Introduction to Programming

Shortest Path Problem – Code Outline

Refer to the assignment for the Shortest Path problem for a complete statement of the problem. The following code uses two 2-dimensional arrays: “f” which has the original data and “s” in which we construct, column by column the table with the shortest paths. Code is in yellow.

Here is f:

\[ f = [[3, 4, 1, 2, 8, 6], [6, 1, 8, 2, 7, 4], [5, 9, 3, 9, 9, 5], [8, 4, 1, 3, 2, 6], [3, 7, 2, 8, 6, 4]] \]

Here is s:

\[ s = [6*[0] \text{ for } i \text{ in range}(5)] \]

As discussed in class, we initialize the leftmost column of s with the leftmost column of f, since the length (value) of shortest path through each “cell” of the leftmost column, is just the value in the cell.

**#initialize column 0 of s - the shortest weight array**

\[
\text{for } i \text{ in range}(5):
\text{ s}[i][0]=f[i][0]
\]

We now define a function “short”. We pass short three parameters: the array s, and the “coordinates” of the row and column (i,j) for the “cell” through which we are currently calculating the shortest path.

The way short(s,i,j) works is that it will look up the values, in s, (1) immediately to the left, (2) up and to the left and (3) down and to the left of cell s[i][j], and return the smallest of the three values. The one thing to be careful of is to consider what is up from row zero and down from row 4 (consider using the % operator).

\[
\text{def short(s,i,j):}
\]

\[
\text{min}=\text{the value to the left of s}[i][j]
\]

\[
\text{# now look up and to the left}
\text{if min }> __________:\n\text{ min}=\text{__________}
\]

\[
\text{# now look down and to the left}
\text{if min }> __________:\n\text{ min}=\text{__________}
\]

\[
\text{return min}
\]

**# calculate the shortest weight array, s**

\[
\text{for } j \text{ in range}(1,6): \text{ # columns}
\text{ for } i \text{ in range}(5): \text{ # rows}
\text{ s}[i][j]=f[i][j]+\text{short}(s,i,j)
\]

**# if you like, you can check your work by printing s**

\[
\text{for } i \text{ in range}(5):
\text{ print(s[i])}
\text{ print()}
\text{ print()}
\]

We now get ready to find the value of the shortest path. It is the smallest value in column 5 of s.

**# find the exit point on the right by finding the minimum value in column 5**

\[
\text{min}=s[0][5]
\text{ row}=0
\]
for i in range(1,5):
    if _________ < min:
        min=______,
        row=i

Print out the length of the shortest path.

print()
print("The length of the shortest path is: ", min)
print()

We now find the actual path. We will create it by moving column by column through s from right to left. We will print it out print it as if we are going from left to right across the “field.” We append to path the row in each successive column that constitutes the shortest path that was generated.

path=[]
path.append(row)

for i in range(4,-1,-1): # going to the left from column 4 -> 0
    The code here is similar to the one in function short. We are looking for the smallest of three values to the left, and keeping track of the row in which it was found. When we have it, we append it to the list “path” as follows:
    path.append(row)

Finally, print the path.

print()
print("The path is: ")

p=[path[6-i] for i in range(1,7)]
print(p)

When you check your answer against the one given in the problem statement, keep in mind that they are numbering the rows in the table starting at 1 while we are starting at 0.