INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

Stamp / Signature of the Invigilator

EXAMINATION (Mid Semester)										**SEMESTER (Autumn)**			
Roll Number										**Section**		**Name**	
Subject Number	C	S	6	0	0	8	6	**Subject Name**		*Selected Topics in Algorithms*			
Department / Center of the Student											**Additional sheets**		

Important Instructions and Guidelines for Students

1. You must occupy your seat as per the Examination Schedule/Sitting Plan.
2. Do not keep mobile phones or any similar electronic gadgets with you even in the switched off mode.
3. Loose papers, class notes, books or any such materials must not be in your possession, even if they are irrelevant to the subject you are taking examination.
4. Data book, codes, graph papers, relevant standard tables/charts or any other materials are allowed only when instructed by the paper-setter.
5. Use of instrument box, pencil box and non-programmable calculator is allowed during the examination. However, exchange of these items or any other papers (including question papers) is not permitted.
6. Write on both sides of the answer script and do not tear off any page. **Use last page(s) of the answer script for rough work.** Report to the invigilator if the answer script has torn or distorted page(s).
7. It is your responsibility to ensure that you have signed the Attendance Sheet. Keep your Admit Card/Identity Card on the desk for checking by the invigilator.
8. You may leave the examination hall for wash room or for drinking water for a very short period. Record your absence from the Examination Hall in the register provided. Smoking and the consumption of any kind of beverages are strictly prohibited inside the Examination Hall.
9. Do not leave the Examination Hall without submitting your answer script to the invigilator. **In any case, you are not allowed to take away the answer script with you.** After the completion of the examination, do not leave the seat until the invigilators collect all the answer scripts.
10. During the examination, either inside or outside the Examination Hall, gathering information from any kind of sources or exchanging information with others or any such attempt will be treated as **'unfair means'**. Do not adopt unfair means and do not indulge in unseemly behavior.

Violation of any of the above instructions may lead to severe punishment.

Signature of the Student

To be filled in by the examiner

Question Number	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Total**
Marks Obtained											

Marks obtained (in words)	**Signature of the Examiner**	**Signature of the Scrutineer**

CS60035/CS60086 Selected Topics in Algorithms, Autumn 2016–2017

Mid-Semester Test

20–September–2016 | CSE-107, 2:00–4:00pm | Maximum marks: 60

[*Write your answers in the question paper itself. Be brief and precise. Answer all questions.*]

1. In the makespan scheduling problem, n jobs with processing times $t_1, t_2, \ldots, t_n$ are to be assigned to m identical processors. All the machines start operating at the same time $t = 0$, and finish the assigned jobs without any idle time between consecutive jobs. The time when the machine with the highest assigned load finishes is called the makespan. The objective is to minimize the makespan. Assume that $m \leqslant n$. Let $T = \max(t_1, t_2, \ldots, t_n)$, so the input size can be taken as $n \log_2 T$. Consider the following two parameterizations of the input:

$$\begin{aligned} Par_1(t_1, t_2, \ldots, t_n; m) &= m^n, \\ Par_2(t_1, t_2, \ldots, t_n; m) &= n^m. \end{aligned}$$

Prove the following assertions.

(a) The makespan scheduling problem has a polynomial-time Par_1-parameterized algorithm. **(6)**

Solution The exhaustive-search algorithms assigns each job to each of the processors. There are m^n assignments possible. For each assignment of all the n jobs, the makespan can be computed in time polynomial in the input size.

(b) The makespan scheduling problem cannot have a polynomial-time Par_2-parameterized algorithm unless P = NP. **(8)**

Solution Assume that there is a Par_2-parameterized polynomial-time algorithm A for the makespan scheduling problem. The running time of A is $O((n^m)^k(n\log T)^l)$ for some constant k,l. We use this algorithm to solve the subset-sum problem in polynomial time. Let $t_1,t_2,\ldots,t_n$ be an instance of the subset-sum problem (so that $t_1+t_2+\cdots+t_n=2S$ is even). We take $m=2$, and run A on $(t_1,t_2,\ldots,t_n;2)$, and return *true* if and only if the optimal makespan returned by A is S. The running time is $O(n^{2k+l}\log^l T)$ which is $O((n\log T)^{2k+l})$, that is, polynomial in the input size. Since the subset-sum problem is NP-complete, this algorithm for solving the subset-sum problem in polynomial time proves P = NP.

2. We want to solve the following real-valued linear-programming problem:

$$\begin{array}{ll} \text{minimize} & 3x_1 + 2x_2 + 2x_3 + 6x_4 \\ \text{subject to} & x_1 + x_2 \geqslant 5, \quad x_1 + x_3 \geqslant 4, \quad x_4 \geqslant 1, \quad x_1, x_2, x_3 \geqslant 0. \end{array}$$

(a) What is the dual of this problem? (No need to derive.) **(4)**

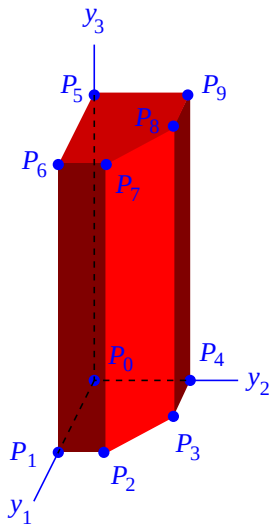
Solution There are three constraints (first, second and third) in the primal problem. Thus, the dual problem is:

$$\begin{array}{ll} \text{maximize} & 5y_1 + 4y_2 + y_3 \\ \text{subject to} & y_1 + y_2 \leqslant 3, \quad y_1 \leqslant 2, \quad y_2 \leqslant 2, \quad y_3 \leqslant 6, \quad y_1, y_2, y_3 \geqslant 0. \end{array}$$

(b) Find all the vertices of the polytope defined by the constraints of the dual problem. **(6)**

Solution The polytope corresponding to the given constraints are shown in the adjacent figure. The ten vertices on the polytope are:

$$\begin{aligned} P_0 &= (0,0,0), \\ P_1 &= (2,0,0), \\ P_2 &= (2,1,0), \\ P_3 &= (1,2,0), \\ P_4 &= (0,2,0), \\ P_5 &= (0,0,6), \\ P_6 &= (2,0,6), \\ P_7 &= (2,1,6), \\ P_8 &= (1,2,6), \\ P_9 &= (0,2,6). \end{aligned}$$



(c) Evaluate the dual objective function at the vertices of the polytope of Part (b), and solve the dual problem optimally. **(6)**

Solution The objective function—call it $f(y_1, y_2, y_3) = 5y_1 + 4y_2 + y_3$—is maximized at a vertex of the constraint polytope. So it suffices to evaluate f at P_i for $i = 0, 1, 2, \ldots, 9$. We have:

$$\begin{aligned} f(P_0) &= 0, \\ f(P_1) &= 10, \\ f(P_2) &= 14, \\ f(P_3) &= 13, \\ f(P_4) &= 8, \\ f(P_5) &= 6, \\ f(P_6) &= 16, \\ f(P_7) &= 20, \\ f(P_8) &= 19, \\ f(P_9) &= 14. \end{aligned}$$

It follows that the dual problem has the optimal (maximum) solution 20 achieved at the vertex P_7.

(d) What is the optimal solution of the original (that is, primal) problem? **(4)**

Solution By the principal of (strong) duality, the primal problem has the optimal (minimum) solution 20.

3. Let $G = (V,E)$ be an undirected graph with n vertices and m edges. Each edge $e \in E$ is associated with a positive integer cost $c(e)$. A cut (S,T) of G with $T = \overline{S} = V \setminus S$ is defined by the set of edges $C(S,T) = \{(u,v) \in E \mid u \in S,\ v \in T\}$. The cost of the cut is $cost(S,T) = \sum_{e \in C(S,T)} c(e)$. The *weighted maximum-cut problem* is to find a cut (S,T) for which $cost(S,T)$ is maximized.

(a) Assuming that the edge costs are bounded by a constant, propose a polynomial-time local-search 2-approximation algorithm for the weighted maximum-cut problem. **(6)**

Solution

```
Set S = ∅.
While (1) {
   Find a vertex u such that switching u to the other part increases cost(S, S̄).
   If no such u is found, return (S, S̄),
   otherwise switch u to the other part.
}
```

Each iteration increases the cost of the cut by at least one. If the edge costs are bounded by B (a constant), the maximum number of iterations is $B \times |E|$, that is, the algorithm runs in polynomial time.

(b) Prove that the approximation factor of your algorithm is 2. **(6)**

Solution For each $u \in V$, let $LN(u)$ denote the set of neighbors of u in the same part (S or T) as u (the local neighbors), and let $RN(u)$ denote the set of non-local (remote) neighbors of u. Also, let $Nbr(u) = LN(u) \cup RN(u)$ (the set of all neighbors of u). The algorithm terminates when for all $u \in V$, we have

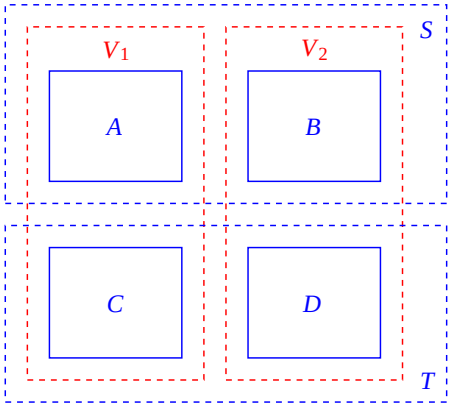
$$\sum_{v \in RN(u)} c(u,v) \geqslant \sum_{w \in LN(u)} c(u,w).$$

Now, v is a remote neighbor of u if and only if u is a remote neighbor of v. It therefore follows that

$$
\begin{aligned}
2 \times cost(S,T) &= \sum_{u \in V} \sum_{v \in RN(u)} c(u,v) \geqslant \sum_{u \in V} \frac{1}{2} \left[\sum_{v \in RN(u)} c(u,v) + \sum_{w \in LN(u)} c(u,w) \right] \\
&= \frac{1}{2} \sum_{u \in V} \sum_{v \in Nbr(u)} c(u,v) = \frac{1}{2} \times 2 \times \sum_{e \in E} c(e) = \sum_{e \in E} c(e) \geqslant OPT.
\end{aligned}
$$

(c) Prove/disprove: The approximation factor 2 of your algorithm is tight, that is, you can construct a graph for which the algorithm may stop with $cost(S,T)$ equal to exactly half of the cost of the optimal cut. **(6)**

Solution Consider the example provided in the following figure.



Here, $n = 4k$, and G is the complete bipartite graph $K_{2k,2k}$ with the bipartition V_1, V_2 having sizes $|V_1| = |V_2| = 2k$. Each edge has cost 1, so the optimal cut has cost $OPT = |E| = m = (2k) \times (2k) = 4k^2$. Suppose that the algorithm reaches an iteration where the current cut (S,T) corresponds to the four subsets $A = V_1 \cap S$, $B = V_2 \cap S$, $C = V_1 \cap T$, and $D = V_2 \cap T$ of sizes k each. With each edge cost being 1, this is an instance of the unweighted maximum-cut problem. Each $u \in V$ has exactly k neighbors in its part (S or T), and exactly k neighbors in the other part. For example, if $u \in A$, its neighbors in S are all the vertices in B, and its neighbors in T are all the vertices in D. Therefore by moving any vertex from its part to the other, we cannot improve the cost of the current cut, and this cost is the sum of the counts of A-D and B-C edges, that is, $cost(S,T) = 2k^2 = \frac{1}{2} \times OPT$.

4. Propose a 0, 1-valued linear-programming formulation of the weighted maximum-cut problem of Exercise 3. You do not have to solve the LP (or its relaxed version). Only express the given problem as a maximization problem, and specify all the required constraints. **(8)**

Solution For each $u \in V$, introduce a variable x_u with the interpretation that

$$x_u = \begin{cases} 0 & \text{if } u \in S, \\ 1 & \text{if } u \in T. \end{cases}$$

For each $e \in E$, introduce a variable y_e with the interpretation that

$$y_e = \begin{cases} 1 & \text{if } e \in C(S,T) \text{ (that is, } e \text{ is a cut edge)}, \\ 0 & \text{otherwise.} \end{cases}$$

For each $e \in V$, introduce another variable z_e with the interpretation that

$$z_e = \begin{cases} 0 & \text{if both the endpoints of } e \text{ are in } S, \\ 1 & \text{otherwise.} \end{cases}$$

Our task is to

$$\text{maximize} \quad \sum_{e \in E} c(e) y_e.$$

For setting up the desired constraints, take any edge $e = (u,v) \in E$. If both $u, v \in S$, we have $x_u = x_v = y_e = 0$. If one of u, v belongs to S and the other to T, we have $x_u + x_v + y_e = 2$. Finally, if both u, v belong to T, we have $x_u = x_v = 1$ and $y_e = 0$. So the constraints are

$$x_u + x_v + y_e = 2z_e \quad \text{for all } e = (u,v) \in E.$$