CSCI 241

Scott Wehrwein

Dijkstra's Algorithm: Proof of Correctness

Goals

Understand a proof that Dijkstra's algorithm correctly computes the shortest paths.

Dijkstra's Algorithm

At this point, we know *how* to execute it.

But does it actually *work*?

Why is this OK?

To safely move f to S, we need to know for sure we've found the shortest path to f.

```
S = { }; F = {v};  v.d = 0;
while  (F ≠ {})  {
    f = node in F with min d value;
    Remove f from F, add it to S;
    for each neighbor w of f {
        if (w not in S or F) {
            w.d =  f.d + weight(f, w);
            add w to F;
        } else if (f.d+weight(f,w) < w.d) {
            w.d = f.d+weight(f,w);
        }
    }
}
```

Proof of Correctness

Dijkstra’s algorithm is greedy: it makes a sequence of *locally* optimal moves, which results in the *globally* optimal solution.

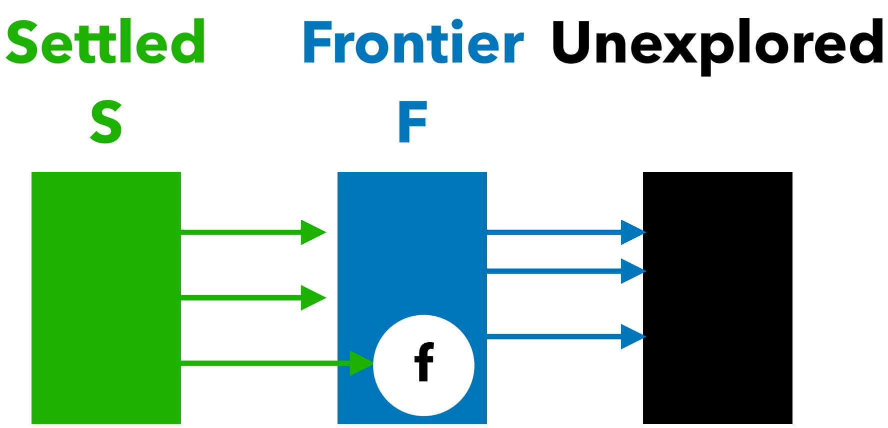
- **In general, this strategy doesn't work!**
- To credibly claim it works here, we need to prove it.

Specifically: It is not obvious that there cannot still be a shorter path to the Frontier node with smallest d-value.

Proof Sketch

1. State a loop invariant.
2. Prove that **if** that invariant is maintained, **then** the algorithm is correct.
3. Prove that the algorithm maintains the invariant.

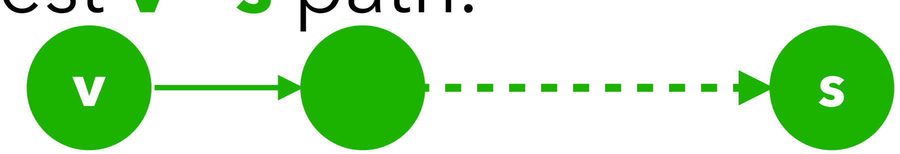
Proof of Correctness: Invariant



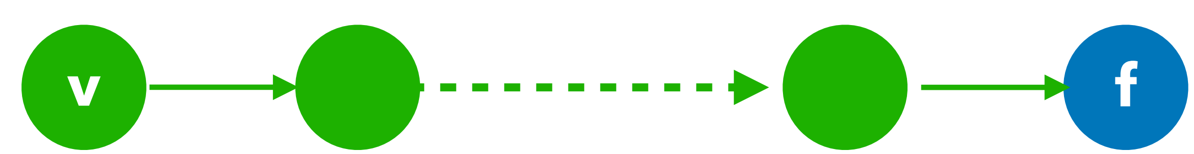
1. **State invariant.**
2. If invariant maintained, then alg is correct.
3. Invariant is maintained

The while loop in Dijkstra's algorithm maintains the following **3-part invariant**:

1. For a Settled node **s**, a shortest path from **v** to **s** contains only settled nodes and **s.d** is length of shortest **v** - **s** path.



2. For a Frontier node **f**, at least one **v** -> **f** path contains only settled nodes (except perhaps for **f**) and **f.d** is the length of the shortest such path



3. All edges leaving S go to F (i.e., no edges from S to Unexplored)

Proof of Correctness: Theorem

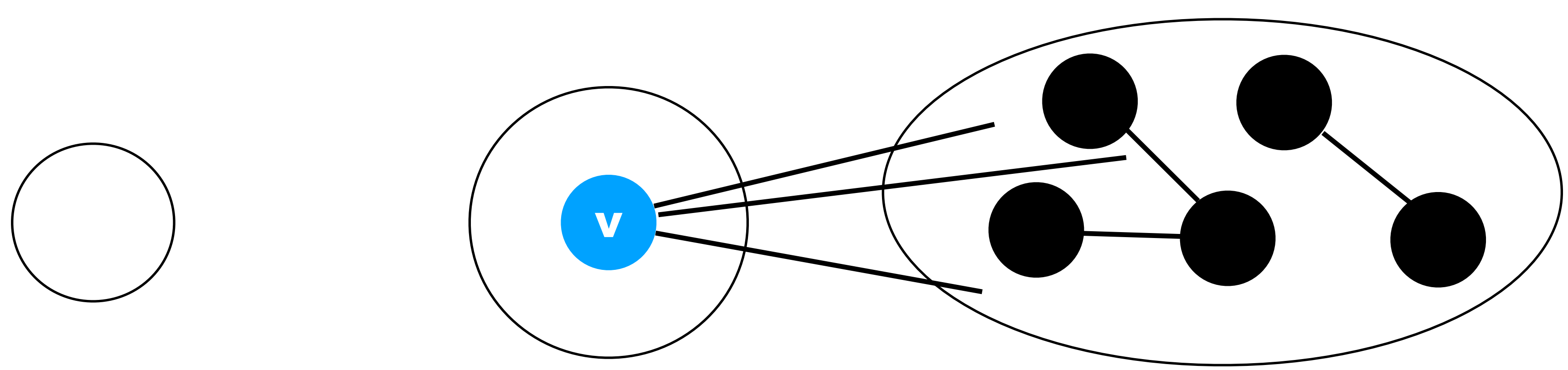
```
S = { }; F = {v};  v.d = 0;
while  (F ≠ {})  {
   f = node in F with min d value;
   Remove f from F, add it to S;
   for each neighbor w of f {
      if (w not in S or F) {
         w.d =  f.d + weight(f, w);
         add w to F;
      } else if (f.d+weight(f,w) < w.d) {
         w.d = f.d+weight(f,w);
      }
   }
}
```

1. State invariant.
2. **If invariant maintained, then alg is correct.**
3. Invariant is maintained

Theorem: For a node f in the Frontier with minimum d value (over all nodes in the Frontier), f.d is the shortest-path distance from v to f.

Proof: Show that any other path from v to if has length >= f.d

Case 1: if **v** is in F, then S is empty and **v.d** = 0, which is trivially the shortest distance from **v** to **v**.



Proof of Correctness: Theorem

```
S = { }; F = {v};  v.d = 0;
while  (F ≠ {})  {
   f = node in F with min d value;
   Remove f from F, add it to S;
   for each neighbor w of f {
      if (w not in S or F) {
         w.d =  f.d + weight(f, w);
         add w to F;
      } else if (f.d+weight(f,w) < w.d) {
         w.d = f.d+weight(f,w);
      }
   }
}
```

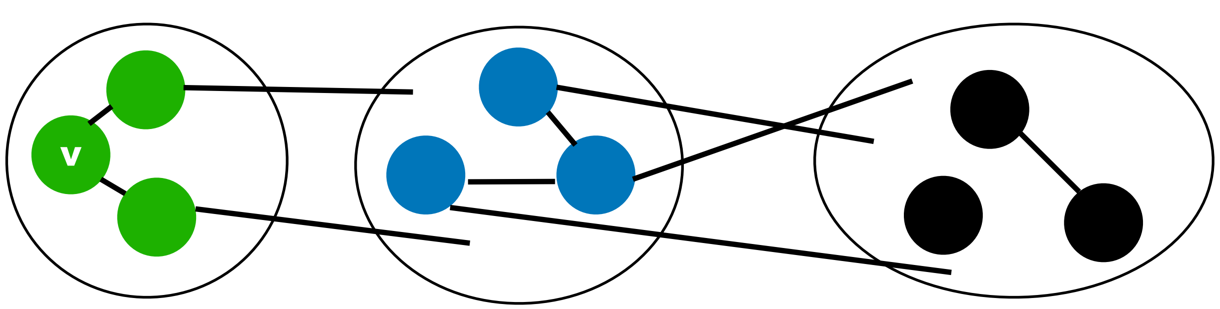
1. State invariant.
2. **If invariant maintained, then alg is correct.**
3. Invariant is maintained

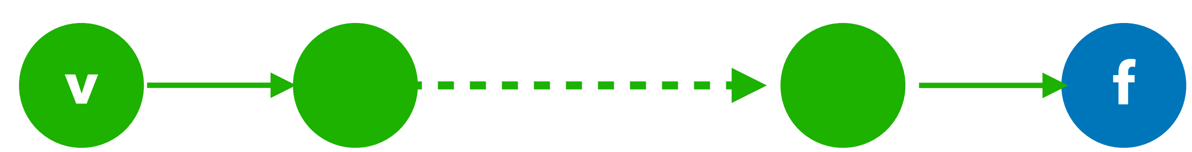
Theorem: For a node f in the Frontier with minimum d value (over all nodes in the Frontier), f.d is the shortest-path distance from v to f.

Proof: Show that any other path from v to if has length >= f.d

Case 2: v is in **S**. Part 2 of the invariant says:

- **f.d** is the length of the shortest path from **v** to **f** containing all settled nodes except **f**, and **f.d** is the length of such a path.





Proof of Correctness: Theorem

```
S = { }; F = {v};  v.d = 0;
while  (F ≠ {})  {
   f = node in F with min d value;
   Remove f from F, add it to S;
   for each neighbor w of f {
      if (w not in S or F) {
         w.d =  f.d + weight(f, w);
         add w to F;
      } else if (f.d+weight(f,w) < w.d) {
         w.d = f.d+weight(f,w);
      }
   }
}
```

1. State invariant.
2. **If invariant maintained, then alg is correct.**
3. Invariant is maintained

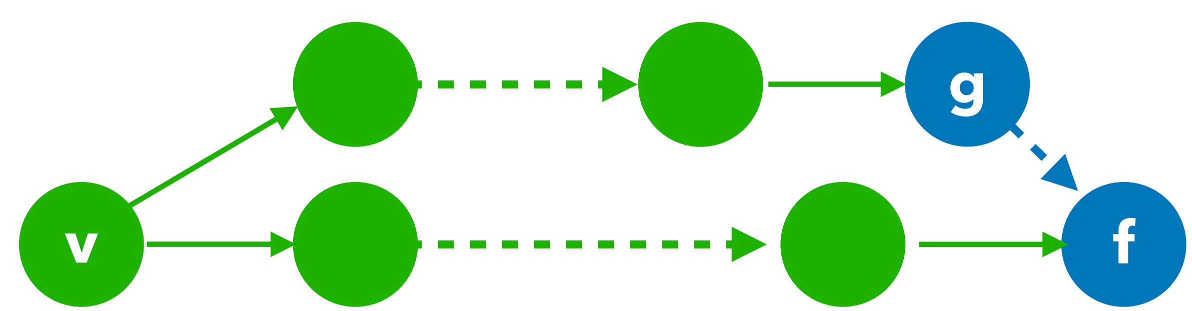
Theorem: For a node f in the Frontier with minimum d value (over all nodes in the Frontier), f.d is the shortest-path distance from v to f.

Proof: Show that any other path from v to if has length >= f.d

Case 2: v is in **S**. Part 2 of the invariant says:

- **f.d** is the length of the shortest path from **v** to **f** containing all settled nodes except **f**, and **f.d** is the length of such a path.

Any other **v**-**f** path must either be longer or go through another frontier node **g** before arriving at **f**:



Proof of Correctness: Theorem

1. State invariant.
2. **If invariant maintained, then alg is correct.**
3. Invariant is maintained

```
S = { }; F = {v};  v.d = 0;
while  (F ≠ {})  {
   f = node in F with min d value;
   Remove f from F, add it to S;
   for each neighbor w of f {
      if (w not in S or F) {
         w.d =  f.d + weight(f, w);
         add w to F;
      } else if (f.d+weight(f,w) < w.d) {
         w.d = f.d+weight(f,w);
      }
   }
}
```

Theorem: For a node **f** in the Frontier with minimum d value (over all nodes in the Frontier), **f.d** is the shortest-path distance from **v** to **f**.

Proof: Show that any other path from **v** to if has length >= **f.d**

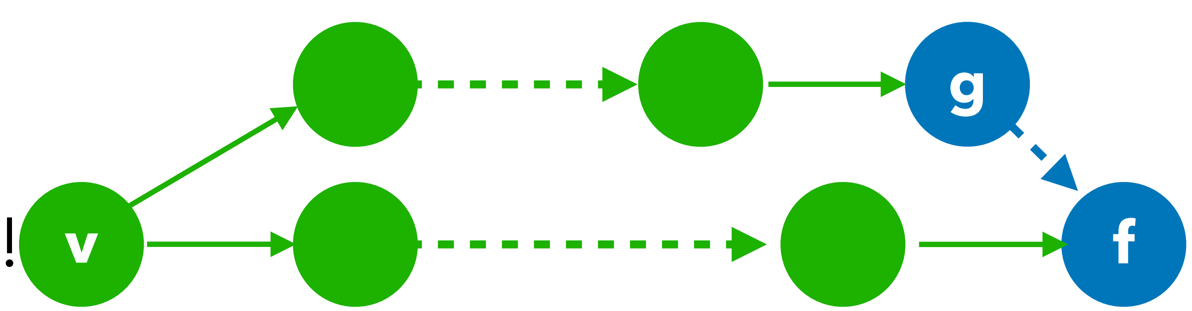
Case 2: v is in **S**. Part 2 of the invariant says:

- **f.d** is the length of the shortest path from **v** to **f** containing all settled nodes except **f**, and **f.d** is the length of such a path.

Any other **v**-**f** path must either be longer or go through another frontier node **g** before arriving at **f**:

but: f.d <= **g.d**,

so the path through g can't be shorter!



Proof of Correctness: Invariant Maintenance

1. State invariant.
2. If invariant maintained, then alg is correct.
3. **Invariant is maintained**

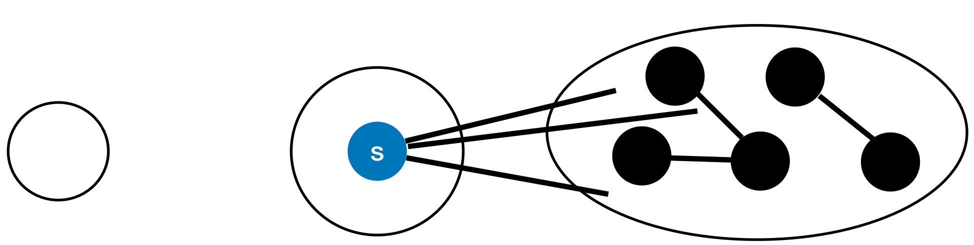
```
S = { }; F = {v};  v.d = 0;
while  (F ≠ {})  {
   f = node in F with min d value;
   Remove f from F, add it to S;
   for each neighbor w of f {
      if (w not in S or F) {
         w.d =  f.d + weight(f, w);
         add w to F;
      } else if (f.d+weight(f,w) < w.d) {
         w.d = f.d+weight(f,w);
      }
   }
}
```

1. For a Settled node s, a shortest path from **v** to **s** contains only settled nodes and **s.d** is length of shortest **v** -> **s** path.
2. For a Frontier node f, at least one **v** -> **f** path contains only settled nodes (except perhaps for **f**) and **f.d** is the length of the shortest such path
3. All edges leaving S go to F (or: no edges from S to Unexplored)

Proof of Correctness: Invariant Maintenance

1. State invariant.
2. If invariant maintained, then alg is correct.
3. **Invariant is maintained**

```
S = { }; F = {v};  v.d = 0;
while  (F ≠ {})  {
  f = node in F with min d value;
  Remove f from F, add it to S;
  for each neighbor w of f {
    if (w not in S or F) {
      w.d =  f.d + weight(f, w);
      add w to F;
    } else if (f.d+weight(f,w) < w.d) {
      w.d = f.d+weight(f,w);
    }
  }
}
```



1. For a Settled node s, a shortest path from **v** to **s** contains only settled nodes and **s.d** is length of shortest **v** -> **s** path.
2. For a Frontier node f, at least one **v** -> **f** path contains only settled nodes (except perhaps for **f**) and **f.d** is the length of the shortest such path
3. All edges leaving S go to F (or: no edges from S to Unexplored)

At initialization:

1. S is empty; trivially true.
2. **v.d** = 0, which is the shortest path.
3. S is empty, so no edges leave it.

Proof of Correctness: Invariant Maintenance

1. State invariant.
2. If invariant maintained, then alg is correct.
3. **Invariant is maintained**

```
S = { }; F = {v};  v.d = 0;
while  (F ≠ {})  {
  f = node in F with min d value;
  Remove f from F, add it to S;
  for each neighbor w of f {
    if (w not in S or F) {
      w.d =  f.d + weight(f, w);
      add w to F;
    } else if (f.d+weight(f,w) < w.d) {
      w.d = f.d+weight(f,w);
    }
  }
}
```

1. For a Settled node s, a shortest path from **v** to **s** contains only settled nodes and **s.d** is length of shortest **v** -> **s** path.
2. For a Frontier node f, at least one **v** -> **f** path contains only settled nodes (except perhaps for **f**) and **f.d** is the length of the shortest such path
3. All edges leaving S go to F (or: no edges from S to Unexplored)

At each iteration:

1. Theorem says **f.d** is the shortest distance to **f**: safe to move to S
2. Updating **w.d** maintains Part 2 of the invariant.
3. Each neighbor is either already in F or gets moved there.

We're done!

What just happened?

1. State a loop invariant.

2. Prove that **if** that invariant is maintained, **then** the algorithm is correct.

   the min d-valued node in F can be moved to S

3. Prove that the algorithm maintains the invariant.

   the invariant is true at the start and after each iteration