Problem 4 - Optimized Method for Finding the Shortest Path

Since enumerating all the paths is inefficient, let's optimize our search algorithm for the shortest path. As you discover new children nodes in your depth-first search, you can keep track of the shortest path that so far that minimizes the distance traveled and minimizes the distance outdoors to fit the constraints.

If you come across a path that is longer than your shortest path found so far, then you know that this longer path cannot be your solution, so there is no point in continuing to traverse its children and discover all paths that contain this sub-path.

Implement the function `directedDFS(digraph, start, end, maxTotalDist, maxDistOutdoor)` that uses this optimized method to find the shortest overall path in a directed graph from start node to end node under the following constraints: the total distance travelled is less than or equal to `maxTotalDist`, and the total distance spent outdoors is less than or equal to `maxDistOutdoor`. If multiple paths are still found, then return any one of them. If no path can be found to satisfy these constraints, then raise a `ValueError` exception.

As with the previous problem, we suggest using one or more helper functions to implement `directedDFS` (see hint from above). In particular, is there any additional information you can pass or variable you can update that can help you reduce the number of nodes traversed?

Test your code by uncommenting the code at the bottom of ps5.py.

```
def directedDFS(digraph, start, end, maxTotalDist, maxDistOutdoors):
    """
    Finds the shortest path from start to end using directed depth-first
    search approach. The total distance traveled on the path must not
    exceed maxTotalDist, and the distance spent outdoor on this path
    must not exceed maxDistOutdoors.

    Parameters:
        digraph: instance of class Digraph or its subclass
        start, end: start & end building numbers (strings)
        maxTotalDist : maximum total distance on a path (integer)
        maxDistOutdoors: maximum distance spent outdoors on a path (integer)

    Assumes:
        start and end are numbers for existing buildings in graph

    Returns:
        The shortest-path from start to end, represented by
        a list of building numbers (in strings), [n_1, n_2, ..., n_k],
        where there exists an edge from n_i to n_(i+1) in digraph,
        for all 1 <= i < k.

        If there exists no path that satisfies maxTotalDist and
        maxDistOutdoors constraints, then raises a ValueError.
    """
    # TO DO
```

Paste your code for both `WeightedEdge` and `WeightedDigraph` in this box. You may assume the grader has provided implementations for `Node`, `Edge`, and `Digraph`. Additionally paste your code for `directedDFS`, and any helper functions, in this box.

Test: map1

```
Testing map 1
```

Output:

```
Looking at map 1:
1->2 (10.0, 5.0)
2->3 (8.0, 5.0)
directedDFS(map1, "1", "3", 100, 100)
['1', '2', '3']
Test completed
```

Test: map2 A

```
Testing map 2
```

Output:

```
Looking at map 2:
1->2 (10.0, 5.0)
1->4 (5.0, 1.0)
2->3 (8.0, 5.0)
4->3 (8.0, 5.0)
directedDFS(map2, "1", "3", 100, 100)
['1', '4', '3']
Test completed
```

Test: map2 B

```
Testing map 2
```

Output:

```
directedDFS(map2, "1", "3", 18, 18)
['1', '4', '3']
directedDFS(map2, "1", "3", 15, 15)
['1', '4', '3']
directedDFS(map2, "1", "3", 18, 0)
ValueError successfully raised
directedDFS(map2, "1", "3", 10, 10)
ValueError successfully raised
Test completed
```

Test: map3 A

```
Testing map 3
```

Output:

```
Looking at map 3:
1->2 (10.0, 5.0)
1->4 (15.0, 1.0)
2->3 (8.0, 5.0)
4->3 (8.0, 5.0)
directedDFS(map3, "1", "3", 100, 100)
['1', '2', '3']
Test completed
```

Test: map3 B

```
Testing map 3
```

Output:

```
directedDFS(map3, "1", "3", 18, 18)
['1', '2', '3']
directedDFS(map3, "1", "3", 18, 0)
ValueError successfully raised
directedDFS(map3, "1", "3", 10, 10)
ValueError successfully raised
Test completed
```

Test: map4 A

```
Testing map 4
```

Output:

```
Looking at map 4:
1->2 (5.0, 2.0)
3->5 (6.0, 3.0)
2->3 (10.0, 5.0)
2->4 (20.0, 10.0)
4->3 (2.0, 1.0)
4->5 (20.0, 10.0)
directedDFS(map4, "1", "3", 100, 100)
['1', '2', '3']
directedDFS(map4, "1", "5", 100, 100)
['1', '2', '3', '5']
Test completed
```

Test: map4 B

```
Testing map 4
```

Output:

```
    directedDFS(map4, "1", "5", 21, 10)
    ['1', '2', '3', '5']
    directedDFS(map4, "1", "5", 21, 9)
    ValueError successfully raised
    directedDFS(map4, "1", "5", 20, 20)
    ValueError successfully raised
    Test completed
```

Test: map5 A

```
Testing map 5
```

Output:

```
    Looking at map 5:
    1->2 (5.0, 2.0)
    3->5 (6.0, 3.0)
    2->3 (20.0, 10.0)
    2->4 (10.0, 5.0)
    4->3 (2.0, 1.0)
    4->5 (20.0, 10.0)
    directedDFS(map5, "1", "3", 100, 100)
    ['1', '2', '4', '3']
    directedDFS(map5, "1", "5", 100, 100)
    ['1', '2', '4', '3', '5']
    Test completed
```

Test: map5 B

```
Testing map 5
```

Output:

```
    directedDFS(map5, "1", "3", 17, 8)
    ['1', '2', '4', '3']
    directedDFS(map5, "1", "5", 23, 11)
    ['1', '2', '4', '3', '5']
    directedDFS(map5, "4", "5", 21, 11)
    ['4', '3', '5']
    directedDFS(map5, "5", "1", 100, 100)
    ValueError successfully raised
    directedDFS(map5, "4", "5", 8, 2)
    ValueError successfully raised
    Test completed
```

Test: map6 A

```
Testing map 6
```

Output:

```
Looking at map 6:
1->2 (5.0, 2.0)
3->5 (5.0, 1.0)
2->3 (20.0, 10.0)
2->4 (10.0, 5.0)
4->3 (5.0, 1.0)
4->5 (20.0, 1.0)
directedDFS(map6, "1", "3", 100, 100)
['1', '2', '4', '3']
directedDFS(map6, "1", "5", 100, 100)
['1', '2', '4', '3', '5']
Test completed
```

Test: map6 B

```
Testing map 6
```

Output:

```
directedDFS(map6, "1", "5", 35, 9)
['1', '2', '4', '3', '5']
directedDFS(map6, "1", "5", 35, 8)
['1', '2', '4', '5']
directedDFS(map6, "4", "5", 21, 11)
['4', '3', '5']
directedDFS(map6, "4", "5", 21, 1)
['4', '5']
directedDFS(map6, "4", "5", 19, 1)
ValueError successfully raised
directedDFS(map6, "3", "2", 100, 100)
ValueError successfully raised
directedDFS(map6, "4", "5", 8, 2)
ValueError successfully raised
Test completed
```