COL 351 : Analysis & Design of Algorithms

Major Exam

Nov 22, 2024 | Rohit Vaish

Problem 1

Problem 1 [15 points] This is a three-part problem about reductions. In each part, use 2-3 sentences to explain the construction of the reduced instance and justify its correctness.

(a) **[5 points]** Recall that, given an undirected graph and an integer k, the INDEPENDENT SET problem asks to compute a set S of *mutually non-adjacent* vertices such that $|S| \geqslant k$. The CLIQUE problem, on the other hand, asks for a set S of *mutually adjacent* vertices such that $|S| \geqslant k$. Show a reduction from INDEPENDENT SET to CLIQUE.

Given an instance $\langle G=(V,E), k \rangle$ of IND. SET, construct an instance of CLIQUE $\langle G'=(V,E'), k \rangle$ where:

$$\forall\ u,v \in V : \quad (u,v) \in E \iff (u,v) \notin E'.$$

Key observation:

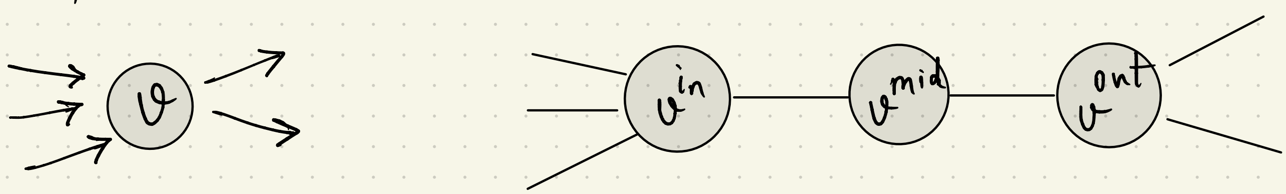
Any indep. set of G is a clique of G' and vice versa.

(b) [**5 points**] Recall that given a directed graph $G = (V, E)$, a starting vertex $s \in V$, and an ending vertex $t \in V$, the DIRECTED HAMILTONIAN PATH problem asks to compute an $s - t$ path in G that visits every vertex in G exactly once or correctly declare that no such path exists. The UNDIRECTED HAMILTONIAN PATH problem is defined similarly for undirected graphs. Show a reduction from DIRECTED HAMILTONIAN PATH to UNDIRECTED HAMILTONIAN PATH.

Hint: Think about simulating "directedness" by splitting a vertex into multiple copies.

Given an instance $\langle G=(V,E), s, t\rangle$ of DIR. HAM. PATH, construct an instance of UNDIR. HAM. PATH $\langle G'=(V',E'), s', t'\rangle$ as follows:

* For every $v \in V$, create v^{in}, v^{mid}, v^{out} in V'.
* For every $(u,v) \in E$, create edges (u^{out}, v^{in}) in E'.
* For every $v \in V$, create edges (v^{in}, v^{mid}) and (v^{mid}, v^{out}) in E'.
* $s' = s^{in}$, $t' = t^{out}$



Key observation:

Any undirected Ham path in G' must be of the form:

$s^{in}, s^{mid}, s^{out}, \ldots, v^{in}, v^{mid}, v^{out}, \ldots t^{in}, t^{mid}, t^{out}.$

Thus, a directed Ham cycle in G can be naturally inferred from an undirected Ham cycle in G' and vice versa.

(c) [**5 points**] Show a reduction from UNDIRECTED HAMILTONIAN PATH problem to the search version of TRAVELING SALESMAN PROBLEM.

Hint: Think about adding a dummy vertex and using only *binary* costs, i.e., all costs in $\{0, 1\}$.

Given an instance $\langle G = (V, E), s, t \rangle$ of UNDIR. HAM. PATH, construct an instance $\langle G' = (V', E'), \{c_e\}_{e \in E'} \rangle$ of TSP as follows:

* $V' = V \cup \{v_0\}$ $\quad v_0$ = dummy vertex
* $E' = E \cup \{(v_0, v)\}_{v \in V}$
* $c_{v_0, s} = c_{v_0, t} = 0$

  $c_{v_0, v} = 1 \quad \forall\, v \in V \setminus \{s, t\}$

  $c_{u, v} = 0 \quad \forall\, u, v \in V$

Key observation:

A TSP tour of cost 0 in G' naturally induces an undirected Ham Path in G and vice versa.

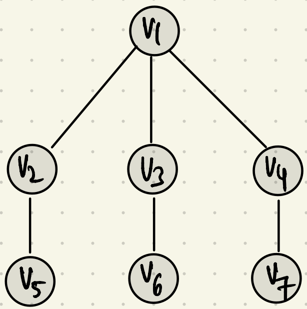
Problem 2

Problem 2 [15 points] In each part of this problem, you will be presented with a problem statement and a greedy approach for that problem. In each case, either briefly justify the correctness of the approach (2-3 sentences) or provide a counterexample.

(a) **[5 points]** *Problem*: Given an undirected graph $G = (V, E)$ and an integer k, compute a *vertex cover* of G of size at most k, if one exists. That is, find a subset $S \subseteq V$ of at most k vertices such that for every edge $e \in E$, there exists a vertex $v \in S$ such that e is incident to v.

Greedy approach: Identify the maximum degree vertex and include it in the solution. Now remove all edges incident to the chosen vertex and repeat the process on the remaining graph. Continue until k vertices are selected in this manner. If needed, break ties as you wish.

The following counterexample shows that the greedy approach is incorrect.



Greedy picks $\{V_1, V_2, V_3\}$ and returns "No".

However, $\{V_2, V_3, V_4\}$ is a valid vertex cover.

$k = 3$

(b) [**5 points**] *Problem*: 3-SAT.

Greedy approach: Find a literal that makes the largest number of clauses TRUE. Set the corresponding variable TRUE/FALSE accordingly. Remove all clauses satisfied by this truth assignment from the 3-SAT instance and repeat the process on the remaining instance.

The following counterexample shows that the greedy approach is incorrect.

$$(x_1 \vee *) \wedge (x_1 \vee *) \wedge (\neg x_1 \vee \neg x_2)$$
$$(x_2 \vee *) \wedge (x_2 \vee *) \wedge (* \vee *)$$

where each $*$ denotes a distinct literal $x_3, \dots, x_8$.

Greedy sets $x_1 = \text{TRUE}$ and $x_2 = \text{TRUE}$.

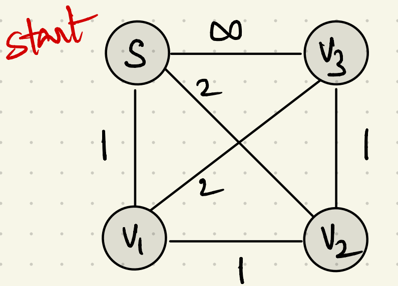
This falsifies the third constraint, so Greedy returns "NO".

However, there exists a satisfying assignment: $x_1, x_2 = \text{FALSE}$, $* = \text{TRUE}$

(c) [**5 points**] *Problem*: TRAVELING SALESMAN PROBLEM, where the input is a complete graph.

Greedy approach: Starting from a vertex of your choice, repeat the following process until all vertices have been visited: If the current vertex is v, visit the "closest unvisited neighbor" of v (i.e., a vertex w that minimizes the cost $c_{v,w}$).

The following counterexample shows that the greedy approach is incorrect.



Greedy visits the vertices in the order S, V_1, V_2, V_3

resulting in a cost of ∞.

Optimal tour is (S, V_1, V_3, V_2) with cost $= 6$.

Problem 3

Problem 3 [15 points]

Suppose there are n agents $a_1, a_2, \ldots, a_n$ and m items $g_1, g_2, \ldots, g_m$. The agents have *binary* values for the items, that is, the value of agent a_i for item g_j is given by $v_{i,j} \in \{0, 1\}$. An agent's value for a set of items is given by the sum of its values for individual items in that set. The goal is to partition the m items among the n agents in a *fair* manner.

Denote an allocation by $A := (A_1, A_2, \ldots, A_n)$, where A_i is the subset of items assigned to agent i. We require that for any $i \neq k$, $A_i \cap A_k = \emptyset$ (i.e., items are not shared between bundles) and $\cup_i A_i$ is the entire set of items (i.e., no item is left unallocated).

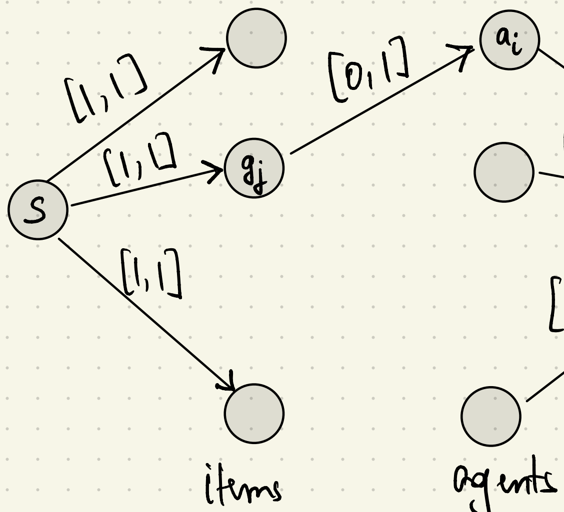
Design a polynomial-time algorithm for computing an allocation that makes the least-happy agent as happy as possible. That is, compute an allocation A that maximizes $\min_i v_i(A_i)$. Justify the correctness and running time of your algorithm. If necessary, you may assume that for every item, there is at least one agent who values it at 1.

We will use max flow with lower and upper bounds on capacities.
(Problem 7, Tutorial Sheet 12).

Let θ be our "guess" of the utility of the least-happy agent.

Note : θ is an integer in $[0, m]$ where m = no. of items.

For any fixed θ, construct a flow network G_θ as follows :



* Add edge $g_j \to a_i$ if $v_{i,j} = 1$.

* Capacity $[l, u]$ denotes lower bound l and upper bound u.

Algorithm

For $\theta \in \{0, 1, 2, \cdots, m\}$

* Compute an integral max flow in G_θ, if a feasible flow exists.

  // recall that lower + upper bounds problem reduces to the upper bound-only problem, which is solved by Edmonds-Karp

* If max flow $= m$, continue.

  Otherwise return $(\theta - 1)$ and the corresponding flow.

NOTE: A feasible flow certainly exists for $\theta = 0$.

* The running time is polynomial because the for-loop iterates at most m times, and, in each iteration, construction of flow network and max flow computation can be done in poly time.

Correctness

* Integral capacities $\Rightarrow$ feasible flow is integral.
* The $[1,1]$ capacity means every good is assigned to exactly one agent, i.e., a valid allocation.
* The $[\theta, m]$ capacity ensures that least-happy agent has value $\geq \theta$.
* Thus, feasible flow corresponds to a desired feasible allocation.

Problem 4

Problem 4 [15 points]

Given a directed graph $G = (V, E)$ and any pair of vertices $u, v \in V$, let $\texttt{dist}(u, v)$ denote the length of the shortest directed path from u to v in G. If no such path exists, then define $\texttt{dist}(u, v) = +\infty$.

By means of an efficient algorithm, show that in any directed graph, there exists an *independent set* of vertices that can reach every other vertex in at most two steps. That is, design a polynomial-time algorithm that, given as input a directed graph $G = (V, E)$, computes a subset of vertices $S \subseteq V$ such that:

1. $\texttt{dist}(u, v) \geqslant 2$ whenever $u, v \in S$ and $u \neq v$, and

2. given any vertex $v \notin S$, there is a vertex $u \in S$ such that $\texttt{dist}(u, v) \leqslant 2$.

Hint: The independent set need not be of the maximum possible size. Can you think of an algorithm that lines up the vertices and computes the desired set by doing left-to-right and/or right-to-left passes?

Algorithm

(1) Consider the vertices in an arbitrary left-to-right order $v_1 v_2 \cdots v_n$.
Mark all vertices as "active".

(2) for $i = 1, 2, \ldots, n$ // left to right pass
    if v_i is active
        for every $j > i$ s.t. $(v_i, v_j) \in E$
            deactivate v_j // deactivate all out-neighbors on the right

(3) for $i = n, n-1, \ldots, 2, 1$ // right to left pass
    if v_i is active
        for every $j < i$ s.t. $(v_i, v_j) \in E$
            deactivate v_j // deactivate all out-neighbors on the left.

return set of active vertices

The algorithm runs in $O(m+n)$ time where $n = \#$ vertices, $m = \#$ edges.

Let $S :=$ set of active vertices returned by the algorithm.

Invariant: A deactivated vertex is never activated again.

Claim 1: S is an independent set.

Proof: Consider any distinct $v_i, v_j \in S$, and suppose $i < j$.

$v_j \in S \Rightarrow (v_i, v_j) \notin E$ (v_j is not an out-neighbor of v_i)

$v_i \in S \Rightarrow (v_j, v_i) \notin E$ (v_i '' '' '' v_j).

$\Rightarrow \mathrm{dist}(v_i, v_j) \geq 2$. $\blacksquare$

Claim 2: For any $v_j \notin S$, $\exists\, v_i \in S$ such that $\text{dist}(v_i, v_j) \leq 2$.

Proof: by case analysis, depending on when v_j is deactivated.

Case I: v_j is deactivated during right-to-left pass

$\Rightarrow$ v_j is deactivated by some v_i s.t. $i > j$

$\Rightarrow$ v_i must remain active throughout, i.e., $v_i \in S$

$\Rightarrow$ $\text{dist}(v_i, v_j) = 1 \leq 2$.

Case II: v_j is deactivated during left-to-right pass

$\Rightarrow$ v_j is deactivated by some v_i s.t. $i < j$

$\Rightarrow$ either $\underbrace{v_i \in S}_{\Downarrow\ \text{dist}(v_i, v_j) = 1}$ or v_i is deactivated during right-to-left pass.

$\exists\, v_k \in S$ s.t. $\text{dist}(v_k, v_i) = 1$ (Case I)

$\Rightarrow$ $\text{dist}(v_k, v_j) = 2$.

$\blacksquare$