CS 6150: HW3 – Greedy algorithms, Graphs basics

Submission date: Monday, Oct 25, 2021 (11:59 PM)

This assignment has 5 questions, for a total of 50 points. Unless otherwise specified, complete and reasoned arguments will be expected for all answers.

Question	Points	Score
Set Cover Analysis	14	
Be lazy or be greedy?	11	
Finding triangles	9	
Road tripping	10	
Santa's tradeoffs	6	
Total:	50	

Instructions. For all problems in which you are asked to develop an algorithm, write down the pseudocode, along with a rough argument for correctness and an analysis of the running time (unless specified otherwise). Failure to do this may result in a penalty. If you are unsure how much detail to provide, please contact the instructors on Piazza.

Question 1: Set Cover Analysis ..... **[14]**

In class, we saw the set cover problem (phrased as picking the smallest set of people who cover a given set of skills). Formally, we have n people, and each person has a subset of m skills. Let the set of skill set of the ith person be denoted by S_i, which is a subset of $[m]$ (shorthand for $\{1, 2, \ldots, m\}$). We analyzed a greedy algorithm that at every step, selects the person with the largest number of *uncovered* skills.

(a) **[2]** First, let us show that our analysis cannot be improved significantly. Consider the instance given by Figure 1 (with $n = 5$ people and $m = 26$), and write down (i) the optimal solution, and (ii) the solution output by the greedy algorithm.
Solution.

- The optimal solution. $[1, 2]$
- The solution output by the greedy algorithm, $[5, 4, 3]$

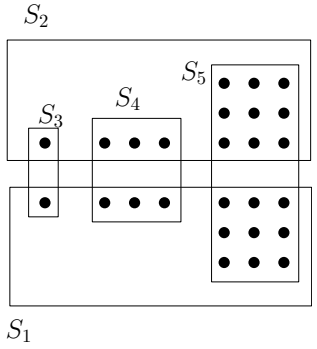

Figure 1: Every dot represents a skill, the rectangles show the skill sets of the people 1, 2, ..., 5.

(b) **[6]** Using Figure 1 as a hint, describe an instance in which the ratio of the size of the solution produced by the greedy algorithm and the size of the optimal solution is $\Omega(\log m)$.
Solution.
There still exists 2 people ($[1, 2]$) which even get $\frac{m}{2}$ unique skills. For other people, each gets $\frac{m+2}{2}, \frac{m+2}{4}, \ldots, 1$ skills. The number of people is,

$$(m+2)(1 - \frac{1}{2^k}) = m \implies k = \log_2(\frac{m+2}{2}) \tag{1}$$

Their skills are evenly distributed to the $[1, 2]$ just like the graph in above question. If we use optimal algorithm, we only need 2 people. If we apply greedy algorithm, we will need k people. So the ratio is $\frac{k}{2} = \Omega(\log(m))$.

(c) **[6]** Next, let us see what happens if we stop the algorithm "mid way". Suppose we are given a parameter k, and we are told that there is a set of k people whose skill sets cover all of $[m]$. Now, suppose we run the greedy algorithm for exactly k iterations. So

the algorithm chooses exactly k people, but it could be that some of the skills are still uncovered. Prove, however, that at least 50% of the skills are covered. [*Hint:* Modify the proof we saw in class.]

Solution.
Using the greedy algorithm, the first person should cover at least $\frac{m}{k}$ skills, the next person should cover $m(1-\frac{1}{k})\frac{1}{k}$. So the uncover skills for each step at most after k steps are

$$u_t < m(1-\frac{1}{k})^k \approx \frac{m}{e} < \frac{m}{2} \tag{2}$$

Question 2: Be lazy or be greedy? ..... **[11]**

Consider the following "street surveillance" problem. We have a graph (V, E) with $|V| = n$ nodes and $|E| = m$ edges. We are allowed to place surveillance cameras at the nodes. Once placed, they can monitor all the edges incident to the node. The goal is to place as few cameras as possible, so as to monitor **all the edges** in the graph.

(a) **[4]** Show how to "cast" the street surveillance problem as an instance of Set Cover, and write down the greedy algorithm. (You don't need to provide any analysis.)
Solution.
All edges must be selected, and the objective is to selected fewer nodes. Just like all skills must be covered, and the objective is to selected fewer people.

(b) **[7]** Let (V, E) be a graph as above, and now consider the following "lazy" algorithm:

1. initialize $S = \emptyset$
2. while there is an unmonitored edge $\{i, j\}$:
   add both i, j to S and mark all their edges as monitored

Clearly (due to the while loop), the algorithm returns a set S that monitors all the edges. Prove that this set satisfies $|S| \leq 2k$, where k is the size of the optimal solution (i.e., the minimum number of nodes we need to place cameras at in order to monitor all the edges).

MORAL. Even though the algorithm looks "dumber" than the greedy algorithm, it has a better approximation guarantee — 2 versus $\log n$.

Hint. Consider the edges $\{i, j\}$ encountered when we run the algorithm. Could it be that the optimal set chooses *neither* of $\{i, j\}$?
Solution.
According to the algorithm, all edges selected are unmonitored, so all chosen edges are disconnected. It means that all neighboring edges of node i, j except i, j will not be chosen. So the optimal solution will at least choose one of node i, j. So that

$$k \geq \frac{|S|}{2} \implies |S| \leq 2k \tag{3}$$

Question 3: Finding triangles ..... **[9]**

We will now see how for some graph problems, "non-standard" approaches can sometimes yield efficient algorithms. Let (V, E) be the set of vertices and edges in a directed graph, respectively.

Our goal is to find out if the graph has a *triangle*, i.e., if there exist $i, j, k \in V$ such that $(i, j), (j, k)$ and (k, i) all belong to E.

(a) [**3**] Show how to do this in time $O\left(\sum_{i \in V} d_{in}(i) d_{out}(i)\right)$, where d_{in} and d_{out} refer to the in-degree and out-degree of a vertex respectively.

Solution.

Just to check whether out node and in node will form a directed edge E or not. Below is the pseudo code. There are 3 loops in the algorithm. Obviously, the running time is $O\left(\sum_{i \in V} d_{in}(i) d_{out}(i)\right)$.

Algorithm 1 Finding triangles

```
Require: Directed graph G of (V, E)
  for each node i in V do
    for each node j in in-degree d_in(i) do
      for each node k in out-degree d_out(i) do
        if edge (k, j) in E then
          return True
        end if
      end for
    end for
  end for
  return False
```

(b) [**6**] Of course, the above reduces to $O(n^3)$ if the graph is dense and d_{in} and d_{out} are $\Theta(n)$ for many vertices. However, assuming that two $n \times n$ matrices can be multiplied in time $O(n^{2.4})$ (which is known to be true), show how to solve the problem in time $O(n^{2.4})$. [*Hint:* Consider the adjacency matrix of the graph A. What can you say about the entries of $A \cdot A$? Can you use this matrix to then check the existence of a triangle?]

Solution.

Non-zero entries (i, j) of $A \cdot A$ mean there exists a node k such that directed edge (i, k) and (k, j) both exist. Then we need to check if (j, i) exists or not to find a triangle. Specifically, we could compute $A \cdot A$ and transpose it. Then we compare it with A and if they share a non-zero entry in the same place, we would say there exist a triangle. This comparison only take $O(n^2)$, the running time is still $O(n^{2.4})$.

Question 4: Road tripping ..... **[10]**

Let us roll back to the days when a road trip meant writing all your favorite music onto CDs and carrying them along.

Suppose we have n songs, of durations $d_1, d_2, \ldots, d_n$, where each d_i is less than one hour. Suppose that every CD can hold one hour of music. The goal is to write all the songs to a set of CDs, so as to minimize the number of CDs required.

Consider the natural greedy algorithm for the problem: at every step, we have a list of CDs containing the music written so far (initially empty). Now consider the songs in the given order, and for song i, write it to the first CD so far that has sufficient space for the song. If there is no such CD, then we buy a new CD, and write song i to it (and add it to the CD list; we are assuming that we can keep adding songs to a CD as long as there is space).

(a) [**3**] Give an example in which the greedy algorithm is not optimal. I.e., give a set of song durations for which the above procedure uses more CDs than an optimal "packing".

Solution.

The duration are $[15, 15, 45, 45]$, the optimal solution is 2 CDs. However, we need 3 if we use greedy selection.

(b) [**7**] Suppose you are told that all the song durations are at most 20 minutes. Now prove that the greedy algorithm is within a factor 3/2 of the "best possible" number of CDs. In other words, prove that if OPT denotes the number of CDs in the best-possible solution, then the algorithm uses $\leq (1.5 \cdot \text{OPT})$ CDs. You will get full credit even if your bound is off by an additive constant, e.g., you prove $\leq (1.5 \cdot \text{OPT} + 2)$. [*Hint:* You may want to compare the number of CDs used by the algorithm with the "total duration" of the songs. Also, play around with small examples to see how the algorithm is doing.]

Solution.
Say, currently, the greedy algorithm gives solution that we need k CDs, Since each song is less than 20 minutes, then there are at least $k-1$ CDs which contains songs more than 40 minutes. The intuition is that if a CD is less than 40 minutes' song, we still can add more songs to it according to greedy algorithm. There may only exists one CDs with less than 40 minutes. So we have,

$$40 * (k-1) < total - duration < 60OPT \implies k < \frac{3}{2}OPT + 1 \tag{4}$$

So the greedy algorithm is within a factor 3/2 of the "best possible" number of CDs.

Question 5: Santa's tradeoffs ..... **[6]**

Recall the matching problem we saw in class: there are n gifts, and n children, and each child has a non-negative valuation for each gift. Formally, the value of gift j to child i is given by $H_{i,j}$. We assume that all $H_{i,j} \geq 0$. Santa's goal is to give one gift to each child, so as to maximize the *total value* (of course, a gift cannot be given to more than one child).

Suppose we now perform a more elaborate local search, this time picking every *triple* of edges in the current solution, and seeing if there is a reassignment of gifts between the end points of these edges that can improve the total value. Prove that a locally optimal solution produced this way has a value that is at least (2/3) the optimum value. [This kind of a trade-off is typical in local search – each iteration is now more expensive $O(n^3)$ instead of $O(n^2)$, but the approximation ratio is better.]

Note. We will see in the coming lectures that this problem can indeed be solved optimally in polynomial time, using a rather different algorithm.

Solution.
Say that we get a locally optimal solution σ which have value of V, and we denote the optimal value as OPT we have,

$$\begin{aligned} H_{i,\sigma(i)} + H_{j,\sigma(j)} + H_{k,\sigma(k)} &\geq H_{i,\pi(i)} + H_{j,\pi(j)} + H_{k,\sigma(i)} \\ &\geq H_{i,\pi(i)} + H_{j,\pi(j)} \end{aligned} \tag{5}$$

where $\pi(i) = \sigma(j)$, $\pi(j) = \sigma(k)$ Then we could have,

$$H_{i,\sigma(i)} + H_{\sigma^{-1}(\pi(i)),\pi(i)} + H_{\sigma^{-1}\pi(\sigma^{-1}\pi(i)),\pi(\sigma^{-1}\pi(i))} \geq H_{i,\pi(i)} + H_{\sigma^{-1}(\pi(i)),\pi(\sigma^{-1}(\pi(i)))} \tag{6}$$

Then we do summation,

$$\sum_i H_{i,\sigma(i)} + H_{\sigma^{-1}(\pi(i)),\pi(i)} + H_{\sigma^{-1}\pi(\sigma^{-1}\pi(i)),\pi(\sigma^{-1}\pi(i))} \geq \sum_i H_{i,\pi(i)} + H_{\sigma^{-1}(\pi(i)),\pi(\sigma^{-1}(\pi(i)))}$$

$$\begin{aligned} 3V &\geq 2 * OPT \\ V &\geq \frac{2}{3} OPT \end{aligned} \tag{7}$$

We can get a locally optimal solution produced this way has a value that is at least (2/3) the optimum value.