Homework 1

Chengyu Lin*

Problem 1 It's easy to come out a polynomial-time algorithm related to the number of types and the value of each coins.

For the polynomial-time algorithm considering only the number of types, refer to *A Polynomial-time Algorithm for the Change-Making Problem* [David Pearson, 1994].

Problem 2 "$\Rightarrow$":

Assume that the greedy algorithm outputs S. And one of the maximum cost set is T.

First I'm going to show that $|S| = |T|$. Clearly that $|S| \leq |T|$, otherwise we can find some $t \in S - T$. And according to the definition of matroid, $T \cup \{t\}$ is a independent set and provide more cost than T, which contradicts the assumption that T is one of the maximum cost set. On the other hand $|T| \leq |S|$, otherwise there is some $t \in T - S$ satisfied $S \cup \{t\}$ is a independent set. And according to the step 3.1 of the algorithm t will be one of the elements in S. Therefore $|S| = |T|$.

Now let $S = \{a_{i_1}, a_{i_2}, \cdots, a_{i_m}\}$ and $T = \{a_{j_1}, a_{j_2}, \cdots, a_{j_m}\}$, $\{i_m\}$ and $\{j_m\}$ are both ascending sequences ranged from 1 to n. Assume T is the maximum cost set which maximized k where $i_p = j_p (p \leq k)$.

If $k = m$ then it's done. When $k < m$, i_{k+1} must be less than j_{k+1} otherwise the greedy algorithm will choose $a_{j_{k+1}}$ first. Now add $a_{i_{k+1}}$ to T, and kick one element from $\{a_{j_{k+1}}, \cdots, a_{j_m}\}$ to form a independent set A. Clearly that the cost of A is not small than T. But the k for A where $i_p = j_p(p \leq k)$ is larger than T, which contradicts the assumption.

Therefore the greedy algorithm always gives the maximum cost set.

"$\Leftarrow$":

I'm going to show that if $\{V, \mathbf{I}\}$ is not a matroid, then the greedy algorithm would fail sometimes.

Assume that $V = \{1, 2, 3\}$ and $\mathbf{I} = \{\emptyset, \{1\}, \{2\}, \{3\}, \{2, 3\}\}$. The cost function is $c(1) = 3$, $c(2) = c(3) = 2$. The greedy algorithm would provides $\{1\}$ but $\{2, 3\}$ is a better solution.

Problem 3 Proof by induction:

After the first enumeration, the algorithm provides $\{a_1\}$ which is the optimal solution for $\{a_1\}$.

Assume after enumerating a_{k-1} (before a_k is considered), the algorithm provides the optimal soluton S for $\{a_1, a_2, \cdots, a_{k-1}\}$ which is the same as offline greedy algorithm provides.

If $S \cup \{a_k\} \in \mathbb{I}$, clearly that $S \cup \{a_k\}$ is the optimal solution for $\{a_1, \cdots, a_k\}$.

WLOG, the first $k - 1$ elements are sorted by their cost. When we inserted a_k, assume that for all $i \leq j$, $a_k <= a_i$ and for all $i > j$, $a_k > a_i$. Now let's perform offline algorithm on that. For the first j elements, both online and offline algorithm will give the same result. After that if a_k contradicts the previous choice then it's done. Else we will put a_k into the result. Then for the rest elements we will choose the elements by greedy, and it will be almost the same as the optimal solution for first $k - 1$ elements except that there maybe one element can not be chosen since we have add a_k before. And clearly that this process will give the same result as the online algorithm does (step 3).

Therefore this online algorithm will provides the same result for the first k elements as offline algorithm does.

So by induction it will gives the optimal solution.

*F1003028-5100309007