CS31005 Algorithms–II, Autumn 2022–2023

Class Test 2

03–November–2022 | 07:00pm–08:00pm | Maximum marks: 20

Roll no: _____ **Name:** _____

[*Write your answers in the question paper itself. Be brief and precise. Answer all questions.*
If you use any algorithm/result/formula covered in the class, just mention it, do not elaborate.]

1. Let $C = (a_1, a_2, a_3, \ldots, a_n)$ be a collection of positive integers with the sum $A = a_1 + a_2 + a_3 + \cdots + a_n$. Also let $T = \left\lfloor \frac{A}{2n} \right\rfloor$. Justify which of the problems in the following three parts is/are NP-complete. No credit without proper justifications. Assume that P $\neq$ NP. (**Hint:** Use the partition problem whenever needed.)

(a) Determine whether there exists a subcollection of C with sum $\leqslant T$. **(4)**
(b) Determine whether there exists a subcollection of C with sum $\geqslant T$. **(4)**
(c) Determine whether there exists a subcollection of C with sum $= T$. **(4)**

Solution **(a)** Not NP-complete. The answer is *yes* if and only if C contains an element $\leqslant T$. This condition can be checked in linear time, so the problem is in P.

(b) Not NP-complete. The problem is trivial, because C itself gives a sum $A \geqslant T$.

(c) NP-complete. First note that the problem is in NP (why?). In order to prove its NP-hardness, we use a reduction from PARTITION. Let $(b_1, b_2, b_3, \ldots, b_n)$ be an instance of PARTITION with $B = b_1 + b_2 + b_3 + \cdots + b_n$ even. Create the instance $C = (b_1, b_2, b_3, \ldots, b_n, Bn)$ for the given problem. For the converted instance, we have $A = (n+1)B$ and $T = \frac{B}{2}$. Since the new element $Bn \geqslant B > \frac{B}{2}$, a sum of $B/2$ can be achieved in C if and only if the sum $B/2$ is achieved from $(b_1, b_2, b_3, \ldots, b_n)$.

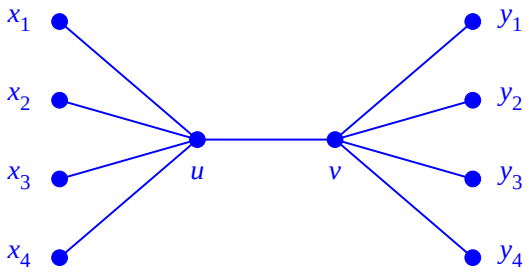
2. Define the approximation ratio of an approximation algorithm. What is the difference between a polynomial-time approximation scheme (PTAS) and a fully polynomial-time approximation scheme (FPTAS)? **(2 + 2)**

3. Consider the matching-based 2-approximation algorithm covered in the class, for computing minimum vertex covers in undirected graphs. We have seen that for the complete bipartite graph $K_{n,n}$, the algorithm outputs a vertex cover with $2 \times \text{OPT}$ vertices (OPT is the minimum size of a vertex cover), no matter how the edges are chosen in the iterations of the algorithm. In general, the suboptimal solution SUBOPT output by the algorithm depends on the choices of the edges in the different iterations. In this exercise, your task is to find an example of *one* connected undirected graph with exactly ten vertices such that for that graph,

(a) some choices of the edges in the iterations give $\text{SUBOPT} = \text{OPT}$, whereas
(b) some other choices of the edges in the iterations give $\text{SUBOPT} = 2 \times \text{OPT}$.

Draw the graph, and mention the edges chosen in the iterations in each case. **(4)**

Solution Consider the following graph.



Since a single vertex cannot cover all the edges, and $\{u, v\}$ is already a vertex cover, we have $\text{OPT} = 2$.

(a) If the algorithm chooses the edge (u, v) in the first iteration, then all the edges are covered, and the minimal vertex cover $\{u, v\}$ is output, so $\text{SUBOPT} = \text{OPT}$ in this case.

(b) If the algorithm chooses an edge (x_i, u) in the first iteration, then an edge (v, y_j) needs to be chosen in the second iteration. So the algorithm produces the cover $\{x_i, u, v, y_j\}$ with $\text{SUBOPT} = 2 \times \text{OPT}$.

For rough work