Name: Michael Rogers
ID: 105667404

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**
Problem Set 3b (50 points) **Fall 2019, CU-Boulder**

Instructions for submitting your solution:

- The solutions **should be typed** and we cannot accept hand-written solutions. Here's a short intro to Latex.
- You should submit your work through **Gradescope** only.
- If you don't have an account on it, sign up for one using your CU email. You should have gotten an email to sign up. If your name based CU email doesn't work, try the identikey@colorado.edu version.
- Gradescope will only accept **.pdf** files (except for code files that should be submitted separately on Gradescope if a problem set has them) and **try to fit your work in the box provided**.
- You cannot submit a pdf which has less pages than what we provided you as Gradescope won't allow it.
- Verbal reasoning is typically insufficient for full credit. Instead, write a logical argument, in the style of a mathematical proof.
- For every problem in this class, you must justify your answer: show how you arrived at it and why it is correct. If there are assumptions you need to make along the way, state those clearly.
- You may work with other students. However, **all solutions must be written independently and in your own words.** Referencing solutions of any sort is strictly prohibited. You must explicitly cite any sources, as well as any collaborators.

Name: Michael Rogers
ID: 105667404

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**
Problem Set 3b (50 points) **Fall 2019, CU-Boulder**

1. (23 pts) Imagine an alternate reality where CU has a small robot that travels around the campus delivering food to hungry students. The robot starts at the C4C and goes to whatever dorm or classroom has placed the order. The fully-charged battery of the robot has enough energy to travel k meters. On campus, there are n wireless charging pods where the robot can stop to charge its battery. Denote by $l_1 < l_2 < \cdots < l_n$ the locations of the charging pods along the route with l_i the distance from the C4C to the ith charging pod. The distance between neighboring charging pods is assumed to be at most k meters. Your objective is to make as few charging stops as possible along the way.

   (a) (10 pts) Write a python program for an optimal greedy algorithm to determine at which charging pods the robot would stop. Your code should take as input k and a *list* of distances of charging pods (first distance in the list is 0 to represent the start point and the last is the destination and not a pod). Print out the charging pods where the robot stops using your greedy strategy.
   Example 1 - If $\mathbf{k = 40}$ and **Pods** = [**0, 20, 37, 54, 70, 90**]. Number of stops required is 2 and the output should be [**37, 70**].
   Example 2 - If $\mathbf{k = 20}$ and **Pods** = [**0, 18, 21, 24, 37, 56, 66**]. Number of stops required is 3 and the output should be [**18, 37, 56**].
   Example 3 - If $\mathbf{k = 20}$ and **Pods** = [**0, 10, 15, 18**]. Number of stops required is 0 and the output should be [].

   (b) (3 pts) Provide the time complexity of your python algorithm, including an explanation.

```
def chargingRoute(k, pods_lst):
    selected_pods = [0] // O(1)
    for i in range(0,len(pods_lst)): // O(n)
        if pods_lst[i] - selected_pods[-1] > k: // O(1)
            selected_pods.append(pods_lst[i-1]) // O(1)

    selected_pods.pop(0) // O(1)
    return selected_pods // O(1)
```

   Solution. This algorithm runs in $\Theta(n)$ time because the loop runs n times to iterate through in list inputted.

Name: Michael Rogers
ID: 105667404

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**
Problem Set 3b (50 points) **Fall 2019, CU-Boulder**

(c) (10 pts) Prove that your algorithm gives an optimal solution.

Solution. Base Case:
Destination is $\leq$ k - In this case podslst[0:n-1] $\leq$ podslst[-1] $\leq$ k where n is the length of the list. This would cause the if statement to never execute, and the selectedpods list to never hold any values resulting in [], which is the same as the optimal solution, and thus our base case holds.

Inductive Hypothesis:

$$\text{Assume that podslst[i] - selectedpods[-1]} > k.$$

Inductive Step:

Since,

$$\text{podslst[i] - selectedpods[-1]} > k$$

then,

podslst[i-1] - selectedpods[-1] must be $\leq k$, and also must be $\geq$ podslst[i-2] - selectedpods[-1].

Because the input, podslst, is sorted to have $i > i-1$, the algorithm will not stop on an i unless it is $> k$. This implies that podslst[i-1] must be the largest step the robot can make between charges because selectedpods[-1] is the robot's previous stop point. If we extrapolate this throughout the rest of the podslst, then our algorithm holds that it finds the largest step between recharging points. This is the same as the optimal solution, $\therefore$ the algorithm is optimal.

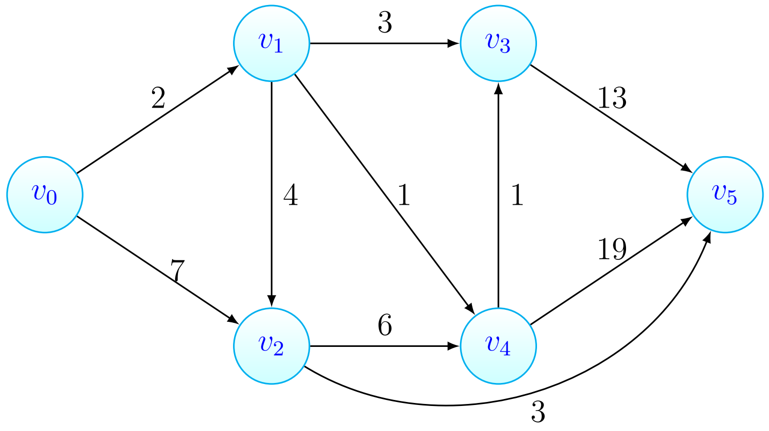
Name: $\boxed{\text{Michael Rogers}}$

ID: $\boxed{105667404}$

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**

Problem Set 3b (50 points) **Fall 2019, CU-Boulder**

2. (7 pts) Using Dijkstra's algorithm, determine the length of the shortest path from v_0 to each of the other vertices in the graph. Clearly specify the distances from v_0 to each vertex **after each iteration** of the algorithm.



First Iteration: $v_0 - v_1 = 2$, $v_0 - v_{2:5} = \infty$

Second Iteration: $v_0 - v_1 = 2$, $v_0 - v_2 = 7$, $v_0 - v_{3:5} = \infty$

Third Iteration: $v_0 - v_1 = 2$, $v_0 - v_2 = 7$, $v_0 - v_3 = 5$, $v_0 - v_{4:5} = \infty$

Fourth Iteration $v_0 - v_1 = 2$, $v_0 - v_2 = 7$, $v_0 - v_3 = 5$, $v_0 - v_4 = 3$, $v_0 - v_5 = \infty$

Fifth Iteration: $v_0 - v_1 = 2$, $v_0 - v_2 = 7$, $v_0 - v_3 = 5$, $v_0 - v_4 = 3$, $v_0 - v_5 = 18$

Name: Michael Rogers
ID: 105667404

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**
Problem Set 3b (50 points) **Fall 2019, CU-Boulder**

3. (20 pts) After years of futility, the Colorado Rockies have decided to try a new approach to signing players. Next year, they have a target number of wins, n, and they want to sign the fewest number of players who can produce exactly those n wins. In this model, each player has a win value of $v_1 < v_2 < \cdots < v_r$ for r player types, where each player's value v_i is a positive integer representing the number of wins he brings to the team. (Note: In a real-world example, All-Star third baseman, Nolan Arenado, contributed 4.5 wins this year beyond what a league-minimum player would have contributed to the team.) The team's goal is to obtain a set of counts $\{d_i\}$, one for each player type (so d_i represents the quantity of players with valuation v_i that are recruited), such that $\sum_{i=1}^{r} d_i = k$ and where k is the number of players signed, and k is minimized.

   (a) (10 pts) Write a greedy algorithm that will produce an optimal solution for a set of player win values of $[1, 2, 4, 8, 16]$ and prove that your algorithm is optimal for those values. Your algorithm need only be optimal for the fixed win values $[1, 2, 4, 8, 16]$. You do **not** need to consider other configuration of win values.

   Solution. Psudocode:

```
def optimal_wins(n):
  win_vals = [1,2,4,8,16]
  remaining = n
  count = 0
  while remaining not equal to 0:
    if win_vals[lastelement] <= remaining:
      remaining -= win_vals[lastelement]
      count += 1
    for i in range(0, len(win_vals)):
      if win_vals[i] == remaining:
          remaining = remaining - win_vals[i]
          count += 1
      elif win_vals[i] > remaining:
          remaining = remaining - win_vals[i-1]
          count += 1
  return count
```

[Additional space for solving Q3a]

Proof:
Base Case: $n = 0$ when the projected wins is 0, the while loop never executes, and the algorithm returns 0 players which is the same as the optimal solution, and thus the base case holds.

Inductive Hypothesis: Assume that at the i^{th} iteration, remaining is equal to zero.

Inductive Step: If remaining is equal to zero at the i^{th} iteration, then that means that at the $i - 1$ iteration, remaining will be equal to some winvals[i]. Per the algorithm, if remaining equals any of the elements in the array, then on the next iteration, remaining will be equal to zero. However, if remaining is in between two numbers in winvals, the for loop will check to see if winvals[i] > remaining, and if so, then it will subtract winvals[i-1] which is the next largest element in the list that is less than remaining. This will result in the largest possible amount being subtracted off of remaining at each iteration. This is the same solution as the optimal, thus the algorithm is optimal.

(b) (10 pts) Find a set of win values where your algorithm does not produce the optimal solution and show where your algorithm fails for those values.

Solution. winvals = [1,10,18] , $n = 20$

This set of numbers will make it so our algorithm will not choose the optimal solution. Because the algorithm chooses the largest element in winvals < remaining the algorithm would choose 18, 1, 1 when in fact the optimal is 10,10.

Name: Michael Rogers
ID: 105667404

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**
Problem Set 3b (50 points) **Fall 2019, CU-Boulder**

Ungraded questions - These questions are for your practice. We won't grade them or provide a typed solution but we are open to discuss these in our OHs and you should take feed backs on your approach during the OHs. These questions are part of the syllabus.

1. Suppose we have a directed graph G, where each edge e_i has a weight $w_i \in (0, 1)$. The weight of a path is the product of the weights of each edge.
   (a) Explain why a version of Dijkstra's algorithm cannot be used here. [**Hint:** We may think about transforming G into a graph H, where the weight of edge i in H is $\ln(w_i)$. It is equivalent to apply Dijkstra's algorithm to H.]
   Solution.
   (b) What conditions does each edge weight w_i need to satisfy, in order to use Dijkstra's algorithm?
   Solution.