Algorithm Design Assignment 9

Jianan Lin, 662024667, linj21@rpi.edu

Problem 8.19. There are n cannisters, m trucks and each truck has k containers. For the following questions, give a polynomial algorithm or prove it is NPC.

(a). Suppose each cannister can be delivered by some trucks. Is there an algorithm to judge whether we can deliver all the cannisters (i.e., no truck is overloaded) and (if yes) give the specific plan?

(b). Suppose each cannister can be delivered by any truck but some cannisters cannot be delivered by the same truck. Is there an algorithm to judge and give the specific plan?

Solution.

(a). We can use the network flow to give a polynomial algorithm. There are three parts.

Part 1. We construct a graph. We have a node u_i for each cannister and node v_j for each truck. If cannister i can be delivered by truck j, then we add a directed edge (u_i, v_j) with capacity 1. We add a source node s and for each u_i, we add a directed edge (s, u_i) with capacity 1. Last we add a sink node t and for each v_j, we add a directed edge (v_j, t) with capacity k. The time complexity of this part is $O(mn)$ because there are at most $mn + m + n$ edges.

We say we have a solution for the original question, if and only if the max $s - t$ flow of this graph is n and prove it. First notice that the max flow cannot be larger than n because the sum of capacity of edges connected to s is n. Then we prove the sufficiency and necessity.

Sufficiency: If the max flow is n, then for each u_i, there is a flow path from u_i passing v_j to t and no edge is beyond its capacity. So if we let each cannister go to its truck in the flow, then we have a feasible solution.

Necessity: If we have a feasible solution, then for each cannister i going to truck j, we draw a path/flow $s \to u_i \to v_j \to t$. So we can have a $s-t$ flow with value n and no edge is beyond its capacity because a cannister can only go to one truck, and because a truck can have at most k cannisters.

Part 2. Solve the max $s - t$ flow with Ford-Fulkerson algorithm. The time complextiy of this part is $O(Ef) = O(mn^2)$ because $E \le O(mn)$, $f \le n$.

Part 3. If the max flow is n, then return the plan, which tells us which cannister should go to which truck, according to the max flow. If not, then return no plan. The time complexity of this part is $O(n)$. Therefore the total time complexity is $O(mn + mn^2 + n) = O(mn^2)$.

(b). This is NPC and we use 3−coloring problem to prove it. Obviously this problem is in NP because we can examine it in polynomial time (in fact $O(mn^2)$ with brute-force method).

Given a 3−coloring problem with graph G and $|V| = n$, we construct this problem. For each $v_i \in V$, we have a cannister i and if there is an edge (v_i, v_j) then cannisters i, j cannot go to the same truck. We have 3 trucks with n containers each. We say this problem has a feasible solution if and only if G can be painted in 3 colors. The time complexity of constructing the problem is polynomial $O(n^2)$.

Sufficiency: If G can be painted in 3 colors, then we let cannisters with same color go to the same truck. We know two cannisters cannot go to the same truck if and only if there is an edge between two nodes, so we can have a feasible solution with this plan. No truck is overloaded with n containers.

Necessity: If we have feasible solution for this problem, then we paint G with color corresponding to each truck. For example, if one cannister goes to truck i then the node will be painted by color i. So there will never exist two neighbor nodes with the same color since the two cannisters go to different trucks. Therefore this problem is at least as hard as the 3−coloring problem, thus NPC.

Problem 8.26. Prove the Number Partitioning Problem is NPC: given n numbers $x_1, ...x_n$, is there a partition of two sets with sum of them are equal?

Solution.

We use subset sum problem to prove it. This problem is obviously in NP because we can examine it in $O(n)$ time. Notice that, here we **allow** that there are equal values in the set.

Given a subset sum problem with set $S = \{s_1, ..., s_n\}$ and a target t, we construct the following number partitioning problem: Let $X = S \cup \{s - 2t\}$, where $s = \sum_i s_i$. The time complexity of constructing this is $O(n)$.

We claim that, there is a subset $T \subseteq S$ so that the sum of T is t, if and only if, X can be divided into two parts with equal sum. Notice that sum of X is $s + s - 2t = 2s - 2t$, so the sum of each part is $s - t$.

Necessity: If there exists $T \subseteq S$ so that $\text{sum}(T) = t$, then we divide X into two parts: $T' = T \cup \{s - 2t\}$ and $R = S - T$. Let r be the sum of R so we have $r = s - t$, which is just half of the sum of X. Also we know the sum of T' is $t' = t + s - 2t = s - t$. Therefore this is feasible partition.

Sufficiency: If there is a partition with set A, B, then we know the sum of A, B are both $s - t$. According to symmetry, we assume $s - 2t \in A$, so let $A' = A - \{s - 2t\}$, then we know $\text{sum}(A') = \text{sum}(A) - (s - 2t) = s - t - s + 2t = t$. Therefore A' is the T we need in the subset sum problem. Since $A \subseteq X = S \cup \{s - 2t\}$ and $A' = A - \{s - 2t\}$, we know $A' \subseteq S$.

Therefore this problem is at least as hard as the subset sum problem, thus NPC.