CSC510 Analysis of Algorithms
Section 04

Homework 4–
Dynamic Programming and Greedy

[redacted]

Full Name: [redacted]
Student ID: [redacted]

Midterm Instructions. All Students Must Read This!

Note: Failure to follow the following instructions in detail will impact your grade negatively.

1. All the assignments MUST have the student full name, and student id. If this information is not there, we won't grade your assignment, and it will be considered not submitted.
2. This homework is worth 15%. This homework has two problems. The second problem is divided into several parts.
3. Handwriting work is allowed as long as the work is clear and readable. If we can't understand your work, then we can't grade it.
4. Students must turn this assignment in on Canvas in PDF format, assignments submitted in a file format other than PDF will be considered not submitted and graded as such.
5. Blank assignments, or assignments missing work will be graded as such. No exceptions!. It is the responsibility of students to check if the assignment was submitted in the correct format.
6. Please read the late work policies stated in the syllabus. Homework submissions that do not follow these policies will get an unsatisfactory grade.

Problem Statement

You are in a new city C where all the streets in this city cross each other at different points. Your goal is to find out the length of the shortest path to walk from building A to building B (in miles). The city has the following layout:

C
- *Street S* = [*index*, *layout*, *is_one_way*, *length*]
  - Index: an integer from 0 to $(N-1)$ that represents the index of the street in the city's layout. Note that two different streets (with vertical and horizontal layout) may have the same street's index.
  - Layout: an integer value represented by (0 or 1) defining the layout of the street in relation to other streets in the city.
    - Value 0: vertical street
    - Value 1: horizontal street
  - Is_one_way: an integer value represented by (0 or 1) that defines whether the street is one-way or not
    - Value 0: False is Intersection
    - Value 1: True is one way
  - Length: an decimal value that represents the length of the street in miles from the starting point to the end point of the street.
- *Building X* = [*street*, *position*, *location*]
  - Street: a street S representing the street where the building is located
  - Position: an interger value represented by (0 or 1):
    - Value 0: right side of the street
    - Value 1: left side of the street
  - Location: An interger value represented by (0, 1, 2):
    - Value 0: the building is located at the start of the street
    - Value 1: the building is located at the middle of the street
    - Value 2: the building is located at the end of the street

- *Person P* $= [buildingA,\ buildingB,\ intersection,\ direction,\ distance\_traveled]$
  - BuildingA: Building A is the source building
  - BuildingB: Building B is the destination building
  - Intersection: the point where two streets intersect represented by a tuple with two streets (S_v, S_h) where S_v is a vertical street, and S_h is a horizontal street
  - Direction: a char value defining the direction this person is taking at any point in the process.
    - 'r': right
    - 'l': left
    - 'd': down
    - 'u': up
  - Distance_traveled: A decimal value representing the distance traveled so far by this person.
- City Policies:
  - Streets in City C do not have sidewalks. They only have a median strip where people can walk down as pedestrians
  - If a specific street is one-way or/and a dead-end, then this rule applies for pedestrians too.
  - Once a person exits from a building, this person must be directed towards the median strip that is located in front of the building.
  - Once a person has reached the destination, and in order to enter the destination building, this person must leave the median strip, cross the street and enter the building that will be located in front of the median strip.

The following is an example of an instance of this problem:

Given a city C with N streets, buildings B_a *and* B_b, and person P_1 as follows:

index ↓ | + d | One Way | + —One Way Index—+ +

- $C = [[\underline{0}, 0, \overline{0}, \underline{10.00}], [1, 0, 1, \underline{8.90}], [2, 0, 0, \underline{5.00}], [\underline{0}, 1, 1, \underline{10.00}], [1, 1, 0, \underline{15.50}], [2, 1, 0, \underline{12.50}]]$

a 0 B 1 C 2 D 3 E 4 F 5

- $B_a = [C[0], 0, 0];\ B_b = [C[5], 1, 2]$
- $P_1 = [B_a, B_b, (C[0], C[3]),$ ‘l’ $, 0.00]$

Find the minimum distance P_1 needs to travel from B_a to B_b in city C

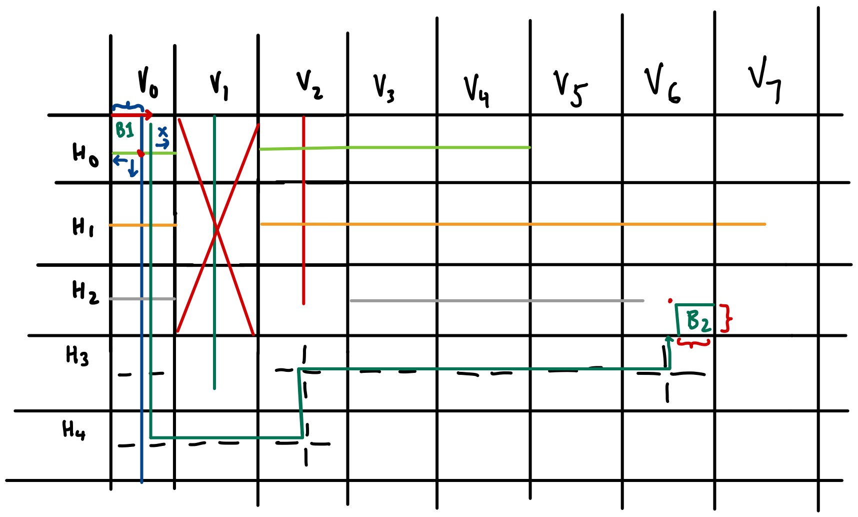
Your work starts here. Good Luck!

1. (5 points) Design an **optimized** algorithm for this problem using a Dynamic Programming or/and Greedy approach. You can use the example instance of this problem given in the problem statement, but you will need to prove that your algorithm works for any instance of this problem (not only for the given example). The design of your algorithm must include diagrams and short written step-by-step instructions explaining (at the high level) how your algorithm works. **Show ALL your work to get credit. Algorithm designs using approaches other than Dynamic Programming or/and Greedy won't get credit for this homework assignment**

C_0	a
C_1	b
C_2	c
C_3	p
C_4	e
C_5	f

B_1, B_2

following the example instance plus a few other streets



1) Building 1 (Ba) is Located at the right side, start of C[0]

2) Building 2 (B_b) is Located at the end of the street, Left of C[5]

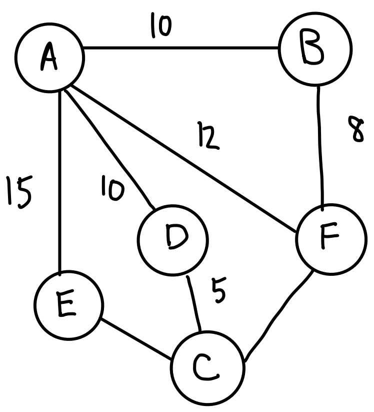
3) move left "1" towards (C[0], C[3])

4) cross intersection and go 'd' (down) because other side is dead end

5) at (C[0], C[2]) go '1' (Left) At this point the person should be at H4 middle intersection

6) then go 'u' (up) 2 miles and go 'r' (right) 8 miles

7) turn 'L' (Left) and building 2 (B_b) should be in front of the person P



a tile = 2 miles

middle Intersection = 1 mile

10 + 5 = 15 miles

2. (5 points) Create the pseudocode to solve all the instances of this problem based on your algorithm from part (a). Your pseudocode must use a Dynamic Programming or/and Greedy approach. **Students writting code instead of pseudocode won't get credit for this problem.**

```
Street (Index , Layout, Is_one_Way , Length)
        For each  node Street  in Index {
            If Index [value] = 0  and  Is_one_way [ value ] = False {
                Layout == vertical
                Street(Is_one_Way == Intersection)
              else {
              Street ( Layout == horizontal )
              return true

Person ( building A , Building B, Intersection , direction, distance)
    Initialize  Building 1 : Source
    Initialize  Building 2 : destination
    Initialize  direction : up, down , Left , right


  Building (street, position , Location)
       street (Index , position, Location)
       return

  Greedy Algorithmn :

       A = [V][V]

          for (k=1; k <= n; k++ ) {
              for (i = 1; i<=n; i++) {
                  for(j=1 ; j<=n; j++) {
                      A[i,J] = min(A[i][j], A[i,k] + A[k,J])
                      }
                 }
           }
```

3. (5 points) Find the complexity and time complexity of your algorithm. Note that you need to use a Step Counting approach to compute complexity and time complexity for this problem. **Show all your work, including summations to get credit.**

$$t(n) = \sum_{k=1}^{n} \sum_{i=1}^{n} \sum_{j=1}^{n} 1 = \sum_{i=1}^{n} i$$

$$\frac{n(n^2+n+1)}{2} = \frac{n^3+n}{2}$$

$$= O(n^3)$$

4. (2 points extra credit and optional) Create a program, using your favorite programming language, that implements your algorithm. To get credit for this problem students must provide some unit tests to test that your program performs as per your algorithm specifications. Create this program in an online editor and share here the link to your program. Broken links or code that doesn't work as per your algorithm specifications won't get credit.