Name: Nikolaas Bender
ID: 106977096

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: Nikolaas Bender
ID: 106977096

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 **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 **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 **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.

   The time complexity is $\Theta(n)$ as the driving time complexity factor is the for loop that goes through the whole list of charging pods, all of the rest of the algorithm consists of some if statements and keeping track of variables.

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

My algorithm chooses the furthest charger within the robot's range if the route will be longer than the range of the robot.

base case:
distance to target is 0, the robot makes 0 stops and is optimal.

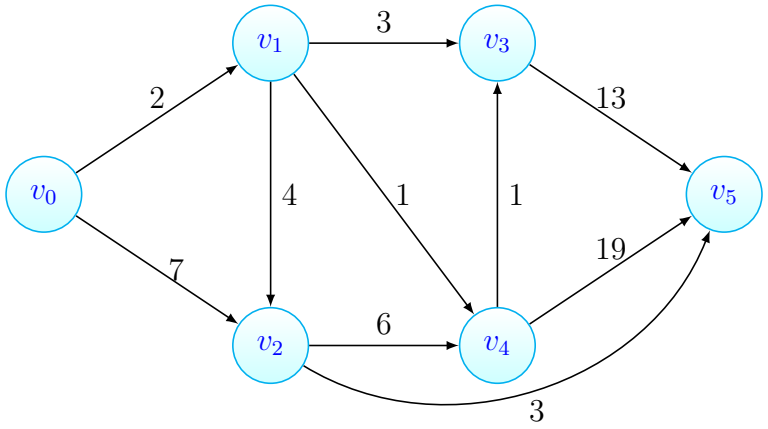
Inductive hypothesis:
at any stop i assume that our greedy algorithm will have chosen at most the same number of stops as the optimal solution.

Inductive step:
at $i+1$ the greedy algorithm chooses the optimal solution using the inductive hypothesis because the greedy algorithm had chosen the optimal number of stops. Then at $i+1$ the greedy algorithm would choose the next largest distance that can be done on a charge. The optimal algorithm can not travel any further than the range of the robot since stop i so the optimal solution will choose the the same stop as the greedy algorithm so all stops are the same of the greedy and optimal algorithm. This proves the greedy algorithm is the optimal algorithm.

Name: Nikolaas Bender
ID: 106977096

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.



iteration
v1 = 2
v2 = 7
iteration
v1 = 2
v2 = 6
v5 = 10
v4 = 3
v3 = 5
iteration
v1 = 2
v2 = 6
v3 = 4
v4 = 3
v5 = 9

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. (this seems like the example of choosing coins for change)

```
def rockie_team(n, lst):
    ret = []
    goal = n
    i = len(lst) - 1
    while i > -1:
        if lst[i] < goal or lst[i] == goal:
            ret.append(lst[i])
            goal = goal - lst[i]
        i -= 1

    return ret
```

   after running a test where I made a loop to set n to values between 0 an 32

```
for i in range(32):
        print(rockie_team(i, [1, 2, 4, 8, 16]))
```

   the output of that test was correct, you can try it. It doesn't fit here.

Using a loop invariant proof:
LI: the algorithm loops through the whole loop from the end to the beginning, adding the largest number in the original list that is less than or equal to the goal to the return list.
INIT: setting up some variables.
matinence: adding the largest element to the return list and subtracting that from goal.
TERM: i == -1 and the algorithm has looped through the whole list.

(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 (0, 1). n = 1
this produces the result of [1, 0]
this is because my greedy algorithm doesn't really respond to players with a 0 value so it will grab all of the terrible players, sorry Rockies (gotta help out my home town team, the Mets).

Name: Nikolaas Bender
ID: 106977096

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.