Due: 5th of November 2017 at 11:59pm

COMP 2007 – Assignment 5

All submitted work must be done individually without consulting someone elses solutions in accordance with the Universitys Academic Dishonesty and Plagiarism policies.

Answers to questions 1a(i–iii), 1b(i–iii), 1c(i–iii) and 2 should be submitted via Blackboard as pdf (no handwriting!). Answers to questions 1a(iv) and 1b(iv) should be submitted via Ed.

Questions

This assignment is all about polynomial time reductions and NP-completeness. To prepare for it I suggest you read Chapter 8.1-4 in detail, and Chapters 7.5-12 might also be useful to see how correctness of a reduction is proven.

1. [**85 points**] Consider the following four problems:

   - **The Knapsack problem:** You are given a set $U = \{u_1, \ldots, u_{n_1}\}$ of n_1 items, and each item u_i has a weight $w_i > 0$, and a value $v_i > 0$. You are also given a weight capacity value $W > 0$, and a target value $V > 0$. You would like to decide if there exists a subset of the items, with weight less than or equal to W and total value greater than or equal to V.
   - **The c-Fair Sharing problem:** You are given a set $T = \{t_1, \ldots, t_{n_2}\}$ of n_2 tasks, where each task t_i has a pay off p_i, and a constant c, with $0 < c \leq 1$. You want to distribute the tasks roughly evenly among two students, that is, you would like to decide if one can divide the tasks in T into two disjoint sets T_1 and T_2, with $T_1 \cup T_2 = T$, such that:
   $$c \cdot \frac{1}{2} \sum_{t_i \in T} p_i \leq \sum_{t_i \in T_1} p_i \leq \frac{1}{2} \sum_{t_i \in T} p_i.$$
   - **The Fair Pairwise Sharing problem:** You are given a set $F = \{(a_1, b_1), \ldots, (a_{n_3}, b_{n_3})\}$ of n_3 pairs of tasks, with $a_i \geq b_i$ for all i. For each item (a_i, b_i) you can either give a_i to the first student and b_i to the second student or vice versa. You want to distribute the tasks exactly evenly among two students, that is, you would like to decide if one can divide the the tasks in F into two disjoint sets F_1 and F_2, with $F = F_1 \cup F_2$, such that:
   $$\sum_{(a_i,b_i) \in F_1} a_i + \sum_{(a_i,b_i) \in F_2} b_i = \sum_{(a_i,b_i) \in F_2} a_i + \sum_{(a_i,b_i) \in F_1} b_i.$$
   Note that this expressions says that the first student (left hand-side) gets the a_i's in F_1 and the b_i's in F_2, while the second student (the right hand-side) gets the b_i's in F_1 and the a_i's in F_2.
   - **The c-Fixed Distance problem:** Consider a robot moving along an infinite horizontal line. You are given a set $X = \{x_1, \ldots, x_{n_4}\}$ of n_4 possible movement commands that can be executed by the robot, and a value c with $1/2 \leq c < 1$. Each x_i is a 2-tuple (d_i, ℓ_i), where $d_i \in \{\text{left,right}\}$ is the direction of the robot's movement and $\ell_i \geq 0$ is

the distance the robot must move. You would like to decide if there is a subset X' of the commands that will move the robot exactly a distance

$$c \cdot \sum_{x_i \in X} l_i$$

to the right.

All problems are closely related. Assume you are given an algorithm for the Knapsack *decision* problem mentioned above. Assume that this is given to you as a black box (**you may not modify it!**). But you do know that its running time is $f_1(n_1, W, V)$, where n_1, W and V are three parameters of the knapsack instance.

(a) Design an algorithm that solves the decision version of the c-Fair Sharing problem, using the algorithm for the Knapsack decision problem as a sub-procedure. Lower complexity generally gives higher marks. (Note that the question specifically asks you to use Knapsack as a sub-procedure. Using a different sub-procedure for your reduction will give 0 points.)

i. Describe your reduction.
ii. What is the running time of your algorithm?
iii. Prove the correctness of your algorithm.
iv. Implement your algorithm. Instructions for the implementation part is available on Ed.

(b) Design an algorithm that solves the decision version of the Fair Pairwise Sharing problem, using the algorithm for the c-Fair Sharing decision problem as a sub-procedure. Lower complexity generally gives higher marks. (Note that the question specifically asks you to use the algorithm for the c-Fair Sharing decision problem as a sub-procedure. Using a different sub-procedure for your reduction will give 0 points.)

i. Describe your reduction.
ii. What is the running time of your algorithm?
iii. Prove the correctness of your algorithm.
iv. Implement your algorithm, using your implementation for the c-Fair Share algorithm as a sub-procedure. Instructions for the implementation part is available on Ed.

(c) Design an algorithm that solves the decision version of the c-Fixed Distance problem, using the algorithm for the Fair Pairwise Scheduling decision problem as a sub-procedure. Lower complexity generally gives higher marks. (Note that the question specifically asks you to use the algorithm for the Fair Pairwise Scheduling decision problem as a sub-procedure. Using a different sub-procedure for your reduction will give 0 points.)

i. Describe your reduction.
ii. What is the running time of your algorithm?
iii. Prove the correctness of your algorithm.
iv. Implement your algorithm, using your implementation for the Fair Pairwise Sharing algorithm as a sub-procedure. Instructions for the implementation part is available on Ed.

2. **[15 points]** Assume that you have polynomial time reductions for Questions 1(a-c) and assume you know that the c-Fixed Distance decision problem is NP-complete.

(a) Prove that the c-Fair Sharing decision problem is NP-complete.

(b) Prove that the Fair Pairwise Scheduling decision problem is NP-complete.

(c) Prove that the Knapsack decision problem is NP-complete.