Given a positive-weighted graph \( G \) and its node \( s \), compute the shortest distance from \( s \) to every other node.

Works the same way on positive-weighted digraphs.

Problem with negative weights:

Question: how to do this for unweighted graphs?

BFS - faster
Dijkstra’s algorithm

1. for every node let node.setDist(∞) O(n)
2. s.setDist(0) O(1)
3. while there are still unfinalized nodes do O(n) iterations
4. let v be the node that has not been finalized and has the minimum distance
   \( \text{if} \ \text{infinite} \ \text{then} \) O(n) if
5. for every neighbor u of v do \( O(n) \) upper-bound
6. if u.getDist() > w(v,u) + v.setDist() then
   u.setDist(w(v,u) + v.getDist())
7. u.setParent(v)
8. \( O(1) \) 9. finalize v + follow the parent pointers if want to get the path in addition to the distance

- Why it works ?
- Running time ? \( O(n^2) \) - easy implementation if using heap to store non-final nodes then \( O(n \log n + m) \)
Minimum spanning trees

Given is a weighted graph $G$, find a spanning tree $T$ of $G$ with the smallest possible weight (where weight of $T$ is the sum of its edge weights).

Constructing the tree of shortest paths from a given starting node does not work.

Weight of yellow spanning tree: $3+1+3+1 = 8$

Blue is the minimum sp-tree of weight 5
Minimum spanning trees

Kruskal’s algorithm

Idea:
- Sort edges by edge-weights (increasingly)
- Keep adding edges from the smallest, skip an edge if it creates a cycle

Note:
- A greedy algorithm
- Why it works?
- Running time?
Minimum spanning trees

Prim-Jarnik algorithm  

- similar to Dijkstra

replace lines 6 & 7. with

if u.getCost > w(v,u) then
  u.setCost(w(v,u))

Note:
- A greedy algorithm
- Why it works?
- Running time?