CSCI 3104 Fall 2021
Instructor: Profs. Grochow and Waggoner

# Problem Set 2

Due Date ........ **September 14, 2021**
Name ........ **Abeal Sileshi**
Student ID ........ **104326364**
Collaborators ........ **Preston Windefelt**

## Contents



## 1 Instructions

- The solutions **must be typed**, using proper mathematical notation. We cannot accept hand-written solutions. Here's a short intro to LaTeX.
- You should submit your work through the **class Canvas page** only. Please submit one PDF file, compiled using this LaTeX template.
- You may not need a full page for your solutions; pagebreaks are there to help Gradescope automatically find where each problem is. Even if you do not attempt every problem, please submit this document with no fewer pages than the blank template (or Gradescope has issues with it).

- You are welcome and encouraged to collaborate with your classmates, as well as consult outside resources. You must **cite your sources in this document. Copying from any source is an Honor Code violation. Furthermore, all submissions must be in your own words and reflect your understanding of the material.** If there is any confusion about this policy, it is your responsibility to clarify before the due date.
- Posting to **any** service including, but not limited to Chegg, Reddit, StackExchange, etc., for help on an assignment is a violation of the Honor Code.
- You **must** virtually sign the Honor Code (see Section 2). Failure to do so will result in your assignment not being graded.

## 2 Honor Code (Make Sure to Virtually Sign)

**Problem 1.**

- My submission is in my own words and reflects my understanding of the material.
- Any collaborations and external sources have been clearly cited in this document.
- I have not posted to external services including, but not limited to Chegg, Reddit, StackExchange, etc.
- I have neither copied nor provided others solutions they can copy.

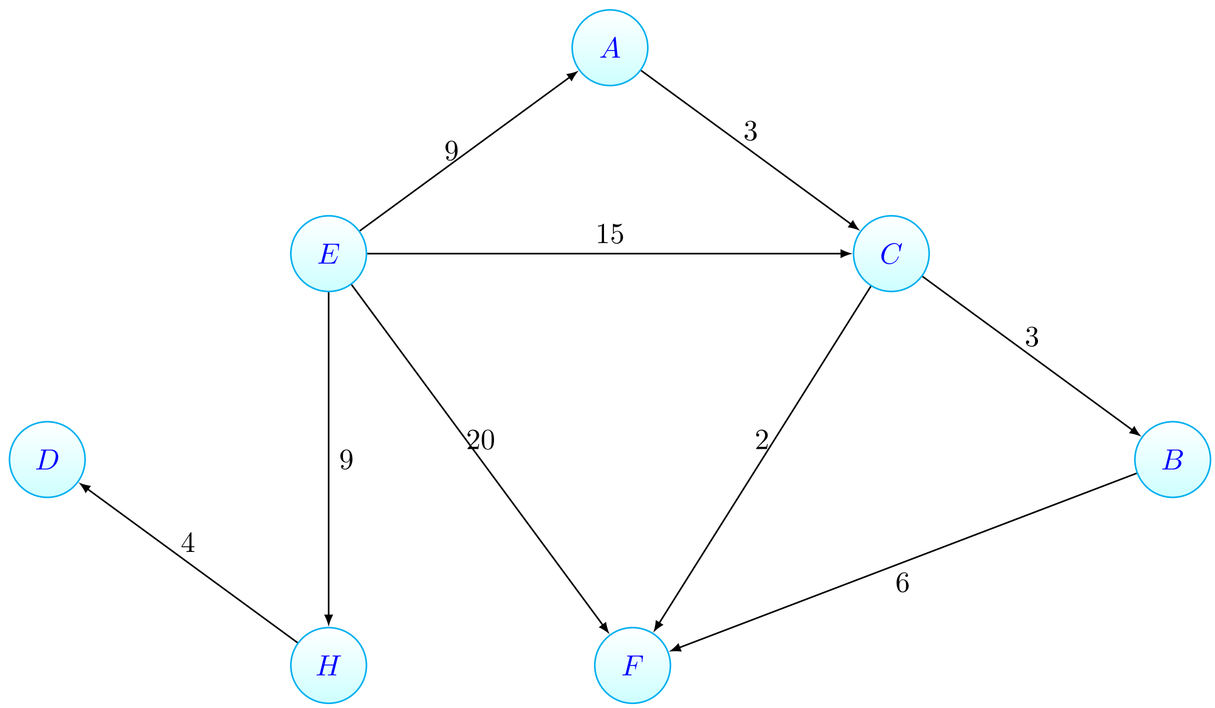
*Agreed (signature here).* I agree to the above, Abeal Sileshi ☐

## 3 Standard 3- Dijkstra's Algorithm

### 3.1 Problem 2

**Problem 2.** Consider the weighted graph $G(V, E, w)$ pictured below. Work through Dijkstra's algorithm on the following graph, using the source vertex $E$.

- Clearly include the contents of the priority queue, as well as the distance from $E$ to each vertex at each iteration.
- If you use a table to store the distances, clearly label the keys according to the vertex names rather than numeric indices (i.e., `dist['B']` is more descriptive than `dist['1']`).
- You do **not** need to draw the graph at each iteration, though you are welcome to do so. [This may be helpful scratch work, which you do not need to include.]




*Answer.* My algorithm comes from chapter 24 of the textbook. Which provides pseudocode for Dijkstra's Algorithm.
$\varnothing \in \notin \cup \{ \infty$
To begin we initialize all vertices to have distance $\infty$ and our source s, (E in this example), to have distance 0.
for each vertex v $\in$ G.V
    v.d = $\infty$
s.d = 0
S, our solution set doesn't currently contain anything. S = $\varnothing$

Q, is our queue, it is currently full
Q = { (E,0), (A, $\infty$), (C, $\infty$), (D, $\infty$), (F, $\infty$), (B, $\infty$) }

while Q is not null we extract min, add it to the solution set, and relax (see, page 649 in textbook) all the neighbor nodes
G

□

Initialize-Single-Source $(G, s)$
for each vertex $v \in G.V$
    $v.d = \infty$
    $v.\pi = NIL$
$s.d = 0$

Relax $(u, v, w)$
if $v.d > u.d + w(u,v)$
    $v.d = u.d + w(u,v)$
    $v.\pi = u$

Dijkstra $(G, w, s)$
Initialize-Single-Source $(G, s)$
$S = \emptyset$
$Q = G.V$ $\quad (Q = V - S)$
while $Q \neq \emptyset$
    $u =$ Extract-Min$(Q)$ $\Rightarrow u = (E, 0)$
    $S = S \cup \{u\}$ $\Rightarrow S = \{(E, 0)\}$
    for each vertex $v \in G.Adj[u]$
        Relax $(u, v, w)$

for each vertex $v \in G.Adj[u]$
$G.Adj[u] = \{(H,\infty), (F,\infty), (C,\infty), (A,\infty)\}$
$v = (H, \infty)$
if $v.d > u.d + w(u,v)$
$\infty > 0 + 9$
$v.d = 9$
$v = (H, 9)$
next neighbor
$v = (F, \infty)$
$\infty > 0 + 20$
$v.d = 20$
$v = (F, 20)$
next
$v = (C, \infty)$
$\infty > 0 + 15$
$v.d = 15$
$v = (C, 15)$

next
$v = (A, \infty)$
$\infty > 0 + 9$
$v.d = 9$
$v = (A, 9)$
$Q = \{(H,9), (F,20), (C,15), (A,9), (D,\infty), (B,\infty)\}$

$u = (A, 9)$
$S = \{(E,0), (A,9)\}$
for each vertex $v \in G.Adj[(A,9)]$
$v = (C, \infty)$
if $v.d > u.d + w(u,v)$
$\infty > 9 + 3$
$v.d = 12$
$Q = \{(H,9), (F,20), (C,12), (D,\infty), (B,\infty)\}$
$Q \neq \emptyset$
$u = (H, 9)$
$S = \{(E,0), (A,9), (H,9)\}$
for each vertex $v \in G.Adj[(H,9)]$
$v = (D, \infty)$
$\infty > 9 + 4$
$v.d = 13$
$Q = \{(F,20), (C,12), (D,13), (B,\infty)\}$
$u = (C, 12)$
$S = \{(E,0), (A,9), (H,9), (C,12)\}$
for each vertex $v \in G.Adj[(C,12)]$
$v = (B, \infty)$
$\infty > 12 + 3$
$v.d = 15$
$Q = \{(F,20), (B,15), (D,13)\}$
$v = (F, 20)$
$20 > 12 + 2$
$v.d = 14$, 14)
$Q = \{(D,13), (B,15), (F,14)\}$
$Q \neq 0$
$u = (D, 13)$
$S = \{(E,0), (A,9), (H,9), (C,12), (D,13)\}$
for each vertex $v \in G.Adj[(D,13)]$
$\rightarrow$ no neighbors
$Q = \{(B,15), (F,14)\}$
$u = (F, 14)$ $\rightarrow S = \{(F,14) \ldots\ldots\}$
for each....
$\rightarrow$ no neighbors
$Q = \{(B,15)\}$
$u = \{(B,15)\}$

$S = \{(E,0), (A,9), (H,9), (C,12), (D,13), (F,14), (B,15)\}$
for each vertex $\in G.Adj[(B,15)]$
$v = (F, 14)$
$14 > 15 + 6$
$14 > 21$
$v.d$ is NOT updated
$Q = \emptyset$
Algorithm over!

## 3.2 Problem 3

**Problem 3.** You have three batteries, with 4200, 2700, and 1600 mAh (milli-Amp-hours), respectively. The 2700 and 1600-mAh batteries are fully charged (containing 2700 mAh and 1600 mAh, respectively), while the 4200-mAh battery is empty, with 0 mAh. You have a battery transfer device which has a "source" battery position and a "target" battery position. When you place two batteries in the device, it instantaneously transfers as many mAh from the source battery to the target battery as possible. Thus, this device stops the transfer either when the source battery has no mAh remaining or when the destination battery is fully charged (whichever comes first).

But battery transfers aren't free! The battery device is also hooked up to your phone by bluetooth, and automatically charges you a number of cents equal to however many mAh it just transfered.

The goal in this problem is to determine whether there exists a sequence of transfers that leaves exactly 1200 mAh either in the 2700-mAh battery or the 1600-mAh battery, and if so, how little money you can spend to get this result.

Do the following.

### 3.2.1 Problem 3(a)

(a) Rephrase this is as a graph problem. Give a precise definition of how to model this problem as a graph, and state the specific question about this graph that must be answered. [**Note:** While you are welcome to draw the graph, it is enough to provide 1-2 sentences clearly describing what the vertices are and when two vertices are adjacent. If the graph is weighted, clearly specify what the edge weights are.]

*Answer.* This is a graph where the edges are the transfers of energy and the nodes are the states (i.e. battery 1 has 0 mAh, battery 2 has 2700 mAh, and battery 3 has 1600 mAh. The adjacent vertices would be a transfer of energy that directly happens, with the edge would be the number of mAh being transfered. □

### 3.2.2 Problem 3(b)

(b) Clearly describe an algorithm to solve this problem. If you use an algorithm covered in class, it is enough to state that. If you modify an algorithm from class, clearly outline any modifications. Make sure to explicitly specify any parameters that need to be passed to the initial function call.

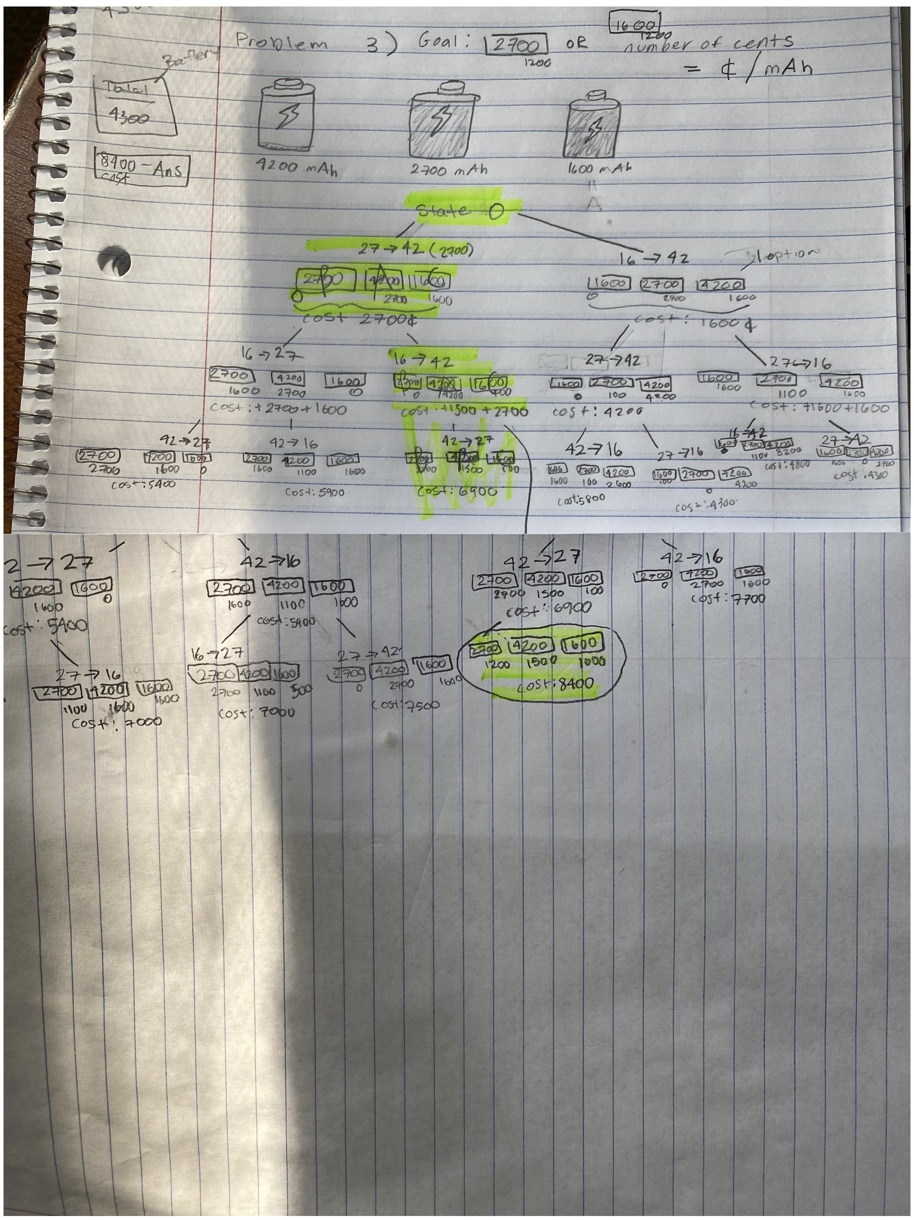
*Answer.* Dijkstra's Algorithm can solve this problem. □

### 3.2.3 Problem 3(c)

(c) Apply that algorithm to the question. Report and justify your answer. Here, justification includes the sequences of vertices visited and the total cost.

*Answer.* Let A = 4200 mAh battery, B = 2700, and C = 1600. With the subscript indicating how much energy the battery contains.

The sequence is as follows:
To start A has no energy, while B and C are fully charged.
$State_0$ : $A_0$ $B_{2700}$ $C_{1600}$ (cost: 0 cents)
$State_1$ : $A_{2700}$ $B_0$ $C_{1600}$ (cost: 2700 cents)
$State_2$ : $A_{4200}$ $B_0$ $C_{100}$ (cost: 4200 cents)
$State_3$ : $A_{1500}$ $B_{2700}$ $C_{100}$ (cost: 6900 cents)
$State_4$ : $A_{1500}$ $B_{1200}$ $C_{1600}$ (cost: 8400 cents) ☐



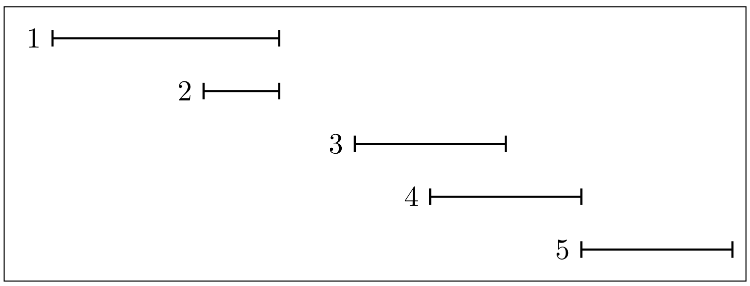
# 4 Standard 4- Examples Where Greedy Algorithms Fail

## 4.1 Problem 4

**Problem 4.** Recall the Interval Scheduling problem, where we take as input a set of intervals $\mathcal{I}$. The goal is to find a maximum-sized set $S \subseteq \mathcal{I}$, where no two intervals in $S$ intersect. Consider the greedy algorithm where we place all of the intervals of $\mathcal{I}$ into a priority queue, ordered earliest start time to latest start time. We then construct a set $S$ by adding intervals to $S$ as we poll them from the priority queue, provided the element we polled does not intersect with any interval already in $S$.

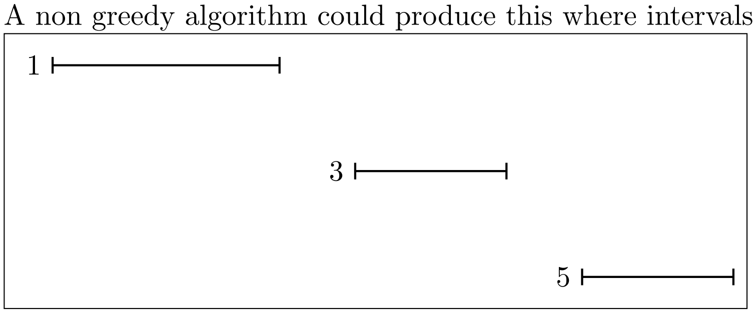
Provide an example with at least 5 intervals where this algorithm fails to yield a maximum-sized set of pairwise non-overlapping intervals. Clearly specify both the set $S$ that the algorithm constructs, as well a larger set of pairwise non-overlapping intervals.

You may explicitly specify the intervals by their start and end times (such as in the examples from class) or by drawing them. **If you draw them, please make it very clear whether two intervals overlap.** You are welcome to hand-draw and embed an image, provided it is legible and we do not have to rotate our screens to grade your work. Your justification should still be typed. If you would prefer to draw the intervals using LaTeX, we have provided sample code below.




*Answer.* A greedy algorithm fails to yield a maximum-sized set of pairwise non-overlapping intervals for intervals: 1(start: 0, end: 3), 2(2,3), 3(4,6), 4(5,7), 5(7,9) □

A non greedy algorithm could produce this where intervals don't overlap



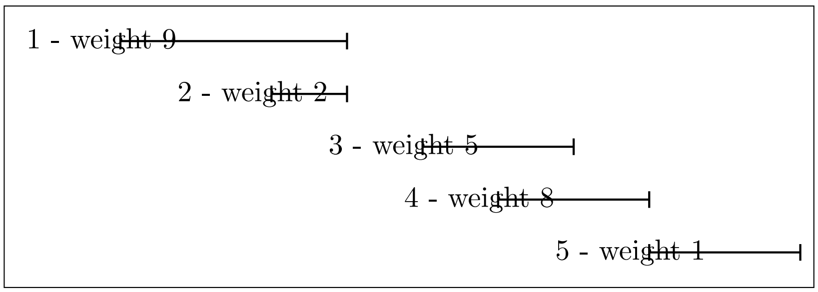
## 4.2 Problem 5

**Problem 5.** Consider now the Weighted Interval Scheduling problem, where each interval $i$ is specified by

$$([\text{start}_i, \text{end}_i], \text{weight}_i).$$

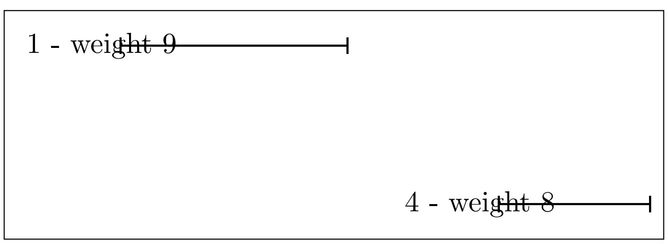
Here, the weight is an assigned value that is independent of the length $\text{end}_i - \text{start}_i$. Here, you may assume $\text{weight}_i > 0$. We seek a set $S$ of pairwise non-overlapping intervals that maximizes $\sum_{i \in S} \text{weight}_i$. That is, rather than maximizing the number of intervals, we are seeking to maximize the sum of the weights.

Consider a greedy algorithm which works identically as in Problem 4. Draw an example with at least 5 appointments where this algorithm fails. Show the order in which the algorithm selects the intervals, and also show a subset with larger weight of non-overlapping intervals than the subset output by the greedy algorithm. The same comments apply here as for Problem 4 in terms of level of explanation.




*Answer.* A greedy algorithm fails to yield a maximum-sized set of pairwise non-overlapping intervals for intervals: 1(start: 0, end: 3, weight: 9), 2(2,3,2), 3(4,6,5), 4(5,7,8), 5(7,9,2) □

*Answer.* In this case the greedy algorithm fails because it doesn't consider the weights, only the times. A non-greedy algorithm may produce this, maximizing weights and without overlaps. □



# 5 Standard 5- Exchange Arguments

## 5.1 Problem 6

**Problem 6.** Recall the Making Change problem, where we have an infinite supply of pennies (worth 1 cent), nickels (worth 5 cents), dimes (worth 10 cents), and quarters (worth 25 cents). We take as input an integer $n \geq 0$. The goal is to make change for $n$ using the fewest number of coins possible.

Prove that in an optimal solution, we use at most 2 dimes.

*Proof.* $\square$

## 5.2 Problem 7

**Problem 7.** Consider the Interval Projection problem, which is defined as follows.

- Instance: Let $\mathcal{I}$ be a set of intervals on the real line.
- Solution: A minimum sized set $S$ of points on the real line, such that (i) for every interval $[s, f] \in \mathcal{I}$, there exists a point $x \in S$ where $x$ is in the interval $[s, f]$. We call $S$ a *projection set*.

Do the following.

### 5.2.1 Problem 7(a)

(a) Find a minimum sized projection set $S$ for the following set of intervals:

$$\mathcal{I} = \{[0, 1], [0.5, 1], [1, 1.5], [1.4, 2], [1.6, 2.3]\}.$$

*Answer.* □

### 5.2.2 Problem 7(b)

(b) Fix a set of intervals $\mathcal{I}$, and let $S$ be a projection set. Prove that there exists a projection set $S'$ such that (i) $|S'| = |S|$, and (ii) where every point $x \in S'$ is the right end-point of some interval $[s, f] \in \mathcal{I}$.

*Proof.* $\square$