CS60007 Algorithm Design and Analysis 2018
Assignment 3

Palash Dey and Swagato Sanyal
Indian Institute of Technology, Kharagpur

Please submit the solutions of problem 1, 2, 9, 21 and 20. The deadline is October 25, 2018 in the class.

General Instructions

- ▷ Please prove correctness of every algorithm you design.
- ▷ Please compute running time of every algorithm you design.
- ▷ If not explicitly specified otherwise, please assume that the graphs under consideration are finite, weighted, and directed graphs which does not contain any self loop.
- ▷ We have not defined some standard problems. You are expected to find the problem statements (from Google say) by yourself.

1. Recall the linear programming formulation of the max-flow problem that we used in the lecture to prove the max-flow min-cut theorem. We turned the max-flow problem into a circulation problem by adding an edge from t to s of infinite capacity. In this exercise you will redo the proof of the max-flow min-cut theorem using the following more natural linear programming formulation of max-flow. Assume that s does not have any incoming edge and t does not have any outgoing edge.

   maximize $\sum_{v:(s,v)\in E} f_{s,v}$

   subject to:

   for each edge e, $f_e \leqslant c_e$, (capacity constraints)

   for each vertex $v \notin \{s,t\}$, $\sum_{(u,v)\in E} f_{u,v} - \sum_{(v,w)\in E} f_{v,w} = 0$, (flow-conservation constraints)

   for each edge e, $f_e \geqslant 0$. (non-negativity constraints)

   Call this LP L.

(a) [1 marks] Turn the equality constraints to 'less than or equal to' constraints, by observing that the constraint $X = 0$ is equivalent to the pair of constraints $X \leqslant 0$ and $-X \leqslant 0$.

(b) [4 marks] Compute the dual of L. Call it L'. Make sure you handle various types of edges (depending on whether or not one or more of the endpoints are in $\{s, t\}$) appropriately.

(c) [2 marks] Each equality constraint in the primal L was split into two inequality constraints. Note that in the dual L', the dual variables corresponding to these two constraints always appear together. More specifically, the difference of these dual variables appears in L'. Argue that if each such difference is replaced by a single variable without non-negativity constraint, the value of L' does not change.

(d) [3 marks] Prove that the coefficient matrix of L (and hence of L') is totally unimodular.

Hint: Recall that in class we showed that the coefficient matrix of the bipartite matching LP is totally unimodular. Can you find a proof along the same lines?

(e) [5 marks] Prove the max-flow min-cut theorem. Try to find a proof along the lines of the proof done in the class.

2. [15 marks] Let $G = (V, E)$ be an undirected graph. A max-cut of G is defined to be a partition $(A, V \setminus A)$ of vertices such that $A \neq V, \emptyset$, that has maximum number of cut edges $|\{(u, v) \in E \mid u \in A, v \in B\}|$[1]. Prove that the value of the following 0-1 integer program is equal to the size of a max-cut of G.

$$\begin{aligned}
&\text{maximize } \textstyle\sum_{e \in E} \gamma_e \\
&\text{subject to:} \\
&\text{for each edge } e = (u, v), \gamma_e \leqslant \begin{cases} \mu_u + \mu_v, \\ 2 - (\mu_u + \mu_v). \end{cases} \\
&\gamma_e, \mu_v \in \{0, 1\}.
\end{aligned}$$

Hint: Split the proof into two parts:

- ▷ Prove that the value of the program $\leqslant$ the size of a max-cut as follows: for any cut of the graph, show that there exists a feasible point of the above program for which the value of the objective function is at most the size of the cut.
- ▷ Prove that the size of a max-cut $\leqslant$ the value of the program as follows: for any feasible point of the above program, show that there exists a cut whose size is at most the value of the objective function on this point.

[1] Do not confuse it with an s-t cut. G is undirected, uncapacitated, and do not have special vertices s, t.

3. Prove that if there exists a polynomial time algorithm for any NP-complete problem, then P=NP.

4. Recall the subset-sum problem: Given a multiset of integers $w_1, \ldots, w_n$ and a target integer W, does there exist a subset S of $\{1, \ldots, n\}$ such that $\sum_{i \in S} w_i = W$? Design a dynamic programming algorithm for the subset-sum problem that runs in time $O(nW)$.

5. Comment on the following 'reasoning':

   Subset-sum is known to be an NP-complete problem. By the problem 4, subset-sum has a polynomial time algorithm. Thus, it is proved that P=NP.

6. The *Component Grouping* problem is defined as follows: Given a graph G that is not connected, and an integer k, does there exist a set of connected components of G whose union has exactly k vertices? Now consider the following claim.

   Component Grouping is NP-complete.
   We present to you the following as a proof of this claim.

   Alleged proof: Component grouping is easily seen to be in NP. Now, the subset-sum problem reduces to Component grouping as follows: for each integer w_i, introduce a distinct path in the graph with w_i vertices. The final graph is the collection of these paths. These paths are the various connected components of this graph. Set the parameter k to the target sum W in the subset-sum instance.

   Is the above proof correct? Now, following is a polynomial time algorithm for solving Component Grouping. By DFS, find the sizes of the various connected components of the graph. Let there be ℓ connected components whose sizes are $n_1, \ldots, n_\ell$. Note that k can be assumed to be at most n, as the size of the union of some connected components can be at most n. Thus we obtain a subset-sum instance where the integers are $n_1, \ldots, n_\ell$ and the target sum is k. Finally, use the dynamic programming algorithm from the previous problem to solve this subset-sum instance in $O(\ell \cdot k) = O(n^2)$ time.

   Have we proved that P=NP?

7. Let us define a complexity class PC as follows: A decision problem $\mathcal{A} \in PC$ if and only if the following two conditions are satisfied:

   - $\mathcal{A} \in P$,
   - For every language $\mathcal{B} \in P$, $\mathcal{B} \leqslant_p \mathcal{A}$.

   Prove that PC=P.

8. Given a graph $G = (V, E)$ and independence set A is a set of vertices no two of which are connected by an edge, i.e., for each $u, v \in A$, $(u, v) \notin E$. The Independence set decision problem IS is defined as follows: For a graph G and an integer k, IS(G, k)= 1 if G has an independent set of size at least k, and 0 otherwise. Prove that IS is NP-complete.

   Hint: Reduce from vertex cover.

9. [15 marks] The Circuit-SAT problem is defined as follows. The input is a Boolean circuit consisting of binary[2] AND, OR and NOT gates. The output is 1 if there exists a truth assignment to the input variables of the circuit such that the circuit evaluates to TRUE, and 0 otherwise. Prove that Circuit-SAT is NP-Complete, by reducing it from some NP-complete problem done in the class (i.e., SAT, 3SAT, Vertex Cover, Subset Sum).

10. In this problem you will reduce Circuit-SAT to SAT as follows. The input to the reduction is a Boolean circuit consisting of binary AND, OR and NOT gates.

    - ▷ For every wire in the circuit, introduce a Boolean variable.
    - ▷ For each AND gate do the following. Let the variables corresponding to the input wires of the gate be z_1, z_2 and the variable corresponding to the output wire be z. Introduce the following clauses: $(z \vee \overline{z_1} \vee \overline{z_2})$, $(\overline{z} \vee z_1 \vee z_2)$, $(\overline{z} \vee z_1 \vee \overline{z_2})$, $(\overline{z} \vee \overline{z_1} \vee \overline{z_2})$. Verify for yourself that AND of these 4 clauses is equivalent to the condition $z = (z_1 \wedge z_2)$.
    - ▷ For each OR gate do the following. Let the variables corresponding to the input wires be w_1, w_2 and the variable corresponding to the output wire be w. Introduce a set of clauses such that their AND is equivalent to the condition $w = (w_1 \vee w_2)$.
    - ▷ For each NOT gate do the following: Let the variables corresponding to the input and output wires be x_1 and x_2 respectively. Introduce a set of clauses whose AND is equivalent to the condition $x_2 = \overline{x_1}$.

    The SAT instance that you output is the collection of all the clauses constructed in the above steps. Prove that it is a valid reduction, i.e., the resultant SAT formula is satisfiable if and only if the original Boolean circuit has a satisfying assignment. Do you see that the reduction runs in polynomial time?

11. We saw that the activity selection problem is solvable by a simple greedy algorithm. In this problem you will show that a similar but more general scheduling problem is NP-complete. However, unlike the activity selection problem where we had to output a schedule, this is a decision problem. There are n jobs. For the i-th job, $i = 1, \ldots, n$,

[2] that is with two inputs.

we are given a release time r_i where it is first available for processing, a deadline d_i by which it must be completed, and a processing duration t_i. Assume that all the parameters are non-negative integers. In order to complete job i, it must be started at or after time r_i, it must be given a contiguous time slot of t_i units, and it must finish by time d_i. The machine can run only one job at a time. The problem is to find out if the jobs can be scheduled such that all of them are completed before their respective deadlines. Prove that this problem is NP-complete.

In the following questions, you are asked to reduce various problems to other problems. The problems can all be easily verified to be in NP. The problems from which you are asked to reduce are all known to be NP-complete.

12. Reduce 3-Dimensional Matching to Exact Set Cover by 3 Sets.
13. Reduce 3-Dimensional Matching to 4-Dimensional Matching.
14. Reduce Subset Sum to Partition.
15. Reduce Partition to Knapsack.
16. Reduce Partition to Bin Packing.
17. Reduce Vertex Cover to Set Cover.
18. Reduce Set Cover to Hitting Set.
19. Reduce Vertex Cover to Integer Programming.
20. Reduce Vertex Cover to Dominating Set.
21. [15 marks] Reduce from 3SAT to Max-cut.
22. [15 marks] Reduce from 3SAT to Not All Equal 3SAT.
23. Reduce from Hamiltonian Cycle to Hamiltonian Path.
24. Reduce from Hamiltonian Path to Hamiltonian Cycle.
25. Reduce from Hamiltonian Cycle to Travelling Salesman.
26. Reduce from Hamiltonian Cycle to Longest Path.
27. Reduce from 3SAT to Graph 3-Colorability.
28. Reduce Clique to Subgraph Isomorphism.