A close-up of a text

Description automatically generated

Dijkstra’s algorithm is not designed to work correctly with graphs that contain negative edge weights. However, there is an edge case where Dijkstra’s algorithm can still function correctly:

* If there are negative edge weights, but **no negative cycles**, Dijkstra's algorithm can work correctly if the negative edges only appear in situations where they don't affect the optimal path. I give an example about no negative cycle as follows:

A diagram of a diagram

Description automatically generated

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Iter** | **PQ** | **d(S)** | **d(A)** | **d(B)** | **d(C)** | **d(D)** |
| 1 | [S] | 0 | ∞ | ∞ | ∞ | ∞ |
| 2 | [A] | 0 | 4 | ∞ | ∞ | ∞ |
| 3 | [C,B] | 0 | 4 | 2 | 1 | ∞ |
| 4 | [B,D] | 0 | 4 | 2 | 1 | 5 |
| 5 | [D] | 0 | 4 | 2 | 1 | 4 |
| 6 | [] | 0 | 4 | 2 | 1 | 4 |

**Dijkstra’s algorithm** can still find the shortest path in this graph because the negative edges are not part of the **shortest path** from the source to the other vertices.

A close-up of a text

Description automatically generated

1.

1. **Initially, calculate G = (V ; E) to use Dijkstra's algorithm twice:**  
   First Run: Find the shortest path from city s to all other cities; Second Run: Find the shortest path from city ttt to all other cities.
2. **Evaluate Each Potential Road:**   
   for u, v, l in potential\_cities:  
    max\_dist = min(dist\_s[u] + l + dist\_t[v], dist\_t[v] + dist\_s[u] + l)
3. **Determine the Maximum Decrease in Distance:**  
   decrease = dist\_s[t] - max\_dist  
   if decrease > max\_deceased:  
    max\_deceased = decrease
4. **Output the Best Road:**  
   def find\_max\_decrease(self, s, t, potential\_cities):  
    dist\_s = self.dijkstra(s)  
    dist\_t = self.dijkstra(t)  
    max\_deceased= 0  
    best\_road = None  
    for u, v, l in potential\_cities:  
    max\_dist = min(dist\_s[u] + l + dist\_t[v], dist\_t[v] + dist\_s[u] + l)  
    decrease = dist\_s[t] - max\_dist  
    if decrease > max\_deceased:  
    max\_deceased = decrease  
    best\_road = (u, v, l)  
    return best\_road, max\_deceased

2.

1. **Dijkstra's Algorithm**: The time complexity of Dijkstra’s algorithm is O((V+E)logV). Running algorithm twice will multiply 2.
2. Comparing the new distances for each road takes O(∣E′∣).
3. Total time complexity is O((V+E)logV) + O(∣E′∣).

A white paper with black text

Description automatically generated

1. Firstly, Start with the largest denomination (10¢). Next, use the 5¢ denomination. Finally, use the 1¢ denomination.
2. The running time is **O(1).**

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **N** | **Order of Coins** | **Number of Coins** |
| 1 | 13 | 10 | 1 |
| 2 | 3 | 1 | 1 |
| 3 | 2 | 1 | 1 |
| 4 | 1 | 1 | 1 |



If the 5¢ coin is replaced with a 6¢ coin, **the greedy algorithm will still work correctly** because:

* The greedy approach always tries to use the largest denomination first.
* The problem with a 5¢ coin is that sometimes it would lead to suboptimal results.
* With a 6¢ coin, the algorithm will still select the largest possible denomination, and since 6 is a divisor of 12, it will correctly make change for numbers like 12 or 13 without suboptimal behavior.

A diagram of a network

Description automatically generated

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Iter** | **PQ** | **d(A)** | **d(B)** | **d(C)** | **d(D)** | **d(E)** |
| 1 | [A] | 0 | ∞ | ∞ | ∞ | ∞ |
| 2 | [C, B] | 0 | 3 | 1 | ∞ | ∞ |
| 3 | [B, D] | 0 | 3 | 1 | 3 | ∞ |
| 4 | [D, E] | 0 | 3 | 1 | 3 | 4 |
| 5 | [E] | 0 | 3 | 1 | 3 | 4 |
| 6 | [] | 0 | 3 | 1 | 3 | 4 |

A diagram of a graph

Description automatically generated

1.

|  |  |  |  |
| --- | --- | --- | --- |
| **Edge** | **Weight** | T/F | Weight |
| AE | 1 | T | 1 |
| EF | 1 | T | 1 |
| EB | 2 | T | 2 |
| BF | 2 | F |  |
| GH | 3 | T | 3 |
| FG | 3 | T | 3 |
| CG | 4 | T | 4 |
| BC | 5 | F |  |
| CF | 5 | F |  |
| DG | 5 | T | 5 |
| AB | 6 | F |  |
| DH | 7 | F |  |
| the cost of MST | | | 19 |

2.

A black line with a letter f

Description automatically generated A black lines with black letters

Description automatically generated

There are 2 MST in this graph.

3.

**Step1: Sort the list of edges**

|  |  |
| --- | --- |
| **Weight** | **Edge** |
| 1 | AE,EF |
| 2 | EB,BF |
| 3 | GH,FG |
| 4 | CG |
| 5 | BC,CF,DG |
| 6 | AB |
| 7 | DH |

**Step2: Calculate the MST**

|  |  |  |  |
| --- | --- | --- | --- |
| **Weight** | **Edge** | **Find\_Vertex** | **Total\_Dist** |
| 1 | **AE,EF** | A,E,F | 2 |
| 2 | **EB**,BF | A,E,F,B | 4 |
| 3 | **GH,FG** | A,E,F,B,G,H | 10 |
| 4 | **CG** | A,E,F,B,G,H,C | 14 |
| 5 | BC,CF,**DG** | A,E,F,B,G,H,C,D | 19 |
| 6 | AB | - | 19 |
| 7 | DH | - | 19 |