CS 577 # Assignment 3: Greedy Algorithms Fall 2022

Answer the questions in the boxes provided on the question sheets. If you run out of room for an answer, add a page to the end of the document.

Name: Yongsang Park Wisc id: 908 022 3069

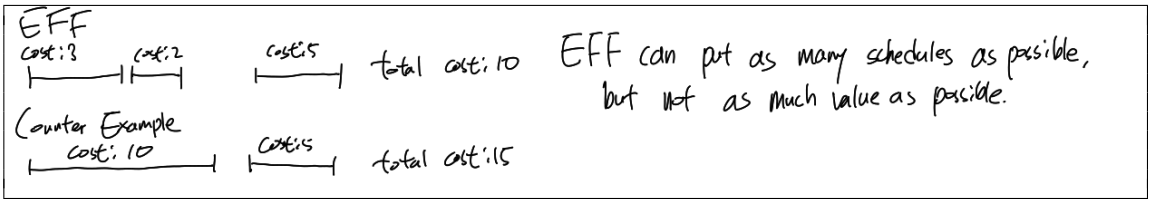
Greedy Algorithms

1. In one or two sentences, describe what a greedy algorithm is. Your definition should be informal, something you could share with a non computer scientist.

A greedy is an algorithm that chooses the best choice in each choice to make the best output which could not be the best.

2. There are many different problems all described as "scheduling" problems. In the following questions, pay attention to the details of the problem setup, as they will change each time!

   (a) Let each job have a start time, an end time, and a value. We want to schedule as much value of non-conflicting jobs as possible. Use a counterexample to show that Earliest Finish First (the greedy algorithm we used for jobs with all equal value) does NOT work in this case.



   (b) *Kleinberg, Jon. Algorithm Design (p. 191, q. 7)* Now let each job consist of two durations. A job i must be preprocessed for p_i time on a supercomputer, and then finished for f_i time on a standard PC. There are enough PCs available to run all jobs at the same time, but there is only one supercomputer (which can only run a single job at a time). The completion time of a schedule is defined as the earliest time when all jobs are done running on both the supercomputer and the PCs. Give a polynomial time algorithm that finds a schedule with the earliest completion time possible.

Let J be the set of job.
Let S be an initially empty set.
While J is not empty do
  Choose $i \in J$ with the largest f_i
    if f_i ties, choose the largest p_i first
  append i to S
end
return S

(c) Prove the correctness and efficiency of your algorithm from part (c).

Correctness

The supercomputer is 1, so the running time of the supercomputer is the same even the work time is changed.

I will use exchange argument to prove my algorithm.

Let S be an schedule.

Even you change the schedule, the total p_i does not change.

$S = S'$ for p_i

The thing we need to concern is f_i. Let's assume $f_a > f_{a+1}$. If f_a starts first, f_a is running while p_{a+1} runs, so $f_a - p_{a+1}$ runs when f_{a+1} runs. But if f_{a+1} runs first. f_a runs without subtraction, so it makes longer than $f_a - p_{a+1}$ runs. So, the longer one should run first.

Efficiency

while-loop ; n times

choose $i \in I$ with the largest f_i ; n times

$n \times n = n^2$

$O(n^2)$

3. *Kleinberg, Jon. Algorithm Design (p. 190, q. 5)*

(a) Consider a long road with houses scattered along it. We want to place cell phone towers along the road so that every house is within four miles of at least one tower. Give an efficient algorithm that achieves this goal using the minimum possible number of towers.

```
Input: H be the set of houses
Sort H by distance from 0
t_1 (first tower) = h_1 + 4
p = 1 (tower index)
for i = 2 to H.length
    if abs(h_i - t_p) > 4
        p = p+1
        t_p = h_i + 4
    end
end
return p
```

(b) Prove the correctness of your algorithm.

My algorithm is based on the house orders and the service distance.
Let S be any optimal solution. Suppose S has p towers at distance $d_1 < d_2 < \cdots < d_p$.
For $\exists i, j \in \mathbb{N}$, $i < j$, if $d_i + 4 > d_j - 4$, it contradicts of being optimal.

$S(n) = p$

Basecase. 1 house - 1 tower

Induction. Suppose $S(k) = p'$ is optimal.

For $S(k+1)$,

Case 1. h_{k+1} is within $t_{p'}$. ('h' is a house, 't' is a tower.)

$S(k+1) = p'$ is still optimal because h_{k+1} is under t_p area.

Case 2. h_{k+1} is not under $t_{p'}$ area. (When h_{k+2} comes, h_{k+2} should be larger than h_{k+1})

$S(k+1) = p'+1$ with maximum area. So it is optimal. ∎

4. *Kleinberg, Jon. Algorithm Design (p. 197, q. 18)* Your friends are planning to drive north from Madison to the town of Superior, Wisconsin over winter break. They have drawn a directed graph with nodes representing potential stops and edges representing the roads between them.

   They have also found a weather forecasting site that can accurately predict how long it will take to traverse one of the edges on their graph, given the starting time t. This is important because some of the roads on their graph are affected strongly by the seasons and by extreme weather. It's guaranteed that it never takes negative time to traverse an edge, and that you can never arrive earlier by starting later.

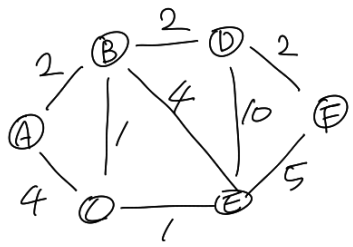
   (a) Design an algorithm your friends can use to plot the quickest route. You may assume that they start at time $t = 0$, and that the predictions made by the weather forecasting site are accurate.

```
Input: Graph G=(V,E)
Set all values of distance to ∞ of the graph.
While G is not empty
    get the minimum distance of u∈V.
    for all edges (u,v) ∈ E
        if the distance of v > distance of u + weight of (u,v)
            distance of v = distance of u + weight of (u,v)
        end
    end
end
```

(b) Demonstrate how your algorithm works using a small example with 6 nodes. Your demonstration should include any data structures you maintain during the execution of your algorithm and any queries you make to the weather forecasting site. For example, if your algorithm maintains a "current path" that grows from (M)adison to (S)uperior, you might show something like the following table:

Path	Total time
M	0
M,A	2
M,A,E	5
M,A,E,F	6
M,A,E	5
M,A,E,H	10
M,A,E,H,S	13

Example graph



start end

	A	B	C	D	E	F
Value of distance	0	∞	∞	∞	∞	∞
	0	2	4	∞	∞	∞
	0	2	3	4	6	∞
	0	2	3	4	4	∞
	0	2	3	4	4	6
	0	2	3	4	4	6
	0	2	3	4	4	6

blue - chosen node
red - updated value
green - used node

⇒ Minimum time to get F from A: 6

Path: A, B, D, F

Coding Question

5. Implement the optimal algorithm for interval scheduling (for a definition of the problem, see the Greedy slides on Canvas) in either C, C++, C#, Java, or Python. Be efficient and implement it in $O(n \log n)$ time, where n is the number of jobs.

   The input will start with an positive integer, giving the number of instances that follow. For each instance, there will be a positive integer, giving the number of jobs. For each job, there will be a pair of positive integers i and j, where $i < j$, and i is the start time, and j is the end time.

   A sample input is the following:

   ```
   2
   1
   1 4
   3
   1 2
   3 4
   2 6
   ```

   The sample input has two instances. The first instance has one job to schedule with a start time of 1 and an end time of 4. The second instance has 3 jobs.

   For each instance, your program should output the number of intervals scheduled on a separate line. Each output line should be terminated by a newline. The correct output to the sample input would be:

   ```
   1
   2
   ```