Longest Path Problem

Thomas Rothman, Nick Buchan, Parth Parikh, Seth Walter

The Problem

Decision problem:

- Does the directed graph G contain a path of at least k edges?

Optimization Version:

- What is the longest path in the directed graph G starting from vertex s?

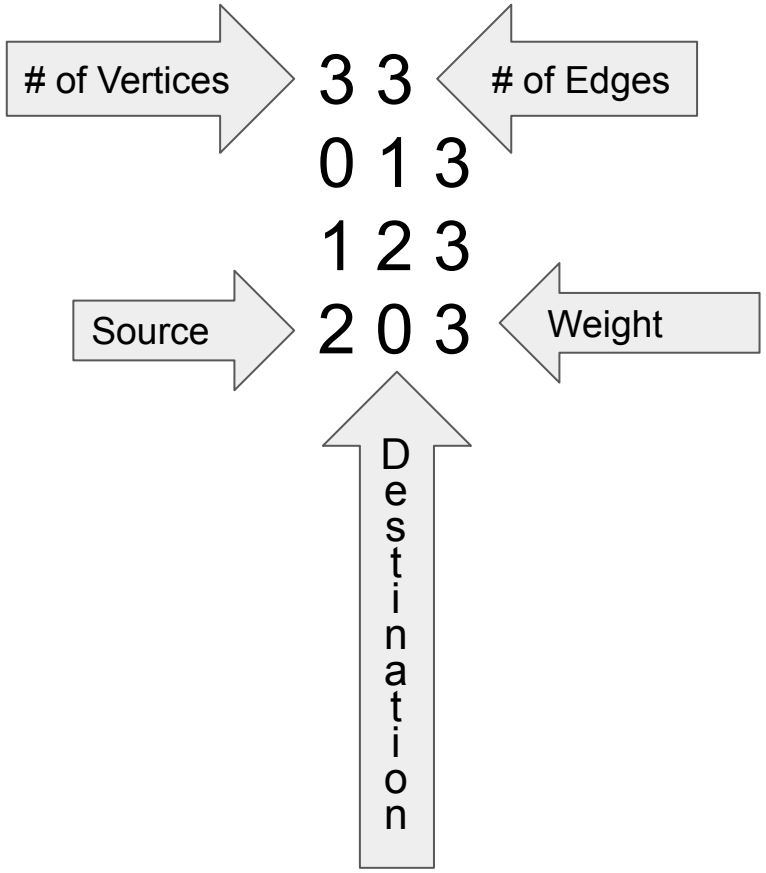
Applications

- A main use of the longest path problem is finding the Critical Path
  - The critical path is determined by finding the longest stretch of dependent events and calculating how long it takes to complete them.
  - Finding the critical path is important for projects and planning, such as building a house.
- The Longest Path Problem can also be used to find Hamiltonian Paths.

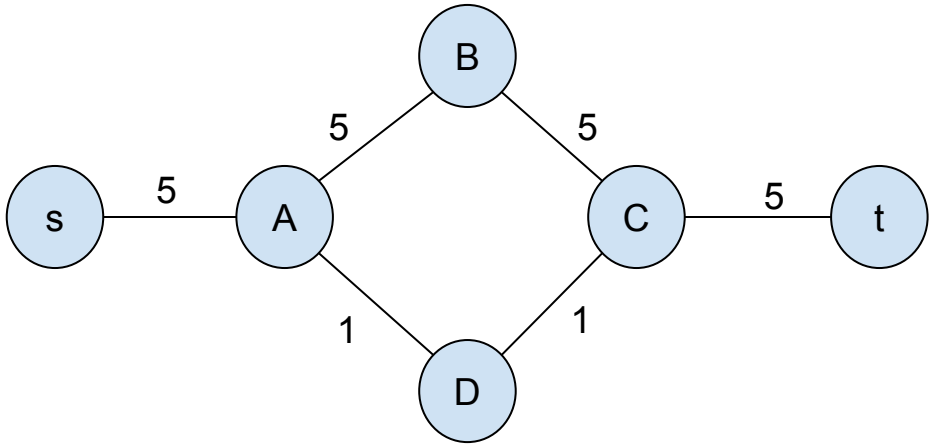
Problem Input

Input: A direct, weighted graph.

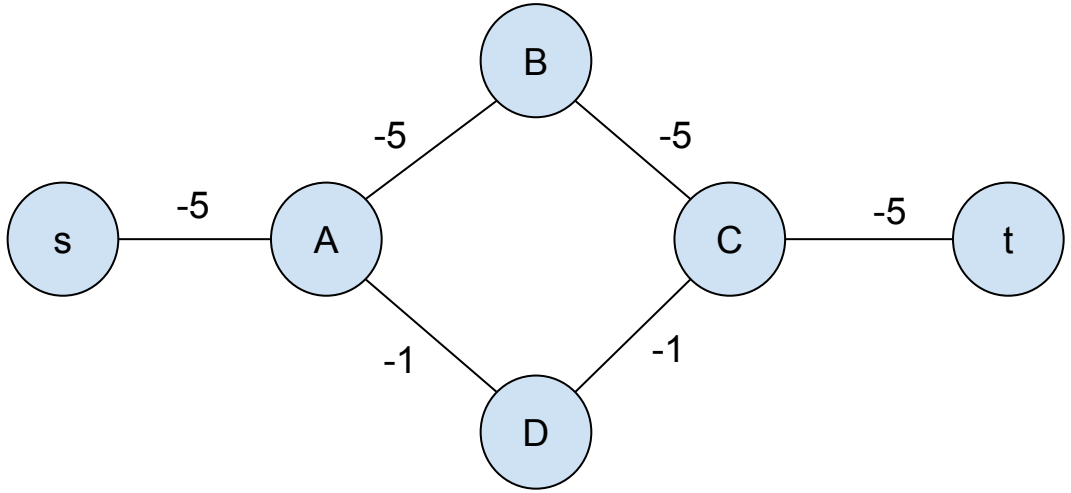
The first line of input is in the form *n m.* The number of vertices in the input graph is *n* and the number of edges is *m.*
The next *m* lines are specified in the form *a b c. a* is the source vertex, *b* is the destination, and *c* is the edge weight.



Negating edge weights and finding the shortest path?

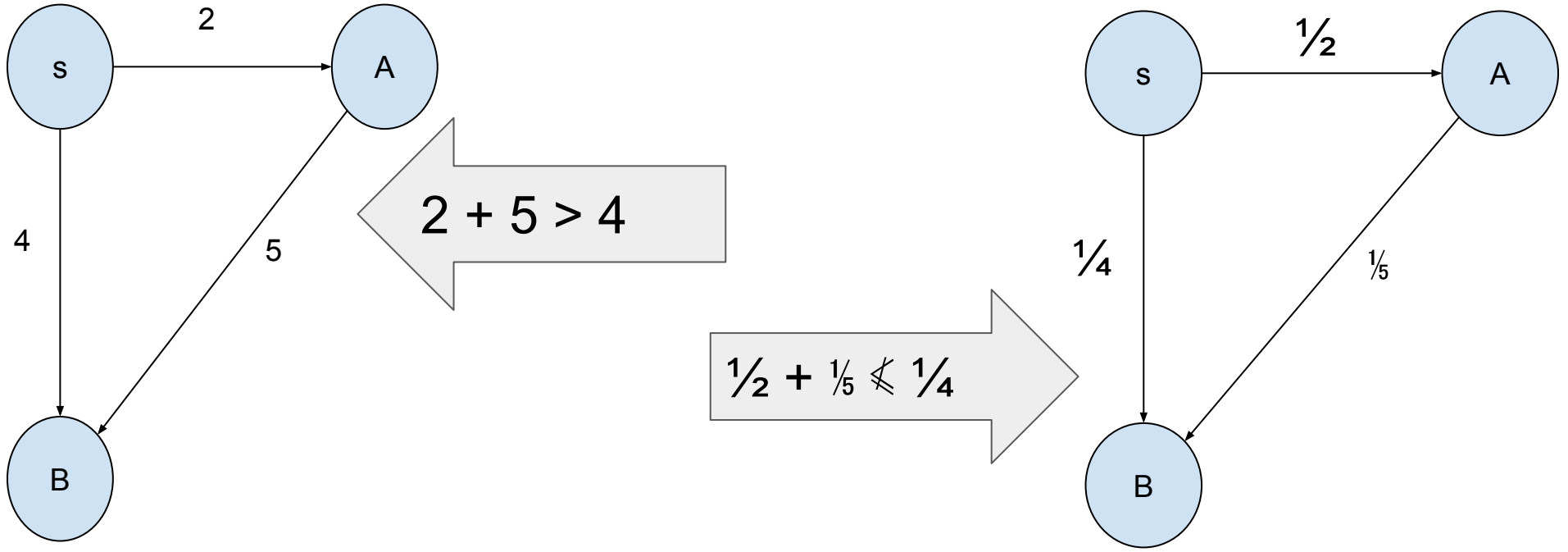


Because the graph contains a cycle, running Bellman-Ford on the negated graph causes it to keep going around the cycle creating a longer path each time.



However, the longest path problem is solvable in linear time if the it is a Direct Acyclic Graph (DAG).

Calculating the inverse weights and finding shortest path?

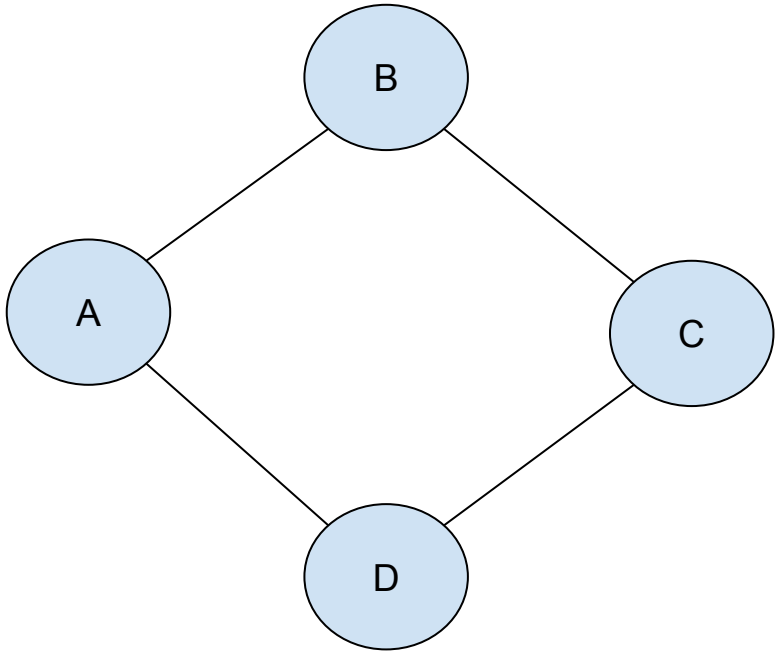


Using the reciprocal of the weights does not work because the inequality does not hold true.

Reduction

Hamiltonian Path $\leq_P$ Longest Path

A graph G has a Hamiltonian path if and only if its longest path has length n−1, where n is the number of vertices in G.



The Hamiltonian Path is ABCD which has length 3.
This is also the longest path because n-1 = 3

Sketch of Exact Solution (pseudo-code)

```
#Finds all the paths from s to t in the graph
def findAllPaths(graph, s, t):
        simples = []
        def pathFinder(graph, s, t, visited, path):
                visited[s] = True
                path.append(s)
                if s == t:
                        simples.append(path.copy())
                else:
                        for v in graph[s]:
                                if visited[v] == False:
                                        pathFinder(graph, v, t, visited, path)

#Loops through all the paths that were found in the graph and returns the longest ones
def findLongestPath (graph, paths):
         for path in paths:
                  if currentPathLength > longestPathLength:
                        longestPath = currentPath
                        longestPathLength = currentPathLength
```

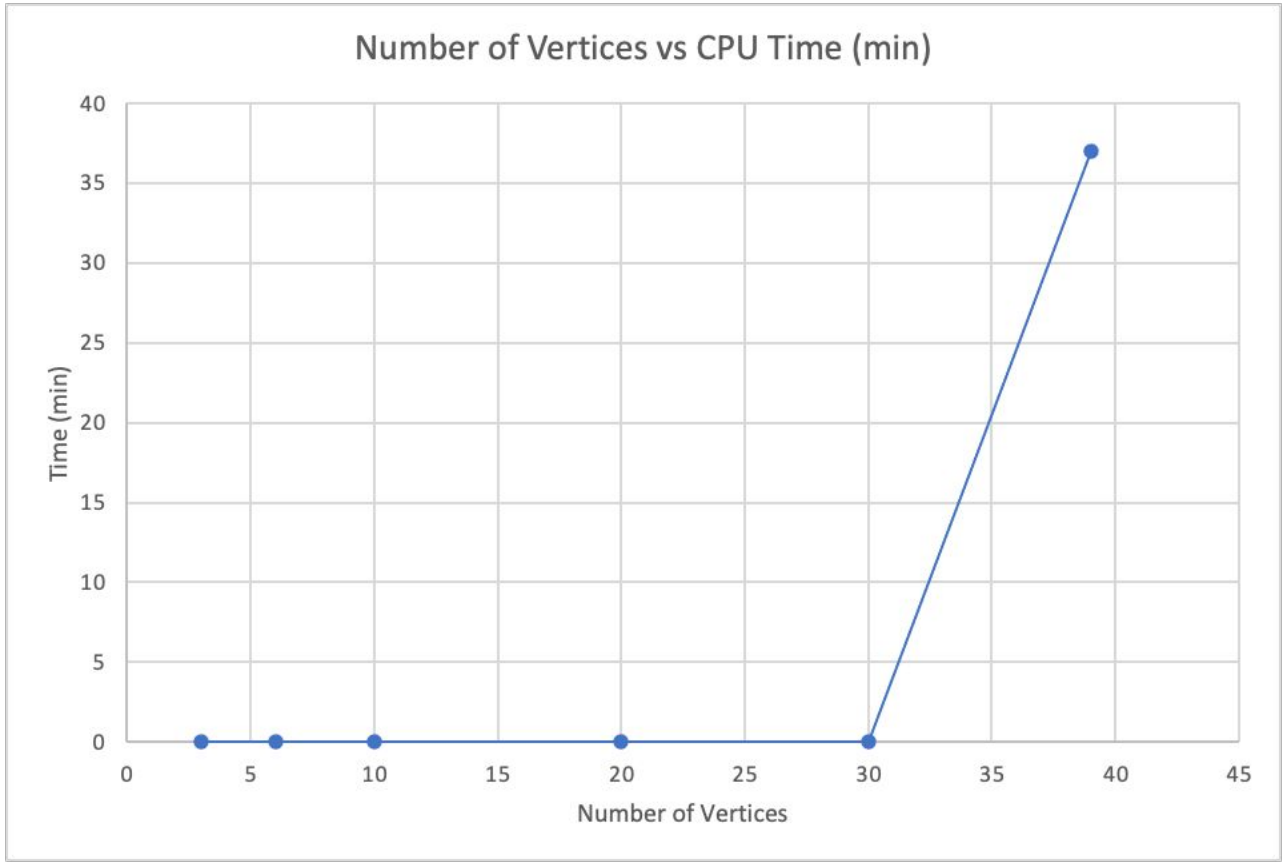
Test Cases

We the generated test cases with a Python program.

We generated test cases of different sizes, ranging from 3-50 vertices.

In a graph with 39 vertices, the program took 37.55 min to find the longest path

Vertices	Time (min)
3	0.00055
6	0.0006
10	0.00053333
20	0.0009
30	0.0415
39	37.0148



Runtime Analysis

The exact solution runs in $O(2^n)$ where n is the number of edges.

The algorithm uses brute force to find all possible paths in the graph, and then compares them to find the longest path.