Name: Alex Book
ID: 108073300
Collaboration: Michael Hasenkamp, Irvin Carbajal, Jamie Foster, Varun Narayanswamy

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: Alex Book
ID: 108073300
Collaboration: Michael Hasenkamp, Irvin Carbajal, Jamie Foster, Varun Narayanswamy

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.
   Solution.
   begin runtime analysis after function definition (after line 1)
   line 2 runs once
   line 3 runs once
   line 4 runs n+1 times
   line 5 runs a maximum of n times
   line 6 runs a maximum of once per iteration of line 5
   line 7 runs a maximum of once per iteration of line 5
   line 8 runs once

   $f(n) = 1 + 1 + (n + 1) + n + n + n + 1$
   $f(n) \in \Theta(n)$

Name: Alex Book
ID: 108073300
Collaboration: Michael Hasenkamp, Irvin Carbajal, Jamie Foster, Varun Narayanswamy

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

Solution.
Let $dist(i, G)$ represent the distance travelled from the start point to stop i by the greedy algorithm. $Letdist(i, O)$ represent the distance travelled from the start point to the stop i by the optimal solution. We need to prove that for all i, $dist(i, G) \geq dist(i, O)$.

Proof by Induction

Base Case: i=1
$dist(1, G) \geq k$ by the problem definition, and it chooses the farthest distance possible that is less than or equal to k. Therefore, it must be true that $dist(1, G) \geq dist(1, O)$. Base case holds.

Inductive Hypothesis:
Assume that for stop $i = s$, $dist(s, G) \geq dist(s, O)$.

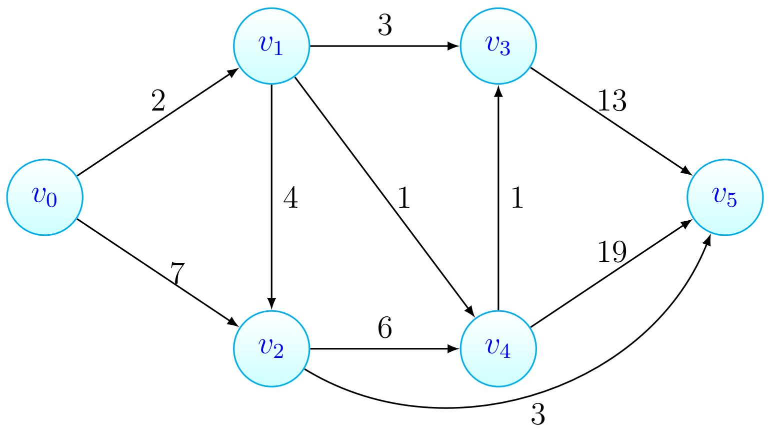
Inductive Step:
For stop $s + l$, the greedy algorithm has travelled the furthest possible distance from stop s such that $dist(s+1, G) - dist(s, G) \leq k$. By the inductive hypothesis, the greedy algorithm was ahead of or even with the optimal solution at stop s. Since the greedy algorithm has travelled the maximum allowed additional distance since that stop, it must still be ahead of or even with the optimal algorithm.
By induction, it is true that for all i, $dist(i, G) \geq dist(i, O)$.

Proof by contradiction:
If the greedy algorithm isn't optimal, the set produced by the greedy algorithm must be larger than the optimal set. That is, $|G|>|O|$. This means that there is at least one stop in the greedy solution that is not in the optimal solution. However, as proven by induction, the greedy algorithm is ahead of or even with the optimal solution at any stop, including the last stop, which is "last" because the distance from the stop to the destination is at most k meters. Therefore there is no way for the greedy algorithm to add an another term that the optimal solution doesn't have. So, the greedy algorithm must be optimal. That is, $|G| = |O|$.

Name: Alex Book
ID: 108073300
Collaboration: Michael Hasenkamp, Irvin Carbajal, Jamie Foster, Varun Narayanswamy

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.



Solution.
Apologies in advance about the answer boxes shifting from here until the end of the document, there simply wasn't enough room provided to fit my solution for this question on same page.

Solution at start: $v_0 = 0, v_1 = \infty, v_2 = \infty, v_3 = \infty, v_4 = \infty, v_5 = \infty$.

Visit v_0 because it is the starting vertex (also the lowest value unvisited vertex).
$v_1 = v_0 + E_{v_0,v_1} = 0 + 2 = 2$
$v_2 = v_0 + E_{v_0,v_2} = 0 + 7 = 7$
Solution so far: $v_0 = 0, v_1 = 2, v_2 = 7, v_3 = \infty, v_4 = \infty, v_5 = \infty$. Visited: v_0

Visit v_1 because it is the lowest value unvisited vertex.
$v_2 = v_1 + E_{v_1,v_2} = 2 + 4 = 6$
$v_3 = v_1 + E_{v_1,v_3} = 2 + 3 = 5$
$v_4 = v_1 + E_{v_1,v_4} = 2 + 1 = 3$
Solution so far: $v_0 = 0, v_1 = 2, v_2 = 6, v_3 = 5, v_4 = 3, v_5 = \infty$; Visited: v_0, v_1

Visit v_4 because it is the lowest value unvisited vertex.
$v_3 = v_4 + E_{v_4,v_3} = 3 + 1 = 4$
$v_5 = v_4 + E_{v_4,v_5} = 3 + 19 = 22$
Solution so far: $v_0 = 0, v_1 = 2, v_2 = 6, v_3 = 4, v_4 = 3, v_5 = 22$; Visited: v_0, v_1, v_4

Visit v_3 because it is the lowest value unvisited vertex.
$v_5 = v_3 + E_{v_3,v_5} = 4 + 13 = 17$
Solution so far: $v_0 = 0, v_1 = 2, v_2 = 6, v_3 = 4, v_4 = 3, v_5 = 17$; Visited: v_0, v_1, v_3, v_4

Visit v_2 because it is the lowest value unvisited vertex.
$v_4 = v_2 + E_{v_2,v_4} = 6 + 6 = 12$
$v_5 = v_2 + E_{v_2,v_5} = 6 + 3 = 9$
Solution so far: $v_0 = 0, v_1 = 2, v_2 = 6, v_3 = 4, v_4 = 3, v_5 = 9$; Visited: v_0, v_1, v_2, v_3, v_4

Visit v_5 because it is the lowest value unvisited vertex.
There are no vertices to travel to from v_5.
Solution so far: $v_0 = 0, v_1 = 2, v_2 = 6, v_3 = 4, v_4 = 3, v_5 = 9$; Visited: $v_0, v_1, v_2, v_3, v_4, v_5$

The whole graph has been visited, so the final solution is:
$v_0 = 0, v_1 = 2, v_2 = 6, v_3 = 4, v_4 = 3, v_5 = 9$.

Name: Alex Book
ID: 108073300
Collaboration: Michael Hasenkamp, Irvin Carbajal, Jamie Foster, Varun Narayanswamy

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.

```
def choosePlayers(n)
    winValues = [1,2,4,8,16]
    picks = []
    for i in range(len(winValues)-1,-1,-1):
        if winValues[i] <= n:
            picks.append(winValues[i])
            n-=winValues[i]
    print(picks)
```

   Proof by contradiction:
   If you have two players of type 2^i for $i < 4$ (since $16 = 2^4$ is the maximum value in the the winWalues array) you can combine those two players into one player of type 2^{i+1}. This mean that an optimal solution would have no duplicates of any player type except 16. So, if the greedy algorithm isn't optimal, the solution it produces would have at least one duplicate of player type 1, 2, 4, or 8.
   However, the greedy algorithm chooses as many 16's as possible.
   It then chooses one 8 if there is a great enough 'leftover' n value (but not more than one, as that would've meant there was room for another 16).
   It then chooses one 4 if there is a great enough 'leftover' n value (but not more than one, as that would've meant there was room for another 8).
   solution continued on next page

Solution.
It then chooses one 2 if there is a great enough 'leftover' n value (but not more than one, as that would've meant there was room for another 4).
It then chooses one 1 if there is a great enough 'leftover' n value (but not more than one, as that would've meant there was room for another 2).

Once there is no 'leftover' n value (n=0), the greedy algorithm returns a set that contains no duplicates other than players of type 16. Therefore, the greedy 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.
Let n=12.
Let winValues = [1,6,9].

Greedy algorithm chooses: [9,1,1,1].
Optimal solution is: [6,6].

The greedy algorithm fails for such win values and goal wins.

Name: Alex Book
ID: 108073300
Collaboration: Michael Hasenkamp, Irvin Carbajal, Jamie Foster, Varun Narayanswamy

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.