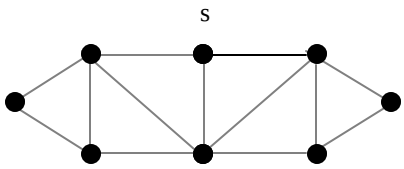
Graph Algorithms

- **Graph traversal**
- Topological sort
- Strongly connected components
- Single-source shortest path

Graph Traversal

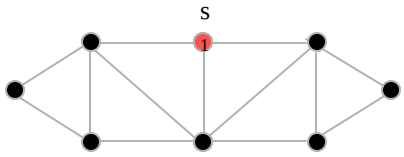
- **Breadth-first**
- Depth-first

Breadth-first(s)



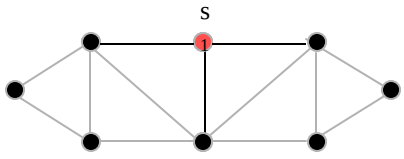
Breadth-first(s)

Q=s



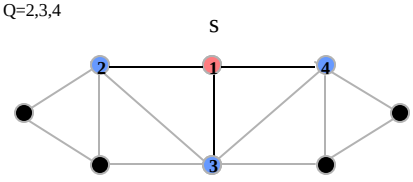
Breadth-first(s)

Q=



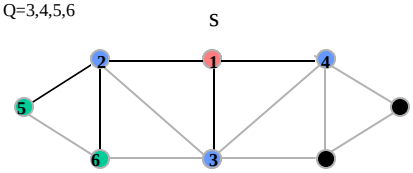
Breadth-first(s)

Q=2,3,4



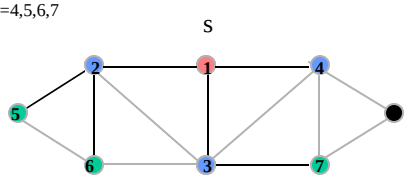
Breadth-first(s)

Q=3,4,5,6



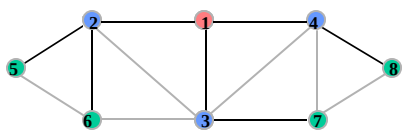
Breadth-first(s)

Q=4,5,6,7



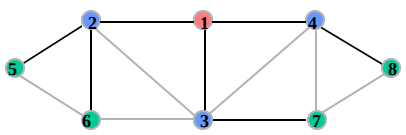
Breadth-first(s)

Q=5,6,7,8



Breadth-first(s)

Q=



Graph Algorithms

- **Graph traversal**
- Topological sort
- Strongly connected components
- **Single-source shortest path**

Single Source Shortest Path

- Input: Graph G with a designated start vertex s.
- Output: For each vertex v, the length of the shortest path between s and v.

Distance

- The distance between vertices v and w in the unweighted graph G is the length of a shortest path between v and w in G.
- We use $d_G(v,w)$ to denote this distance.

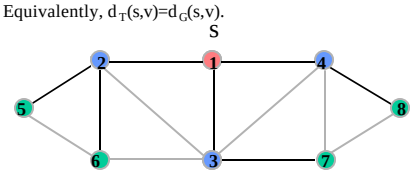
Single Source Shortest Path

- Input: Graph G with a designated start vertex s.
- Output: For each vertex v, $d_G(s,v)$.

Shortest path tree for s

The path between s and v in T is a shortest path in G.

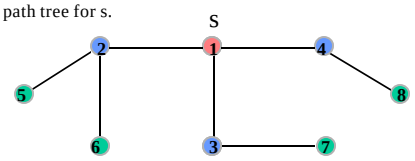
Equivalently, $d_T(s,v)=d_G(s,v)$.



Shortest path tree for s

The path between s and v in T is a shortest path in G.

The edges traversed in Breadth-First(s) form shortest path tree for s.



Shortest path tree for s

- Is it clear such a tree exists?
- Is it clear the Breadth-first(s) finds it?

Shortest path tree for s

- Is it clear such a tree exists?
- Order the vertices of G by distance from s: $v_0=s, v_1, v_2, \ldots, v_n$
- Claim: There is a subtree T of G on vertices $\{v_0, \ldots, v_i\}$ such that for every k, $0 \le k \le i$, $d_T(s,v_k)= d_G(s,v_k)$.

Proof of Claim

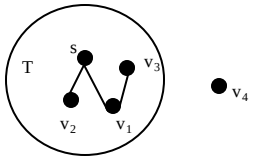
- Claim: There is a subtree T of G on vertices $\{v_0, \ldots, v_i\}$ such that for every k, $0 \le k \le i$, $d_T(s,v_k) = d_G(s,v_k)$.
- Proof: The claim is true for i=0.

Proof of Claim

- Claim: There is a subtree T of G on vertices $\{v_0, \ldots, v_i\}$ such that for every k, $0 \le k \le i$, $d_T(s,v_k) = d_G(s,v_k)$.
- Proof: Now suppose i>0. Assume the claim is true for i-1. We'll show the claim is true for i.

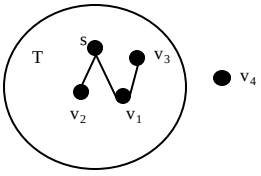
Proof of Claim

- Claim: There is a subtree T of G on vertices $\{v_0, \ldots, v_i\}$ such that for every k, $0 \le k \le i$, $d_T(s,v_k) = d_G(s,v_k)$.
- Proof: Let T be the tree containing $\{v_0, \ldots, v_{i-1}\}$



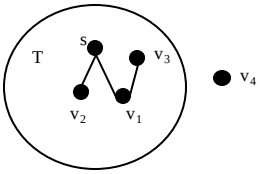
Proof of Claim

- Let s, …, u, v_i be a shortest path between s and v_i in G ($u \neq v_i$, possibly u=s).



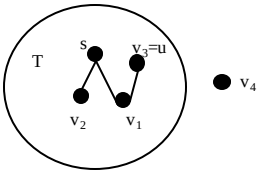
Proof of Claim

- Let s, …, u, v_i be a shortest path between s and v_i in G ($u \neq v_i$, possibly u=s).
- Is it possible that u is not already in T?



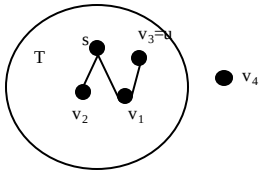
Proof of Claim

- NO: s, … , u, v_i is a shortest path from s to v_i so $d_G(s,v_i) = 1 + d_G(s,u) > d_G(s,u)$.
- Thus u precedes v_i in the ordering of vertices and it must be in T.



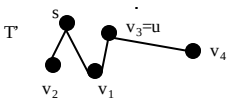
Proof of Claim

- By our induction hypothesis $d_G(s,u)=d_T(s,u)$.



Proof of Claim

- By our induction hypothesis $d_G(s,u)=d_T(s,u)$.
- Let T'=T+{(u,v_i)}. Then $d_{T'}(s,v_i)=1+d_T(s,u)=d_G(s,v_i)$.



QED

- There is a subtree T of G on vertices $\{v_0, \ldots, v_{i+1}\}$ such that for every k, $0 \le k \le i+1$, $d_T(s,v_k)= d_G(s,v_k)$.

Shortest path tree for s

- Is it clear such a tree exists?
- **Is it clear the Breadth-first(s) finds it?**

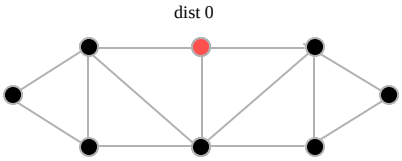
Shortest path tree for s

- Is it clear such a tree exists?
- **Is it clear the Breadth-first(s) finds it?**
  - **The previous proof with some slight modification does it.**

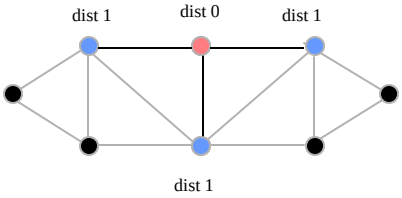
Single Source Shortest Path

- Input: Graph G with a designated start vertex s.
- Output: For each vertex v, the length of the shortest path between s and v.
- Algorithm: Modify Breadth-first to compute d(s,v) along the way.

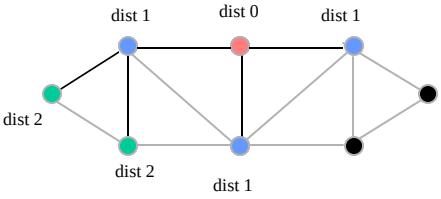
Breadth-first search
Compute distance from s



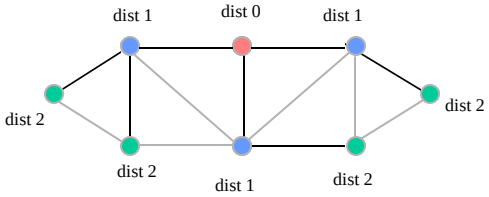
Breadth-first search
Compute distance from s



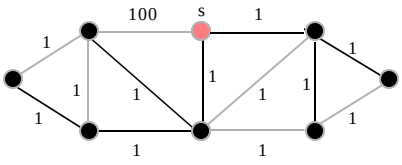
Breadth-first search
Compute distance from s



Breadth-first search
Compute distance from s



What if G is weighted?

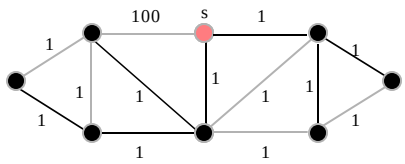


Single Source Shortest Path

- Input: Weighted graph G with a designated start vertex s. Weights are positive!
- Output: For each vertex v, the length of the **shortest path** between s and v.

Shortest path tree for s

The path between s and v in T is a **shortest path** in G.



Shortest path tree for s

- Is it clear such a tree exists? YES by same argument.

Shortest path tree for s

- Is it clear such a tree exists?
- Order the vertices of G by distance from s: $v_0=s, v_1, v_2, \ldots, v_n$
- Claim: There is a subtree T of G on vertices $\{v_0, \ldots, v_i\}$ such that for every k, $0 \le k \le i$, $d_T(s,v_k) = d_G(s,v_k)$.

Algorithm?

- Does this give you any ideas?

Shortest path tree for s

- Claim: There is a subtree T of G on vertices $\{v_0, \ldots, v_i\}$ such that for every k, $0 \le k \le i$, $d_T(s,v_k) = d_G(s,v_k)$.
- What is the tree for $\{v_0\}$?

Shortest path tree for s

- Claim: There is a subtree T of G on vertices $\{v_0, \ldots, v_i\}$ such that for every k, $0 \le k \le i$, $d_T(s,v_k) = d_G(s,v_k)$.
- What is the tree for $\{v_0\}$?

s ●

Shortest path tree for s

- Claim: There is a subtree T of G on vertices $\{v_0, \ldots, v_i\}$ such that for every k, $0 \le k \le i$, $d_T(s,v_k) = d_G(s,v_k)$.
- If you have a tree for $\{v_0, \ldots, v_i\}$ can you find the tree for $\{v_0, \ldots, v_i, v_{i+1}\}$?

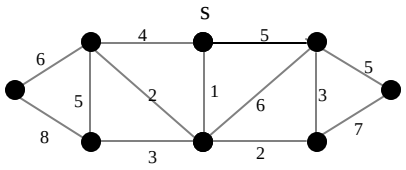
But Wait

- You don't even know which vertex is v_{i+1}!

Shortest path tree for s

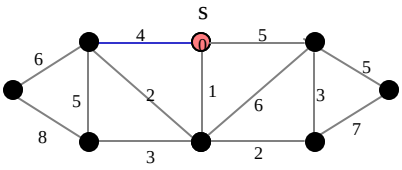
- Claim: There is a subtree T of G on vertices $\{v_0, \ldots, v_i\}$ such that for every k, $0 \le k \le i$, $d_T(s,v_k) = d_G(s,v_k)$.
- If you have a tree for $\{v_0, \ldots, v_i\}$ can you find the tree for $\{v_0, \ldots, v_i, v_{i+1}\}$?
- Find vertex $v \notin T$ that minimizes $\min_{u \in T} d_T(s,u)+w(u,v)$.

Dykstra's Algorithm



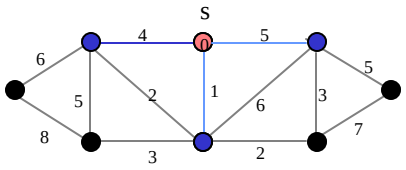
Dykstra's Algorithm

Number in node u indicate $d_G(s,u)$



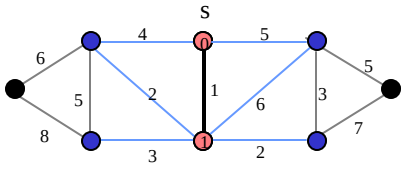
Dykstra's Algorithm

Number in node u indicate $d_G(s,u)$



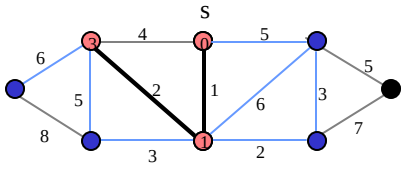
Dykstra's Algorithm

Number in node u indicate $d_G(s,u)$



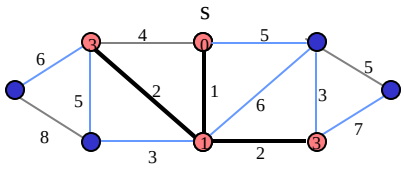
Dykstra's Algorithm

Number in node u indicate $d_G(s,u)$



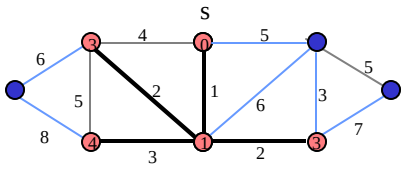
Dykstra's Algorithm

Number in node u indicate $d_G(s,u)$



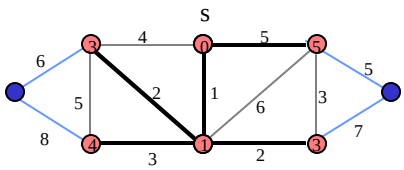
Dykstra's Algorithm

Number in node u indicate $d_G(s,u)$



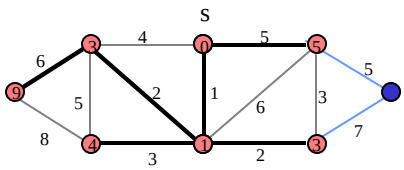
Dykstra's Algorithm

Number in node u indicate $d_G(s,u)$



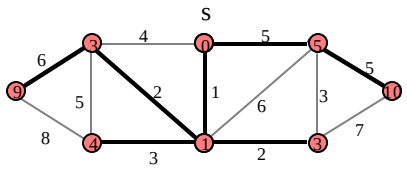
Dykstra's Algorithm

Number in node u indicate $d_G(s,u)$



Dykstra's Algorithm

Number in node u indicate $d_G(s,u)$



Does this sound familiar?

- Prim's algorithm for MST is VERY similar.
- The implementation details are almost identical.