ShanghaiTech University

CS240 Algorithm Design and Analysis

Spring 2022

Problem Set 3

Due: 23:59, May 17, 2022

1. Submit your solutions to Gradescope (www.gradescope.com).
2. In “Account Settings” in Gradescope, set your FULL NAME to your Chinese name.
3. If you want to submit a handwritten version, scan it clearly.
4. When submitting your homework in Gradescope, match each of your solutions to the corresponding problem number.
5. Late homeworks submitted within 24 hours of the due date will be marked down 25%, submitted within 48 hours will be marked down 50%, and will not be accepted past 48 hours.
6. You are required to follow ShanghaiTech’s academic honesty policies. You are allowed to discuss problems with other students, but you must write up your solutions by yourself. You are not allowed to copy materials from other students or from online or published resources. Violating academic honesty can result in serious sanctions.

Problem 1:

There is a room with m people who all want to charge their phones. Each person has one phone, and each phone uses a specific type of plug. The room has n different electric outlets, and for each type of plug, the room has either 0 or 1 outlet of that type. The room also contains electrical adapters which allow changing between plug types. Only certain kinds of adapters are available, but for each kind there are an infinite number of adapters. In addition, the adapters can be plugged into each other to enable a sequence of plug type changes. Design an algorithm based on max-flow to maximize the number of phones which can be plugged in, either directly or using a series of adapters.

As an example, suppose there are 5 phones, with plug types B, B, B, C, X, and the room has 4 outlets with types A, B, C, D. In addition, there are 3 kinds of adapters, $B \to X, X \to A$ and $X \to D$. Then it is possible to plug in 4 phones. For example, we can plug one B phone into outlet B, another B phone into adapter $B \to X$ into adapter $X \to A$ into outlet A, the C phone into outlet C, and the X phone into adapter $X \to D$ into outlet D. This leaves one B phone not plugged in.

Solution:

The source Node would link to all electronic device Node with the capacity equal to 1.
The electronic device Node should link to all its initial suitable receptacle Node with the capacity equal to 1.
When you got some adapters, link X to S as the example describes, with the capacity equal to INF, for the enough amounts of adapters.
Finally link all receptacles Nodes to sink Node with capacity to 1.
Calculate the max flow, and the answer is m - mf.

Problem 2:

You own a factory with m machines and you need to finish n tasks. The i'th task can only start on or after day S_i, needs to be processed for a total of P_i days, and must be finished on or before day E_i. Each machine can only work on one task at a time, and each task can be processed by at most one machine at a time. However, a task can be interrupted and processed on different machines on different days. Design an algorithm based on max flow to determine whether it is possible to finish all the tasks.

Solution:

The source Node would link to Task Node with the capacity equal to P_i. Connect the task Node to each day Node of between when it starts and when it ends with capacity to 1.
And due to it has only M machines, connect each day Node to the sink Node with the capacity equal to M.

Problem 3:

You own m bakeries, and n customers want to buy cakes from you. On day i, $1 \leq i \leq n$, customer i wants to buy a total of c_i cakes. To do this, he visits a set B_i out of the m stores, and can buy cakes from any of them. You know ahead of time which customers will visit which stores, i.e. you know $B_1, ..., B_n$. Also, after customer i is done buying cakes, you can redistribute cakes from the stores in B_i among each other. Each store j starts with b_j cakes initially. Design an algorithm based on max-flow to maximize the total number of cakes you sell. That is, you need to decide for each day how many cakes to sell from each store and how to redistribute the cakes.

As an example, suppose there are 3 stores A, B, and C with 3, 1 and 10 cakes initially. Also, suppose there are 3 customers, where customer 1 visits stores A, B and wants 2 cakes, customer 2 visits stores A, C and wants 3 cakes, and customer 3 visits store B and wants 6 cakes. Then by selling 2 cakes from A and 1 cake from B to customer 1, then redistributing 2 cakes from A to B, then selling 3 cakes from C to customer 2, then selling 2 cakes from B to customer 3, you can sell a total of 7 cakes.

Solution:

The source Node would link to all storehouses Node with the capacity equal to each one's initial storage.
Each storehouses which be connected to official should have a INF capacity edge to the official node.
The official node should have a edge whose capacity is the maximum amount units of food, to the sink Node.
Due to the inheriting mechanism, the storehouses opened at the same time should be linked through the official Node with INF capacity edges, meaning any amounts of food could be transported between storehouses.

Problem 4:

Consider the 4 complexity classes listed below. Give as many inclusion relationships between them as you can. That is, give relationships of the form $X \star Y$, where X and Y are complexity classes, and $\star$ is either $\subset$, $\subseteq$ or $=$.

(1) NP (2) NP-hard (3) NP-complete (4) P

Solution:

$P \subseteq NP, NP - complete \subseteq NP, NP - complete \subset NP - hard.$

or, divide into two situations: P=NP and $P \neq NP$

Problem 5:

In the optimization version of the Knapsack problem, we are given n items, where the j'th item has weight w_j and value v_j, and a weight limit W. All values are positive integers. We need to find a subset of items with total weight at most W and which has the maximum value.

Reformulate Knapsack as a decision problem, and show that the problem is in NP.

Solution:

The decision problem corresponding to Knapsack is defined as follows. Given an instance of Knapsack and a threshold y, does there exist a valid solution of Knapsack with objective value $\geq$ y?

A 'yes' solution can be encoded in an array in O(n) space: just put a '1' if the item is present in the solution, and a '0' otherwise. Checking the feasibility requires adding the sizes of the items in the solution and comparing this to B; this can be done in O(n) time. Checking whether the answer is 'yes' requires adding the values of the items in the solution and comparing this to y; this can be done in O(n) time.

Problem 6:

In the HITTING-SET problem, we are given sets $S_1, S_2, ..., S_m$, where each S_i is a subset of $V = \{1, ..., n\}$, and a parameter $k > 0$. We need to determine whether there is a subset $H \subseteq V$ with $|H| \leq k$ such that $H \cap S_i \neq \emptyset$ for all i.

Show that HITTING-SET is NP-complete.

Solution:

1. Prove HITTING-SET problem is NP:

For a given subset H, we could verify whether $|H| \leq k$ and $H \cap S_i \neq \emptyset$ for all i in polynomial time, so this problem is in NP.

2.VERTEX COVER could reduce to the HITTING-SET problem in polynomial time:

construction: formally VERTEX COVER problem is given a graph G(V,E), we need to find a subset of vertices $V^{'}$ such that for $|V^{'}| \leq k$ and all edges e=(u,v), we have u$\in V^{'}$ or v$\in V^{'}$.

For every edge e=(u,v), we can construct a set $S_e = \{u, v\}$, we will have total number of set $|S| = |E| = m$ and we set budget b=k. The construction of the set will take O($|E|$) time, which is in polynomial time.

3.HITTING SET problem is satisfied if and only if the VERTEX COVER problem is satisfied.

$\Rightarrow$: if we have a solution subset H of HITTING SET, we could construct a graph G(V,E) according to the above rule, each edge is constructed by the set S_e. so $|H| \leq k$ and $H \cap S_i \neq \emptyset$ for all i is equivalent to the $|V^{'}| \leq k$ and all edges e=(u,v), we have u$\in V^{'}$ or v$\in V^{'}$. Thus, VERTEX COVER is satisfied.

$\Leftarrow$: if we have a solution $V^{'}$ of the VERTEX COVER problem. so There is an intersection between $V^{'}$ and each edge, which means that H$\cap S_e \neq \emptyset$. Thus, HITTING SET is satisfied.