**Algorithms for shortest path in graphs**

**Dijkstras shortest path algorithm**

Greedy algoritm. Chooses the lowest cost path locally, to obtain the globally shortest path.

A picture containing text, clock

Description automatically generated

Example of how the greedy algorithm can make mistakes. A->E is clearly the shortest path, but Dijkstra’s algorithm would choose A->B->C->D->E.

Lower time-complexity, but doesn’t work with negative weight. This means that the algorithm always expects that the weight of the path it takes, will only increase. Real life example:

Roadway system. You would never expect taking a certain road would add a negative amount of meters to an edge.



**TODO: SHOW THE SHIT**

|  |  |  |
| --- | --- | --- |
| Vertex | Shortest  Distance  From A | Previous  vertex |
| A | 0 |  |
| B | 3 | D |
| C | 7 | E |
| D | 1 | A |
| E | 2 | D |

Chart, scatter chart

Description automatically generated

**Bellman-ford shortest path algorithm**

Dynamic programming approach

For directed graphs

Works with negative weights

Can handle negative cycles



Bellman Ford algorithm works by overestimating the length of the path from the starting vertex to all other vertices. Then it iteratively relaxes those estimates by finding new paths that are shorter than the previously overestimated paths.

Bellman ford algorithm iterates through the whole graph v(vertices) -1 times. We can make the algorithm better if we make it stop iterating when a certain iteration doesn’t improve the shortest path.

**TODO: DEMONSTRER ALGORITME** Shape

Description automatically generated