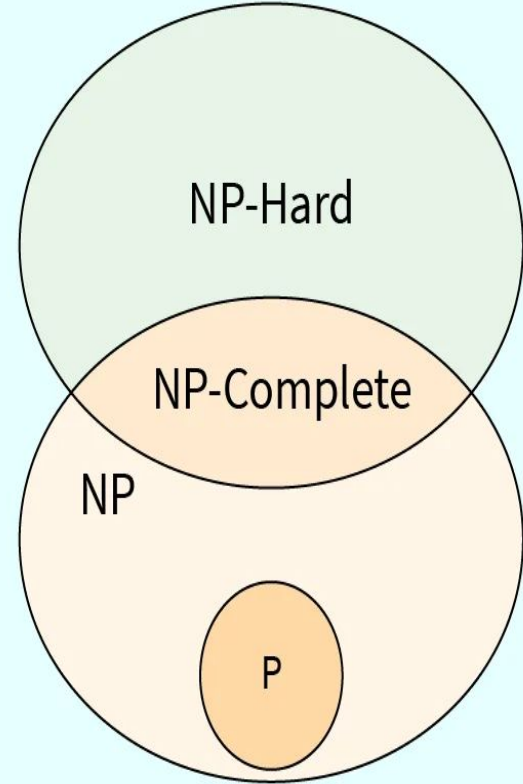# Longest Path NP Complete Analysis
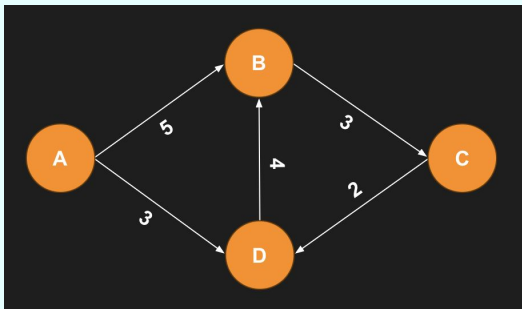
By:
Jackson Greer
Julian Hamze
Ian Gribb

Exact Solution

Goal: Find the simple path (no repeated vertices) between any two vertices with maximum total weight.
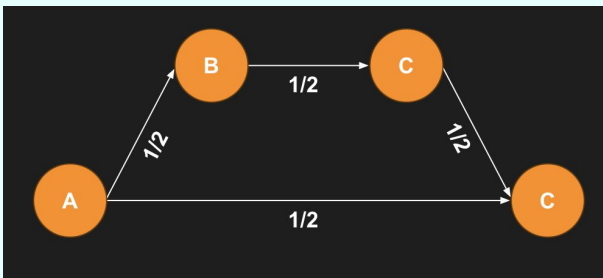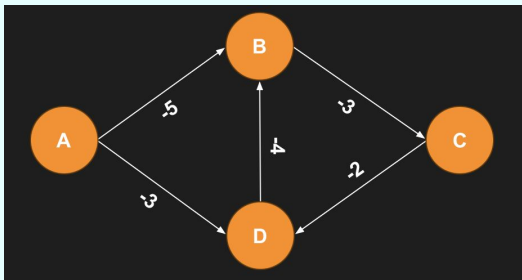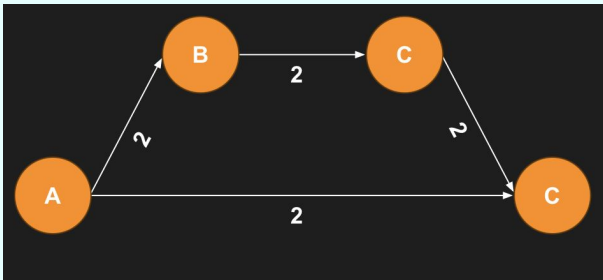
- Decision: YES/NO

- Optimization: What is the longest path

Intuitive Solutions That Don't Work

Inverted Edges

Reciprocal Edges

Certifier is Polynomial

Certifier(G, s, t, k, path):

Time Complexity:
O(n)

1. Check that the path starts at s and ends at t.
   If not, reject the certificate.

2. Check that no vertex appears more than once in the path.
   (A simple path cannot repeat vertices.)
   If any vertex repeats, reject the certificate.

3. For each pair of consecutive vertices in the path:
   Check that an edge actually exists between them in the graph.
   If any required edge is missing, reject the certificate.

4. Count how many edges the path contains.
   If the length of this path is at least k, accept the certificate.
   Otherwise, reject it.

Big-O Runtime Analysis

```
def find_longest_path(vertices, graph):
    ...

    for start in vertices: # multiplies total cost by n
        ...

        def dfs(current_vertex, path_length, path):
            ...

            # EXPONENTIAL BRANCHING
            for neighbor, weight in graph[current_vertex].items():
                if neighbor not in visited:
                    …

                    dfs(neighbor, …) # EXPONENTIAL RECURSION

                    …

        dfs(start, 0, [start])
        ...
```

Time Complexity:
O(n! * n)

Exact Solution Runtime
(Part D Additional Test Cases)
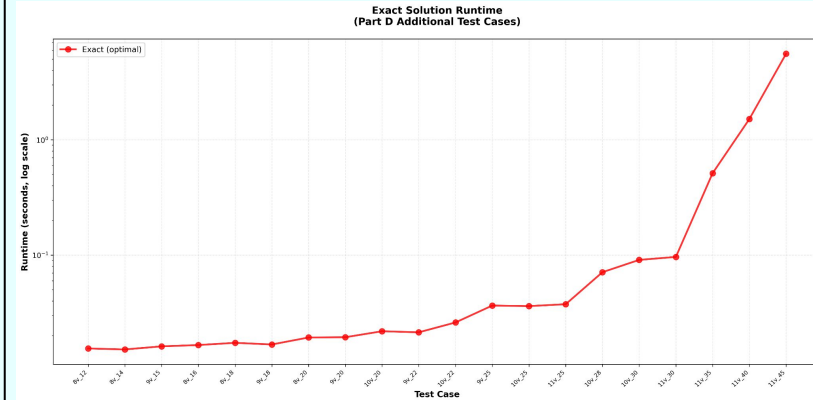
Hamiltonian Path => Longest Path

1. Copy all vertices in G (O(n))
2. Copy every edge with weight 1 (O(m))
3. Calculate k (O(n))

Hamiltonian Path	Polynomial Transformation →	Longest Path	Polynomial Transformation →	YES/NO