Alexandra Yavhilovitch
Abby Rechkin

Homework 10

a) We are given the problem of creating an optimal course schedule with n available courses where each has a corresponding number of credits c_n. We want to satisfy the requirements of T total credits and minimize the number of courses taken.

We are trying to find $I \subseteq [n]$ such that $\sum_{i \in I} c_i = T$ and $\forall J \subseteq [n]$ such that $\sum_{j \in J} c_j = T$, and we have $|I| \leq |J|$.

We are provided with a feature in myUW which accepts a list of courses, an integer T which is the ~~total~~ number of credits, and an integer m which is a maximum numbers that will be taken. It outputs yes based on whether it is possible to take $\leq m$ courses to satisfy the credit requirement T.

GOAL: Perform a reduction from our given course scheduling problem to the provided myUW feature.

scheduling problem $\leq_T^P$ myUW feature

ALGORITHM: In order to figure out the minimum number of courses required to fulfill T credits, we want to call our oracle myUW with inputs of the n courses, T credits, and m. m initially is equal to n and is decremented until the oracle outputs "no." Meaning, the last m value for which the oracle outputs "yes" is the minimum number of courses.

Now, we want to find the m courses specifically to include in our schedule. We will do this by iterating through the list of courses, determining whether that course is essential to our schedule, by calling the oracle for the list of courses without it, credits T, and the minimum number of courses we can take m.

PROOF OF CORRECTNESS

We claim there is some schedule with a minimal number of courses q which add up to T total credits if and only if myUW outputs true, meaning that m courses sum up to t total credits.

⇒ If we have a schedule of T credits with a minimum number of courses q, then we observe a direct mapping between t and T (t=T) and m and q (m= q), meaning that the schedule is possible to generate.

⇐ If myUW outputs decisions determining whether or not m courses sum up to t credits in our list of courses, then we can determine a schedule with a minimal number of courses that sum up to T by using our algorithm in polynomial time.

RUN TIME

We are calling our oracle, at first, at most n times (O(n)) to determine the minimal number of courses. Then, we call our oracle again n times to determine the actual scheduling (O(n)).

$O(n) + O(n) = O(n)$.

b) In part a, we have detailed the specifications of the myUW feature. Now, we introduce the subset sum problem which takes in a list of integers $a_1, a_2, \ldots a_n$, an integer t and determines if there is a subset $I \subseteq [n]$ with $\sum_{i \in I} a_i = T$.

We want to perform a reduction from myUW to subset sum.

$$\text{myUW} \leq_T^P \text{subset sum}$$

ALGORITHM: our myUW has inputs of an array of n integers, m courses, and total credits t. We will manipulate our inputs for myUW into inputs to subset sum. In order to prevent accidentally outputting an incorrect decision, we will add a very large number to the credit count for each course in the array. Then, t will become $R = t + mQ$ because our input credits have changed and we only want to have a subset of m courses.

Large number $\geq$ sum of credits of all courses.

Example: courses by credits = {1, 2, 4, 7, 10, 83}

$t = 11 \quad m = 2$

Step 1: Add large number 10,000 for this instance to each element in the array

{10,001, 10,002, 10,004, 10,007, 10,010, 10,083}

Step 2: $R = t + mQ$. Get new total credit value R.

$R = 11 + 2 \cdot 10{,}000 = 20{,}011$

Step 3: Call oracle subset sum w/ new inputs and receive decision.

Step 4: If returns no, we return to step 1 and decrement m by 1. Otherwise, the subset sum decision is our myUW decision.

PROOF OF CORRECTNESS

We claim that myUW outputs a correct decision if and only if subsetSum outputs a correct decision.

⇒ Given the inputs to the myUW problem, we manipulate our input values to fit the specifications of subset sum. We do this as per our algorithm described above. Since we know the output of myUW is correct, the output of subset sum should give a correct output w/ the updated values in polynomial time.

⇐ Given that subset sum makes the correct decisions with any inputs t and n, an array, we conclude that myUW works by reversing the above arithmetic operations.

$T = R - mQ$ and we can achieve our original input array n by subtracting the large value added to each element.

We already showed that the output of subset sum can be used to determine an output for myUW our algorithm in polynomial time.

RUN-TIME

We make constant arithmetic operations to modify our input to subset sum. We make at most m calls which is $O(m)$.