut of just t) and so no job has unincluded resources and the minimal cut is a valid job selection. Second, any subset of jobs and resources can be represented as a cut and therefore any job/resource solution can be. The total benefit (to be maximized) of a valid selection is the reward of the taken projects minus the cost of the taken resources, $|P_{taken}| - |R_{taken}|$. For a cut, the value is infinite if it does not correspond to a valid job/resource selection. Otherwise it is the total project reward available minus those not taken plus the cost of the taken resources, $|P - P_{taken}| + |R_{taken}| = K - (|P_{taken} - |R_{taken}|)$. This is monotonically decreasing in benefit (linear and opposite), therefore finding the minimum cut maximizes the benefit.