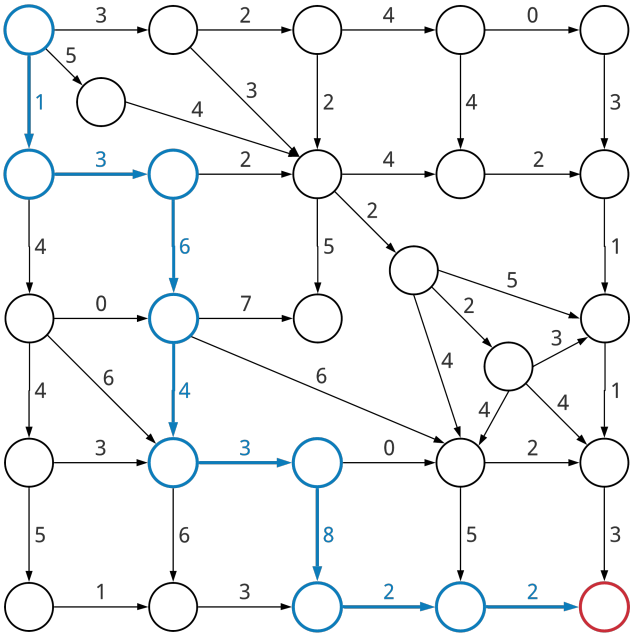
5D Find a Longest Path in a DAG

Longest Path in a DAG Problem
Find the longest path between two nodes in an edge-weighted DAG.

Input: An edge-weighted directed acyclic graph (DAG) and nodes *Source* and *Sink* in this DAG.
Output: A longest path from *Source* to *Sink* in this DAG and its length.



Formatting

Input: Two space-separated integers labeling the source and sink nodes of an edge-weighted DAG, followed by an edge list representing an edge-weighted DAG.
Output: The length of a longest path in the edge-weighted DAG as an integer followed by a space-separated list of integers representing a longest path in the edge-weighted DAG (if multiple longest paths exist, you may return any one).

Constraints

- The values of *Source* and *Sink* will be between 1 and 10^2.
- The number of nodes in the DAG will be between 1 and 10^2.
- The number of edges in the DAG will be between 1 and 10^2.
- All nodes in the DAG will be labeled with integers.
- All edges in the DAG will have integer edge weights.

Test Cases

Case 1

Description: The sample dataset is not actually run on your code.

Input:

```
0 4
0 1 7
0 2 4
1 4 1
2 3 2
3 4 3
```

Output:

```
9
0 2 3 4
```

Case 2

Description: This test makes sure that your code is actually finding the *longest* path in a DAG not the *shortest* path in a DAG. The *shortest* path in this DAG goes 0 1 2 3 with path length of 3 while the *longest* path goes 0 3 with a path length of 10. If your code outputs a path length of 3 it is likely that you are finding the *shortest* path instead of the *longest* path.

Input:

```
0 3
0 1 1
0 3 10
1 2 1
2 3 1
```

Output:

```
10
0 3
```

Case 3

Description: This test makes sure that your code correctly parses the input. An input line of the form x y z represents an edge from node x to node y with a weight of z. A common mistake is parsing the input so that x y z represents an edge from node x to node z with weight y. If your code outputs a path length of 4 with a path of 0 2 3 it is likely that you are making the described parsing error.

Input:

```
0 3
0 1 2
0 2 1
1 3 3
2 3 3
```

Output:

```
5
0 1 3
```

Case 4

Description: This test makes sure that your code isn't an implementation of a greedy approach to this problem. A simple greedy approach that chooses paths at branching points based on edge weights at that point is not guaranteed return the correct output for this problem. If your code outputs a path weight of 6 and a path of 0 2 3 then it's possible that you are using a greedy approach and fall into a non-optimal solution immediately by choosing the 0 2 5 edge.

Input:

```
0 3
0 1 1
0 2 5
1 3 10
2 3 1
```

Output:

```
11
0 1 3
```

Case 5

Description: This test makes sure that your code does not rely on the source node being labeled 0. This dataset is the same as test dataset 4, except that each node label is incremented by one. If your output doesn't match the correct output then your code likely assumes that the source node is labeled 0. Make sure that your implementation uses the source node label from the input instead of making any assumptions about the label of the source node.

Input:

```
1 4
1 2 1
1 3 5
2 4 10
3 4 1
```

Output:

```
11
1 2 4
```

Case 6

Description: This test makes sure that your code can correctly parse inputs in which there are double digit node labels. All previous datasets only have nodes with single digit labels, so if your code relies on node labels only having single digits it will likely fail on this dataset. If your output doesn't match the correct output make sure that your code can handle nodes that have double digit labels.

Input:

```
1 10
1 2 1
2 3 3
3 10 1
```

Output:

```
5
1 2 3 10
```

Case 7

Description: This test makes sure that your code can correctly handle inputs that only contain one edge. If your output doesn't match the correct output make sure that your implementation doesn't contain an off-by-one error that prevents the use of the only edge in the graph.

Input:

```
0 4
0 4 7
```

Output:

```
7
0 4
```

Case 8

Description: A larger dataset of the same size as that provided by the randomized autograder. Check input/output folders for this dataset.