DIJKSTRA ALGORITHM

Overview

- Optimization problem, specifically the shortest single path from one vertex to another vertex.
- For Dijkstra to work correctly, all weights on the edges must be non negative
- If there is at least one edge with negative weight, Dijkstra might not work.
- Dijkstra works both on directed on undirected and directed graph. In the undirected graph case, we treat each edge as 2 edges on the opposite directions with the same weights

The algorithm:

- Initialization: set all vertices to ∞ except the source to 0.
- Traverse the graph:
  - Greedy selection:

    Select unvisited vertex with the shortest distance. If this vertex is not the target and has distance of ∞, it means the remaining unvisited vertices are not reachable, hence stops the algorithm.
  - Relaxation:

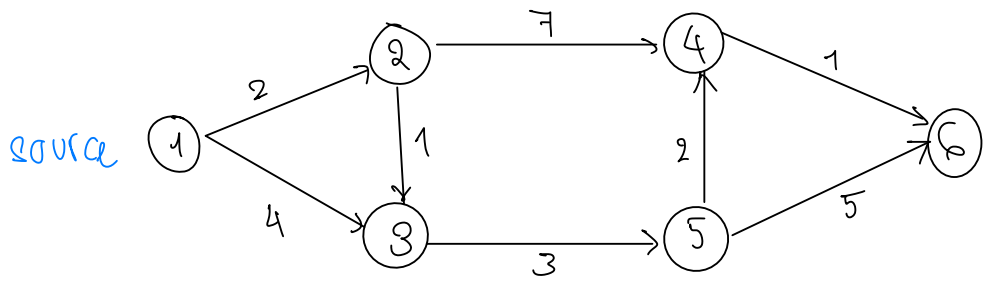
    After vertex is selected, check all of its neighbors that are reachable, and perform:

    if vertex + dist(vertex, neighbor) < neighbor:

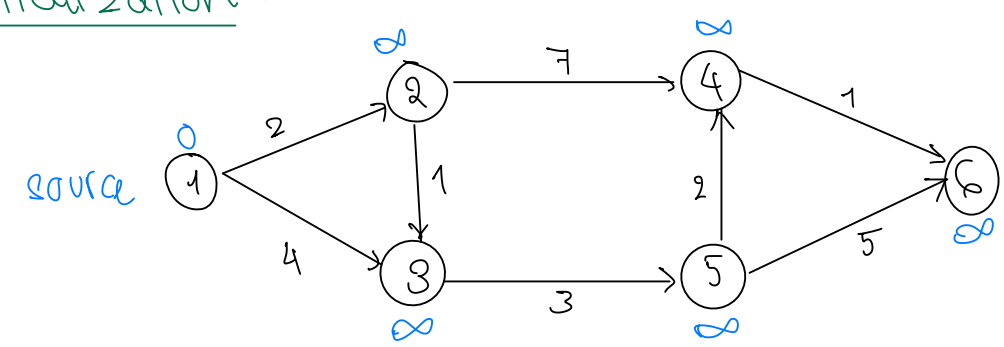
    neighbor = vertex + dist(vertex, neighbor)
- Termination:

  As mentioned in Greedy step, once found a vertex with infinity distance, we can terminate

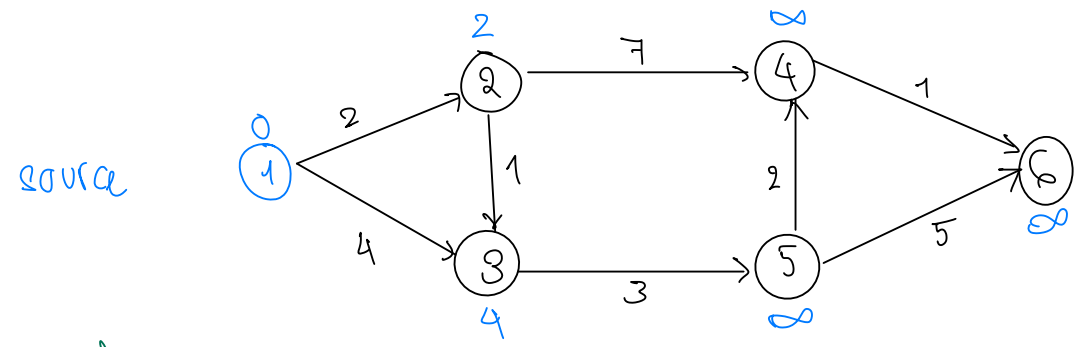
Example 1: Visual, target reached



Initialization:



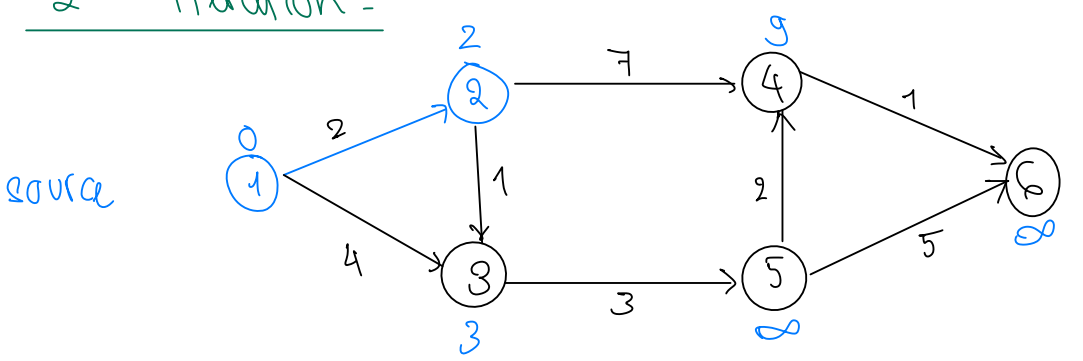
1st iteration:



Greedy: pick ①

Relaxation: update ②, ④

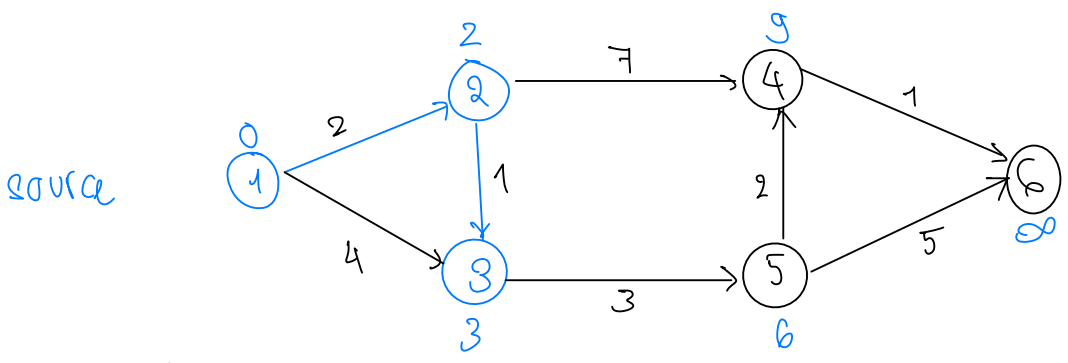
2nd iteration:



Greedy: pick ②

Relaxation: update ③, ④

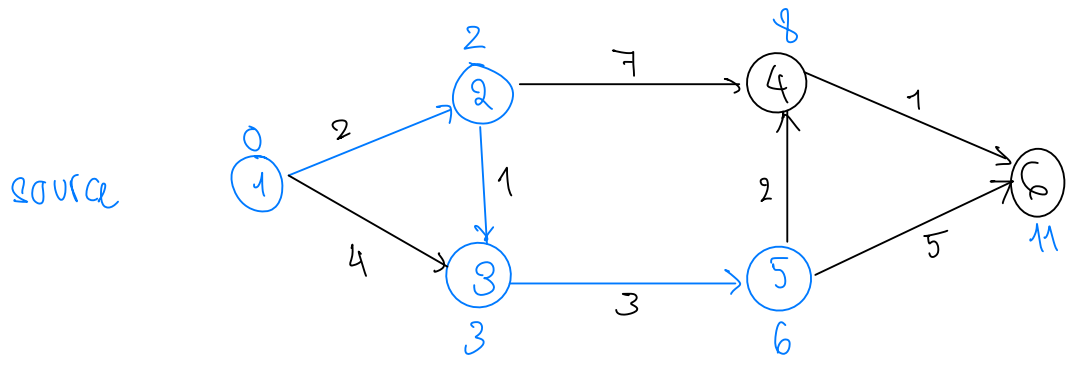
3rd iteration:



Greedy: pick ③

Relaxation: update ⑤

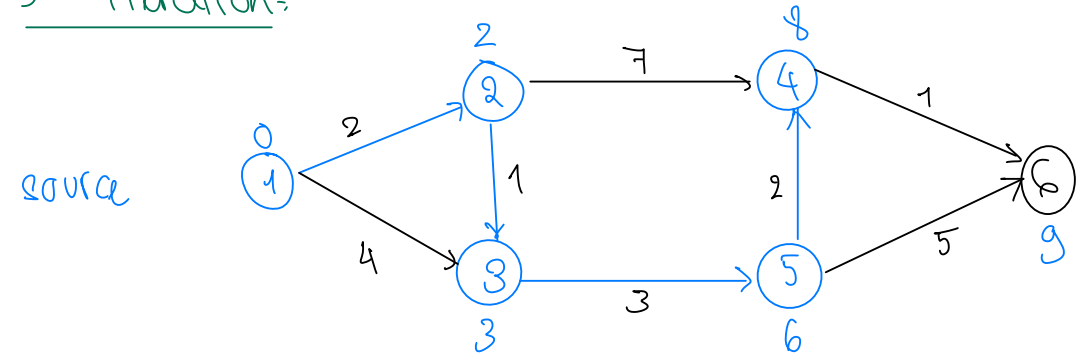
4th iteration:



Greedy: pick ⑤

Relaxation: update ④, ⑥

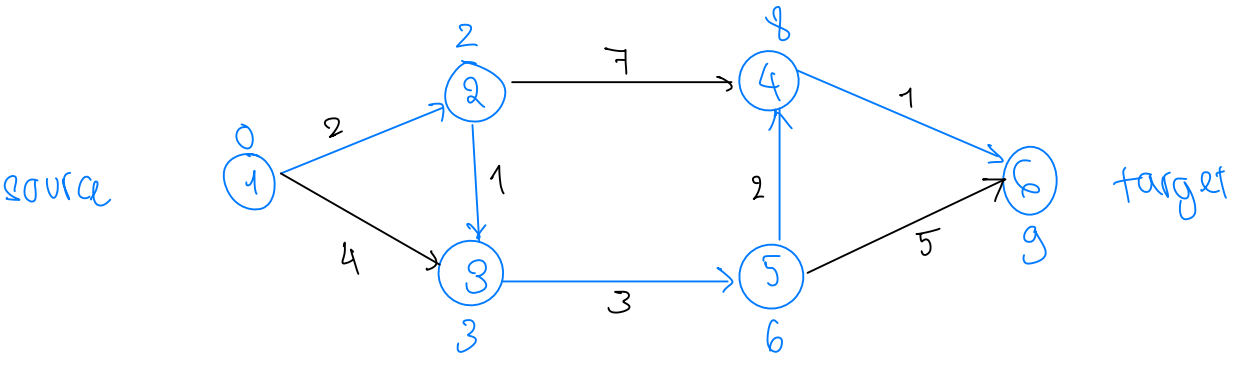
5th iteration:



Greedy: pick ④

Relaxation: update ⑥

6th iteration:



Greedy: pick ⑥

Terminate: reached target

return:

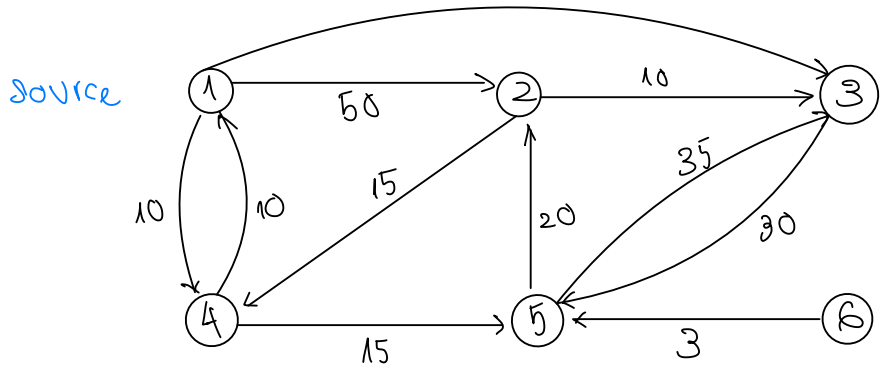
$$v = [①, ②, ③, ⑤, ④, ⑥]$$
$$d[v] = [0, 2, 3, 6, 8, 9]$$

Time Complexity:

- We can greedy pick at most $|V|$ vertices
- Every time we pick a vertex, we can update at most $|V|$ vertices

Therefore, the overall time complexity is $O(|V| \cdot |V|) \sim O(n^2)$

Example 2: Table, target not reachable



Selected vertex	②	③	④	⑤	⑥	
④	50	45	10	∞	∞	
⑤	50	45	10	25	∞	
②	45	45	10	25	∞	
③	45	45	10	25		→ no change

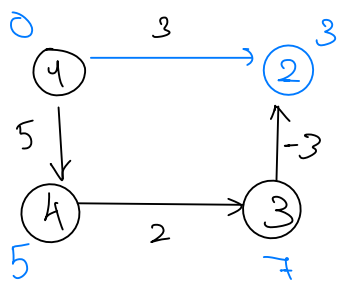
termination since ③ → ⑤ are both visited

⇒ ⑥ cant be reached

Dijkstra limitation:

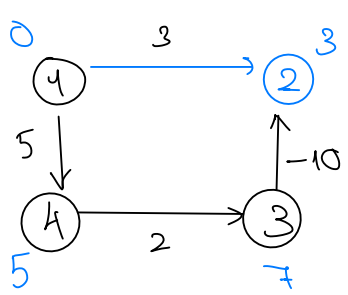
When we introduce negative weight / distance, the algorithm might break, here we present 2 scenarios of success and failure.

Success example:



In this case, the path picked by Dijkstra is still correct as the alternative path: ① → ④ → ③ → ② is clearly longer

Failure example



Now, we changed the negative weight to -10, can clearly see that the path picked by Dijkstra is no longer the correct answer