

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  $C = (a_1, a_2, a_3, \dots, a_n)$  be a collection of positive integers with the sum  $A = a_1 + a_2 + a_3 + \dots + a_n$ . 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  $\leq T$ . (4)
  - (b) Determine whether there exists a subcollection of  $C$  with sum  $\geq T$ . (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  $\leq T$ . This condition can be checked in linear time, so the problem is in P.

It is not mentioned in the question that the collection has to be non-empty. Give full credit if some takes the empty collection.

No credits for just writing Yes/No.

No credit for falsely proving NP-completeness.

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

No credit for writing Yes/No only, or for proving NP-completeness.

(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  $(b_1, b_2, b_3, \dots, b_n)$  be an instance of PARTITION with  $B = b_1 + b_2 + b_3 + \dots + b_n$  even. Create the instance  $C = (b_1, b_2, b_3, \dots, b_n, Bn)$  for the given problem. For the converted instance, we have  $A = (n+1)B$  and  $T = \frac{B}{2}$ . Since the new element  $Bn \geq B > \frac{B}{2}$ , a sum of  $B/2$  can be achieved in  $C$  if and only if the sum  $B/2$  is achieved from  $(b_1, b_2, b_3, \dots, b_n)$ .

No credit for writing Yes/No only, or for disproving NP-completeness.

Students are allowed to use any problem other than PARTITION for the reduction. Give due credits based on the correctness.

1 mark for arguing correctly that the problem is in NP. Either a certificate or a non-deterministic algorithm is fine (if correct).

2 marks for proposing a reduction map that works.

1 mark for proving that the reduction works.

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)**

These are standard textbook definitions. However, some authors take the approximation ratio as  $\text{SUBOPT}/\text{OPT}$  for all types of optimization problems. Some others take  $\max(\text{SUBOPT}/\text{OPT}, \text{OPT}/\text{SUBOPT})$ , that is,  $\text{SUBOPT}/\text{OPT}$  for minimization problems and  $\text{OPT}/\text{SUBOPT}$  for maximization problems, as the approximation ratio. Give credit to both the definitions (if written correctly).

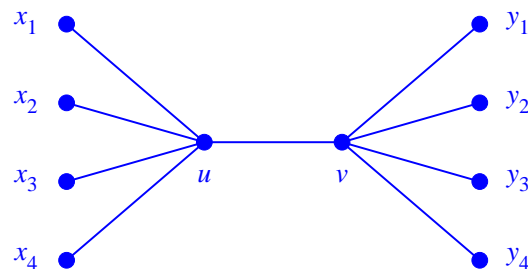
3. Consider the matching-based 2-approximation algorithm covered in the class, for computing minimum vertex covers in undirected graphs. We have seen that for the complete bipartite graph  $K_{n,n}$ , the algorithm outputs a vertex cover with  $2 \times \text{OPT}$  vertices (OPT is the minimum size 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  $\text{SUBOPT} = 2 \times \text{OPT}$ .

2 marks for a correct graph. Deduct 1 mark if either (a) or (b) cannot be proved. No credit if neither (a) nor (b) can be proved.

1 mark for a correct sequence establishing (a).

1 mark for a correct sequence establishing (b).

No credit in the following cases:

(i) Two graphs are drawn, one for (a) and the other for (b).

(ii) The graph does not contain exactly ten vertices. However, if it is a minor mistake that can be repaired easily, give partial credit.

(iii) Only a proof/argument is supplied without any specific graph as an example.

