Announcements

- Homework 1 Solutions online
- Homework 2 online due Friday
- Dian’s OH changed
  - Now (probably) Tuesday 2-4, Thursday 2-3
- Stanislaw’s OH changed
  - Now Tuesday 5:30-6:30

HW0 Q3 (the graph one)

- Induction on graph
  - Induct on n = |V|
  - Assume true for all graphs with n vertices
  - Given G with n+1 vertices
  - G-v has n, apply inductive hypothesis
  - Add v back
  - DO NOT: start with graph on n vertices and add one more
- Note: removing a vertex does not necessarily decrease the max degree

General HW Reminder

- In order to get full credit, algorithm problems always need (unless otherwise specified):
  - The algorithm you are using
  - A proof of correctness (i.e. a proof that your algorithm correctly computes the thing that it is supposed to)
  - A proof of an appropriate runtime bound

Last Time

- Shortest Paths in Graphs
- BFS
  - Computes minimum number of edges from s to each other vertex in graph
  - At distance d only if adjacent to distance d-1
  - $O(|V|+|E|)$

Today

- Shortest paths with edge lengths
- Dijkstra’s algorithm
- Priority queues

Edge Lengths

The number of edges in a path is not always the right measure of distance. Sometimes, taking several shorter steps is preferable to taking a few longer ones.

Edge Lengths

The number of edges in a path is not always the right measure of distance. Sometimes, taking several shorter steps is preferable to taking a few longer ones.

We assign each edge (u,v) a non-negative <u>length</u> $\ell(u,v)$. The length of a path is the sum of the lengths of its edges.

Problem: Shortest Paths

Problem: Given a Graph G with vertices s and t and a length function ℓ, find the shortest path from s to t.

Question: Shortest Path

What is the length of the shortest s-t path below?

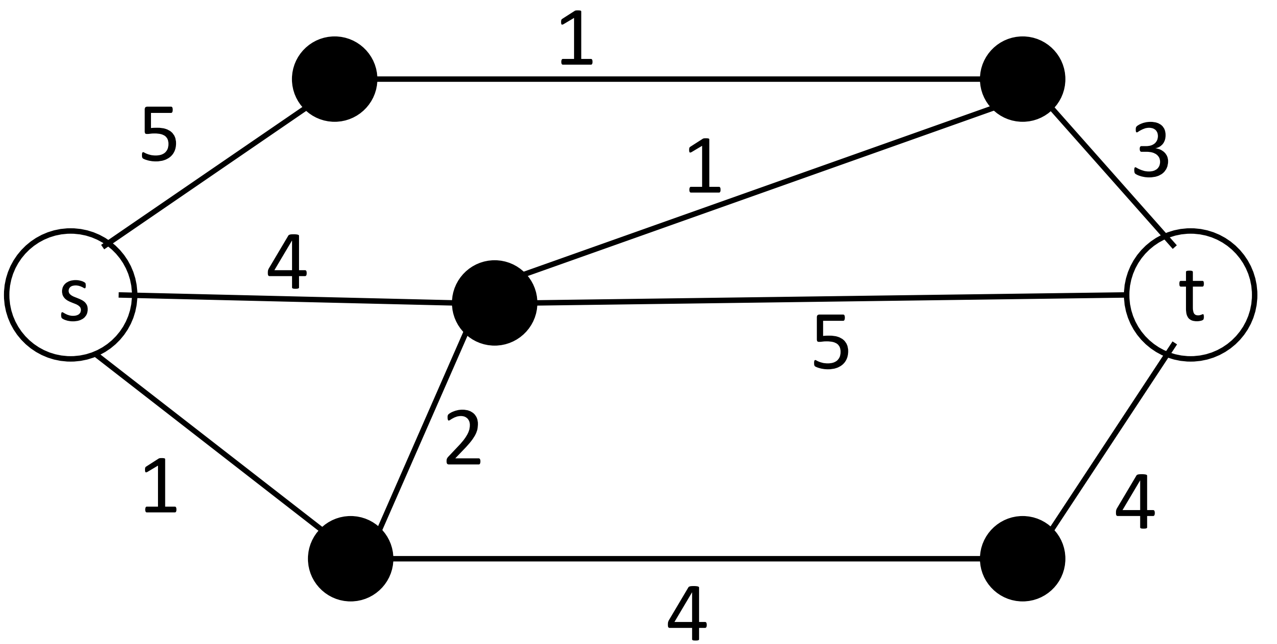
A) 4

B) 5

C) 6

D) 7

E) 8



Question: Shortest Path

What is the length of the shortest s-t path below?

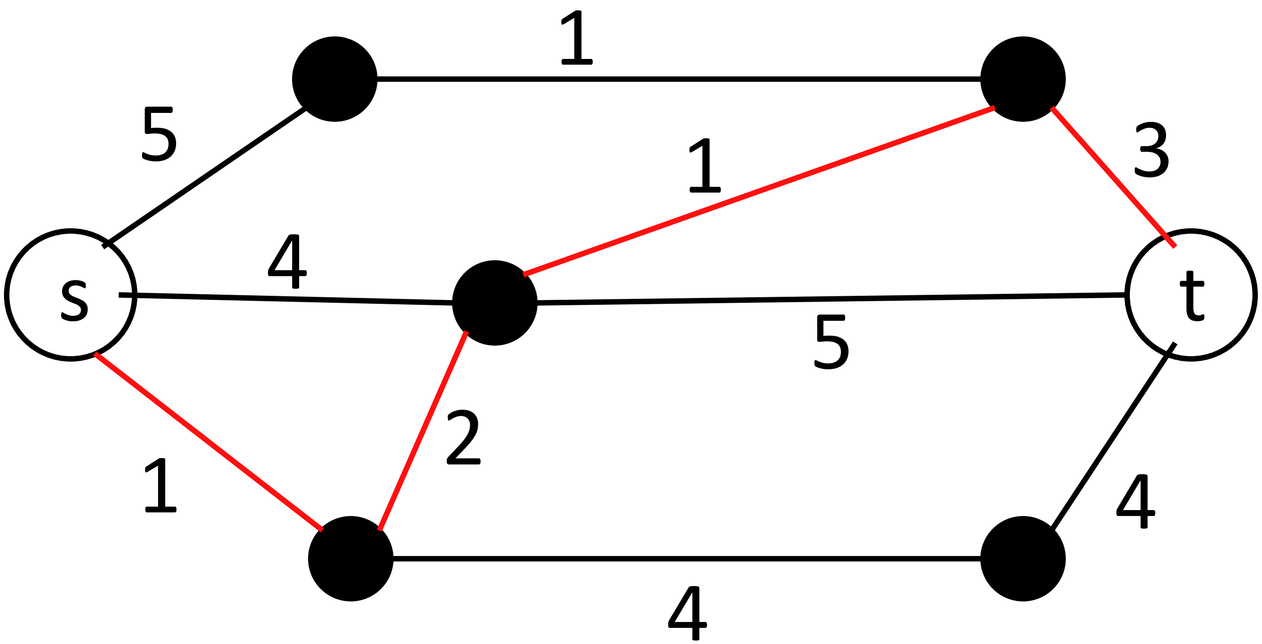
A) 4

B) 5

C) 6

D) 7

E) 8



Question: Shortest Path

What is the length of the shortest s-t path below?

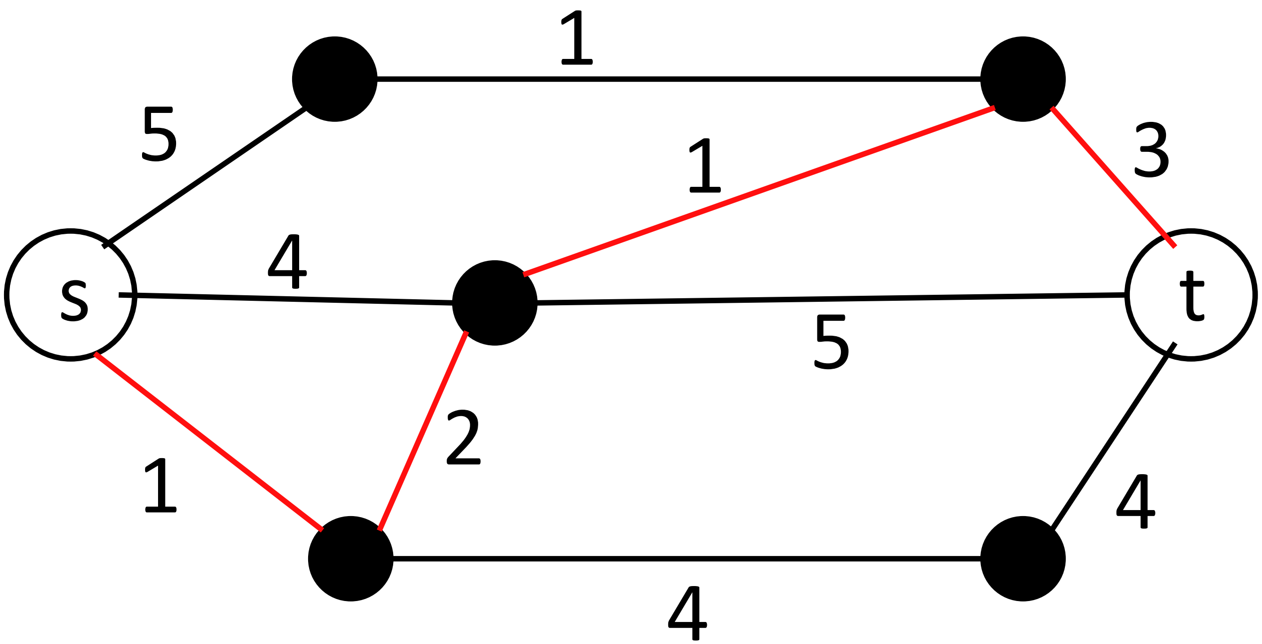
A) 4

B) 5

C) 6

D) 7

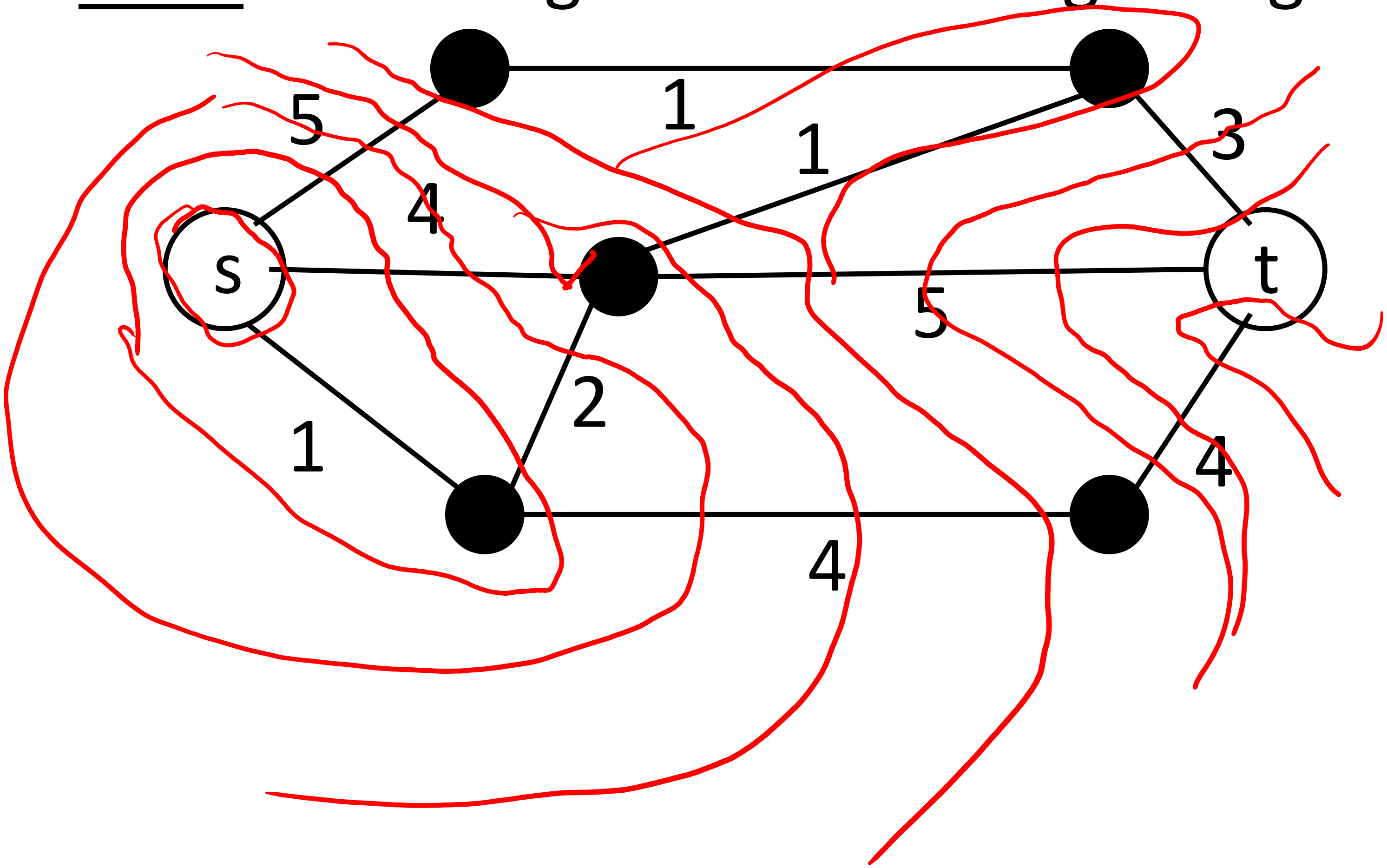
E) 8



1+2+1+3 = 7

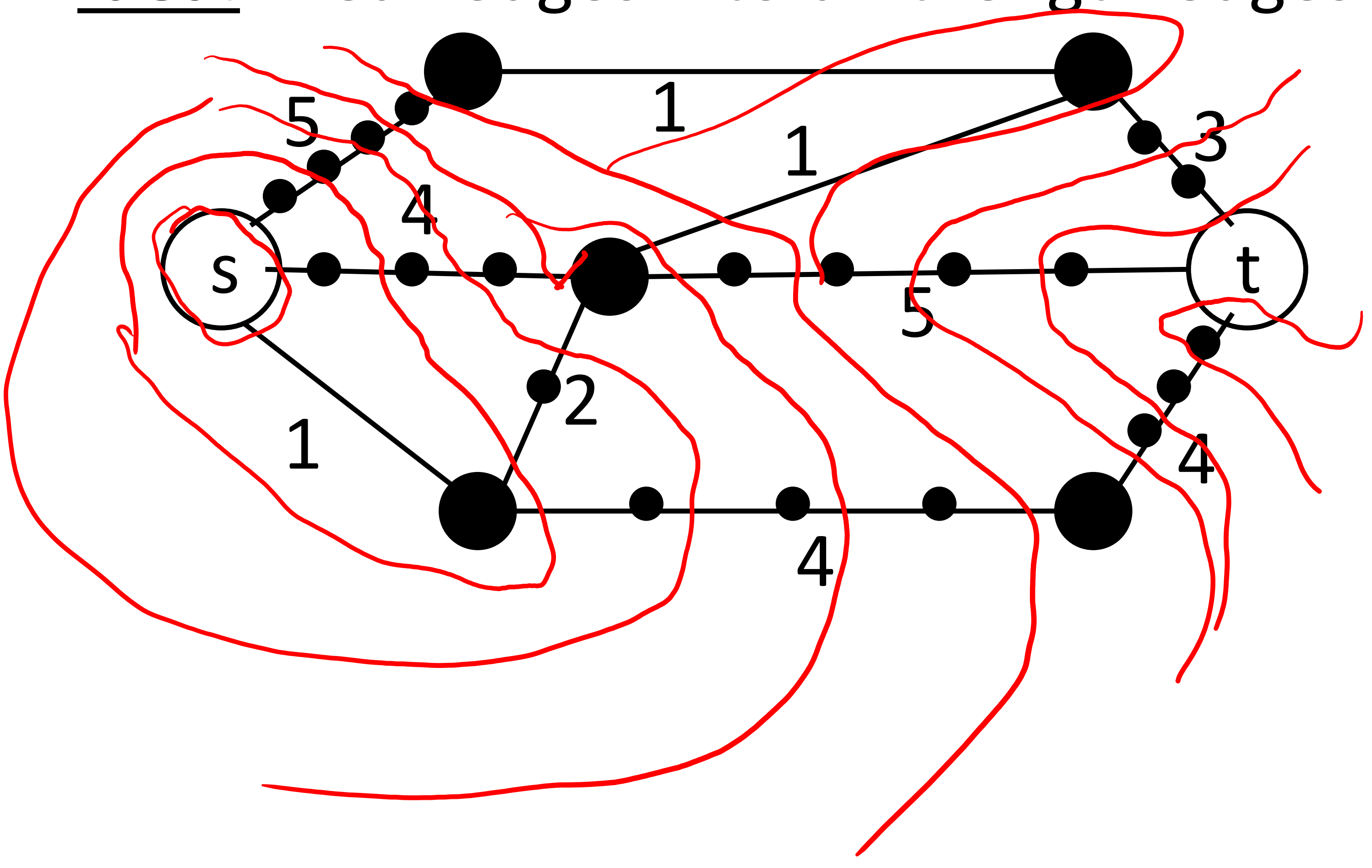
How to Compute

Idea: Break edges into unit length edges.



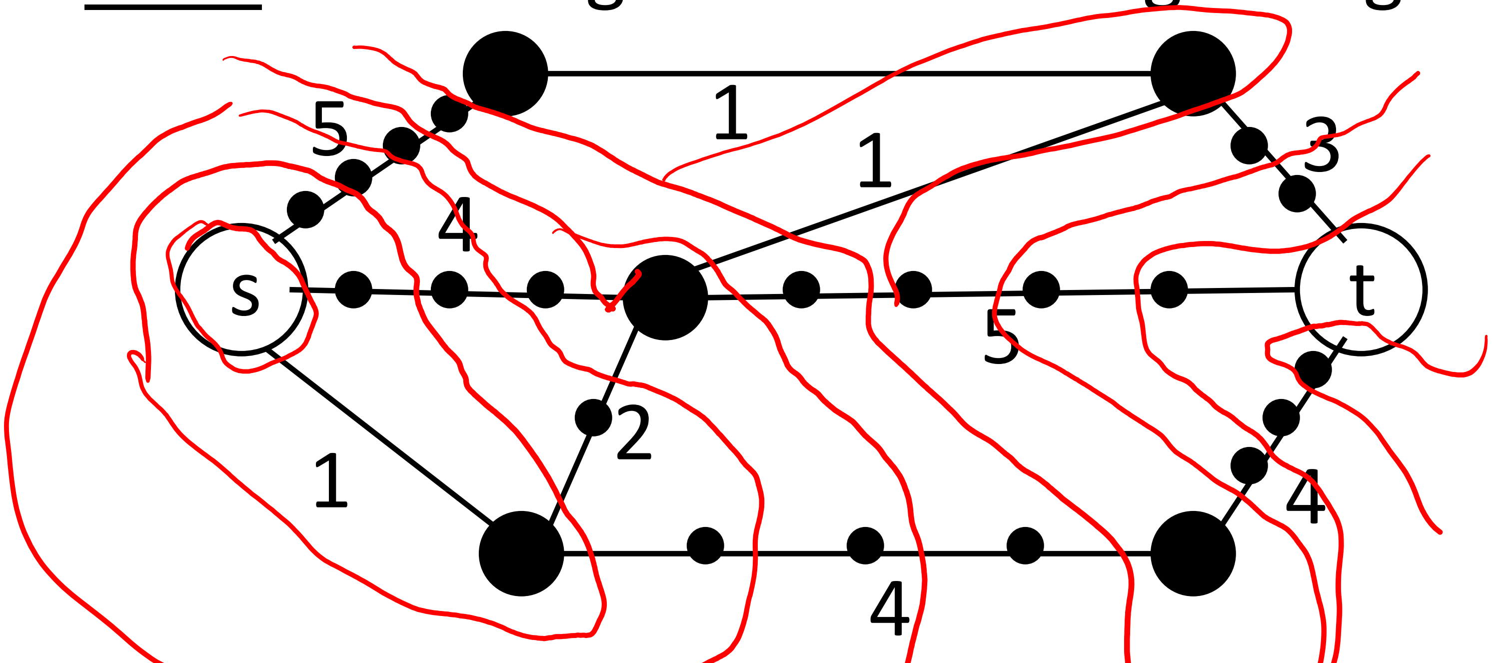
How to Compute

Idea: Break edges into unit length edges.



How to Compute

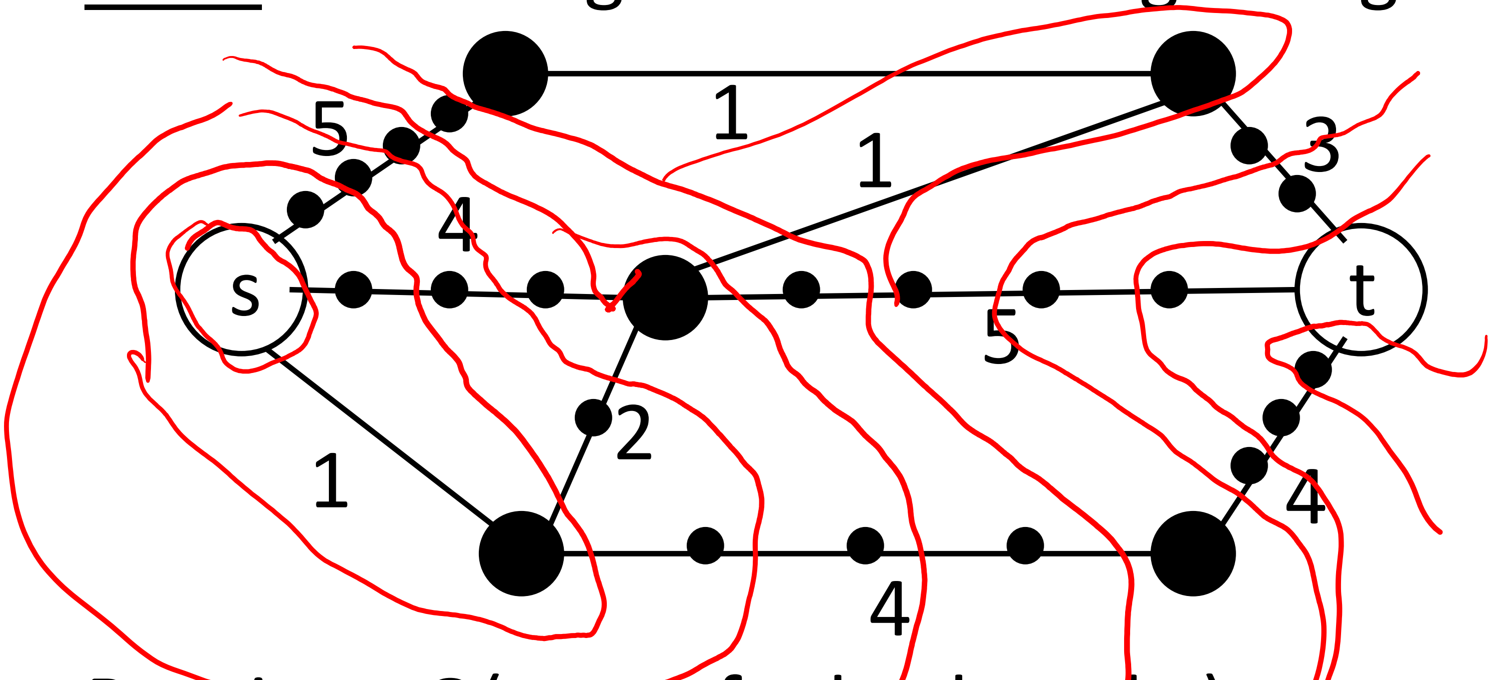
<u>Idea:</u> Break edges into unit length edges.



Runtime: O(sum of edge lengths)

How to Compute

Idea: Break edges into unit length edges.



Runtime: O(sum of edge lengths)

This is a problem if some edge lengths are large.

Another Way

- If you have very long edge lengths, most steps will just consist of advancing slightly along a bunch of edges.

Another Way

If you have very long edge lengths, most steps will just consist of advancing slightly along a bunch of edges.

Try to find a way to fast forward through these boring steps.

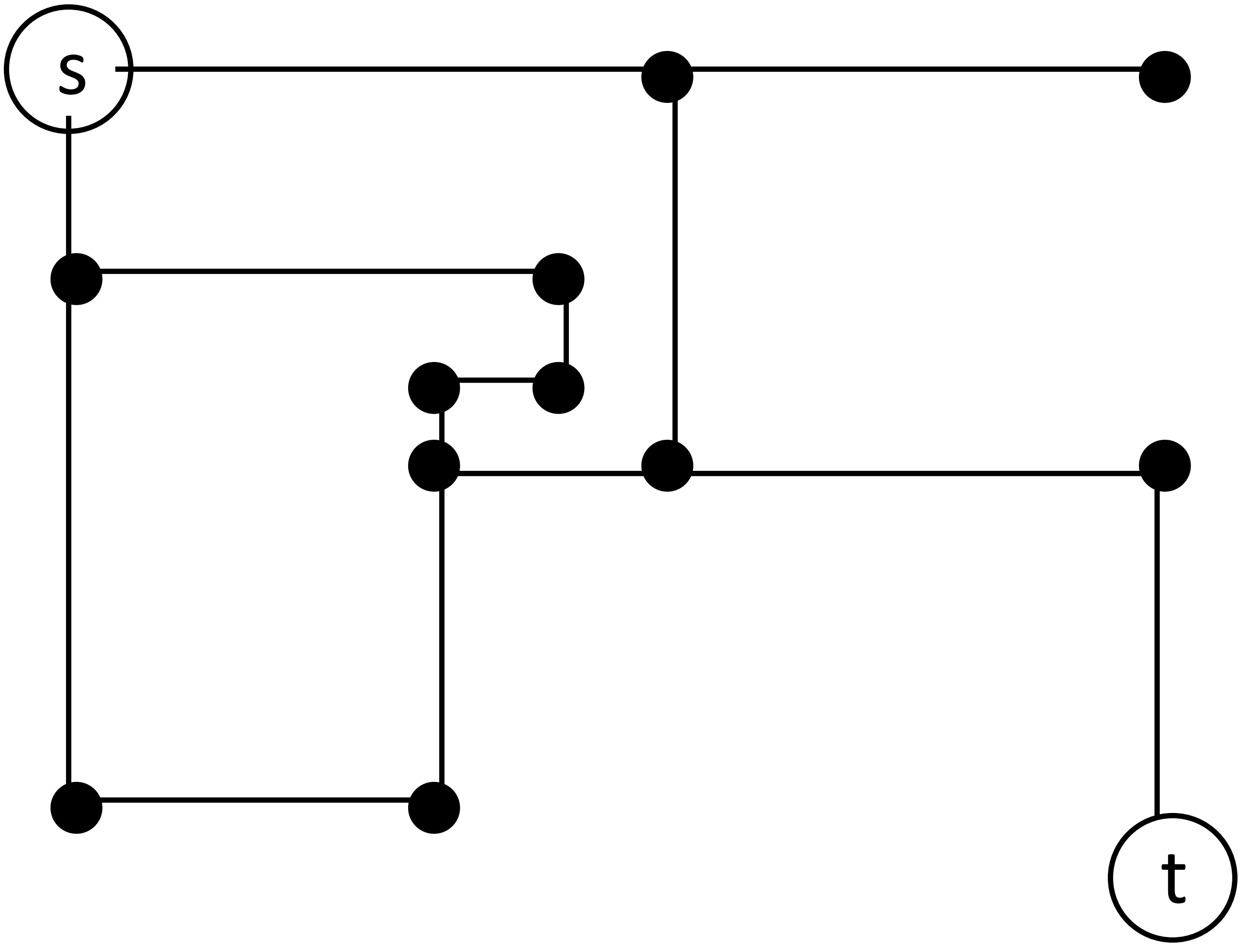
Another Way

If you have very long edge lengths, most steps will just consist of advancing slightly along a bunch of edges.

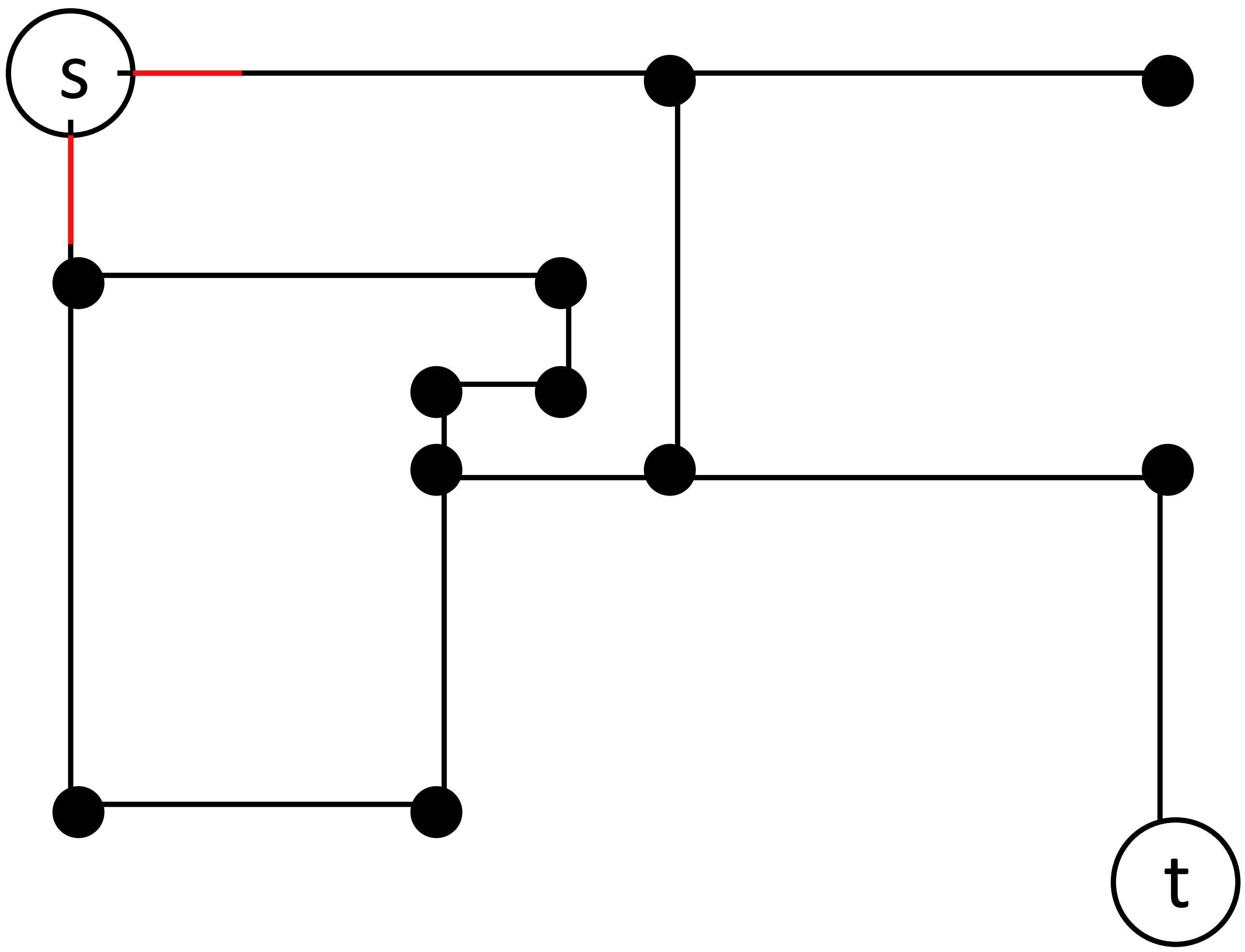
Try to find a way to fast forward through these boring steps.

Occasionally have interesting steps where the wavefront hits a new vertex.

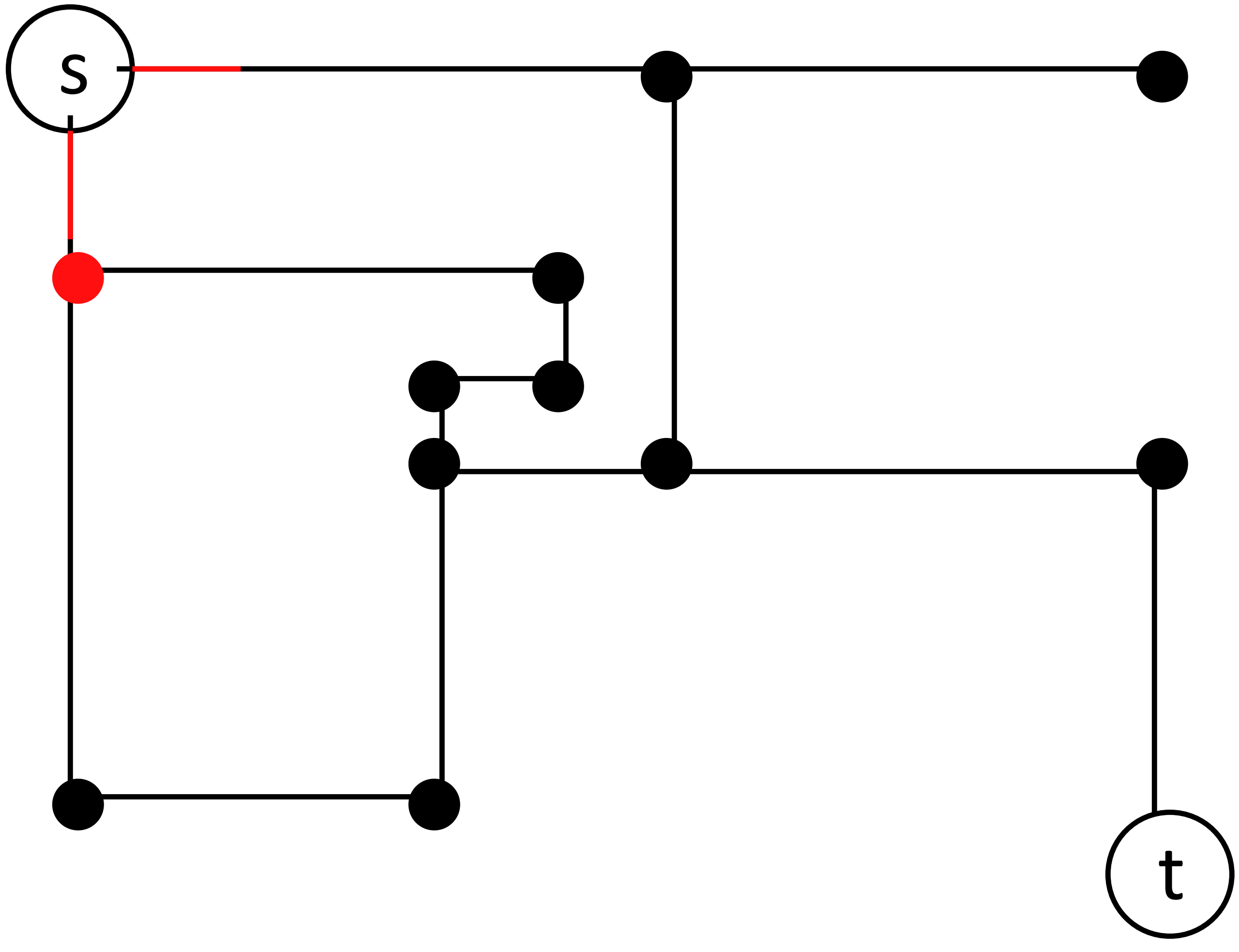
Ooze



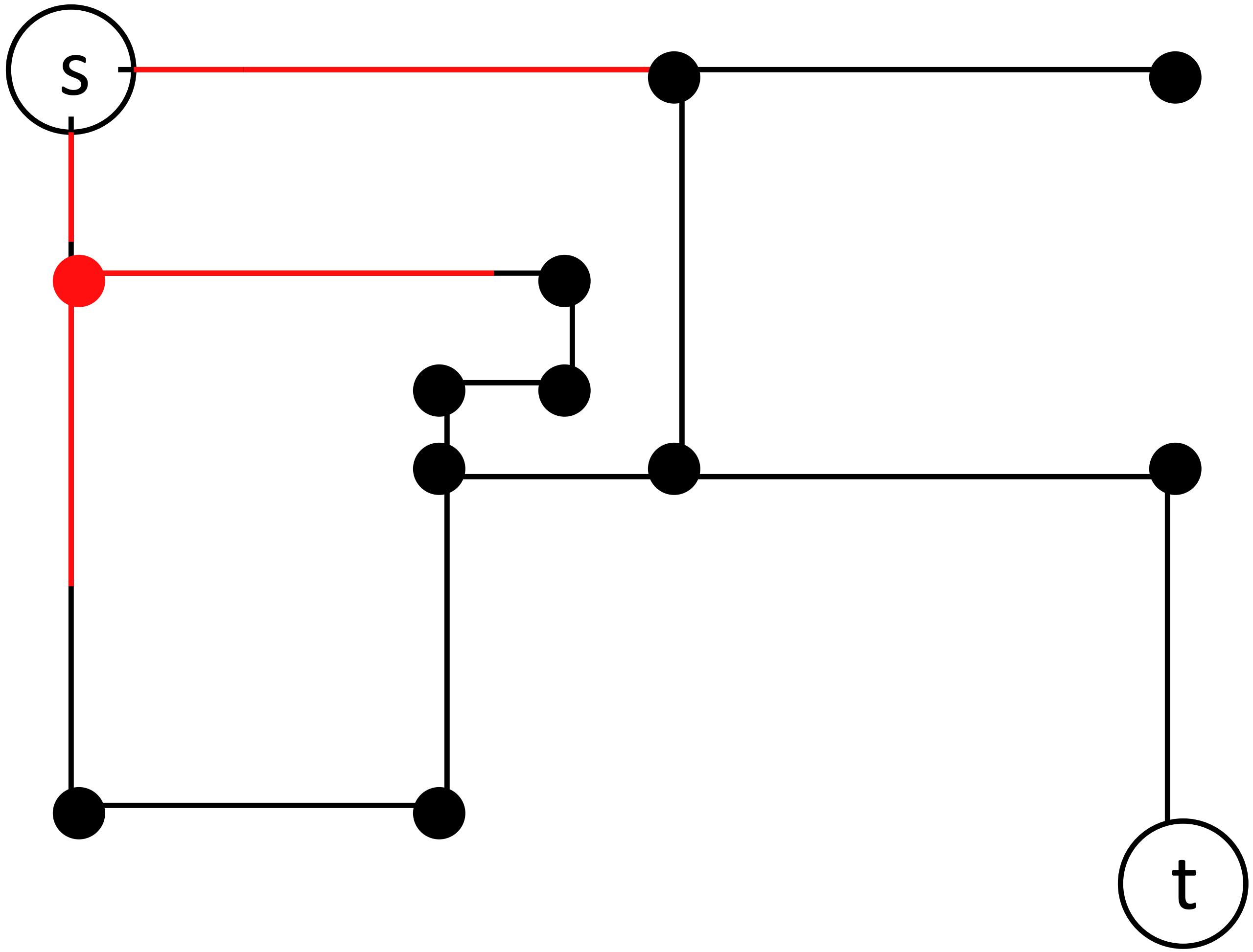
Ooze



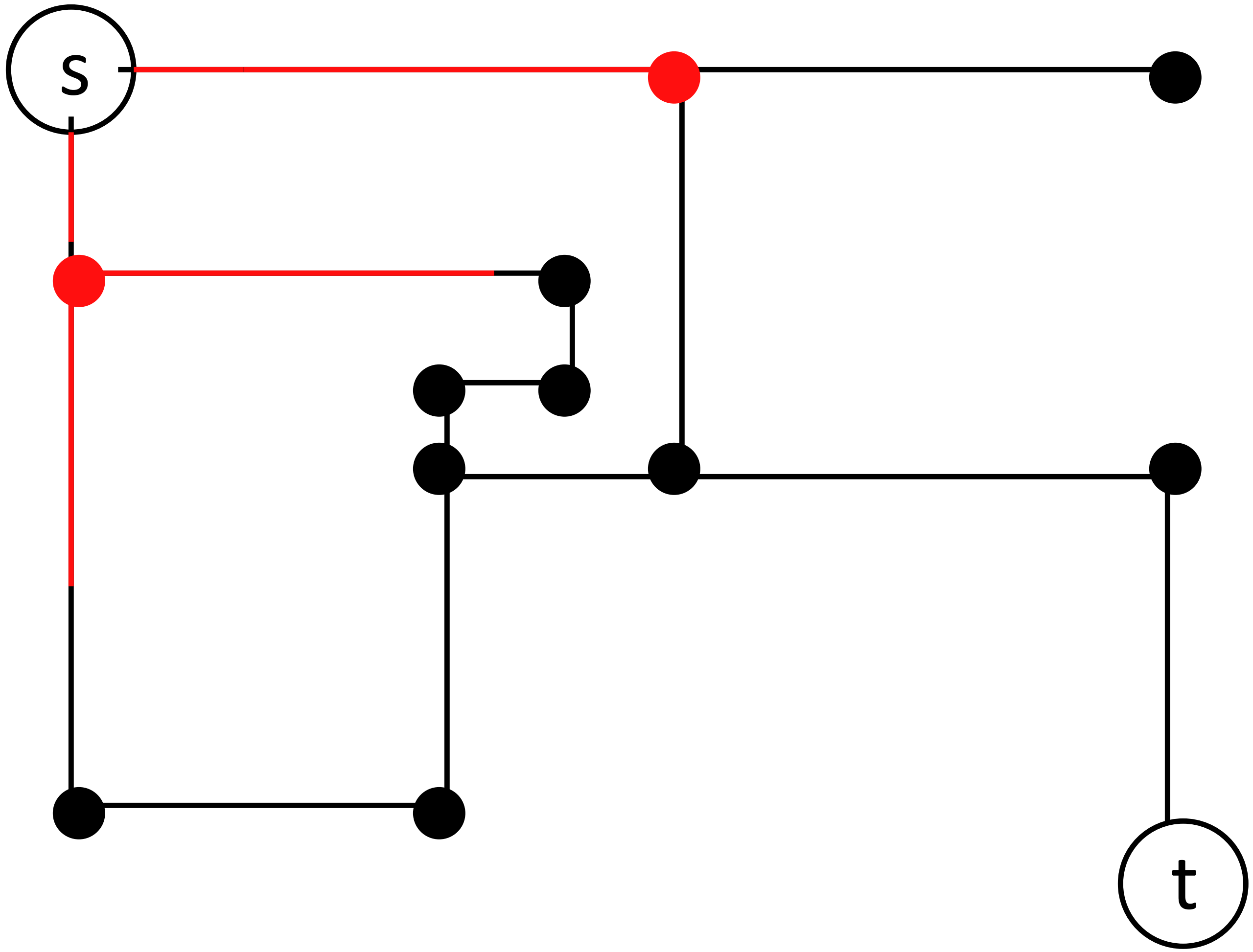
Ooze



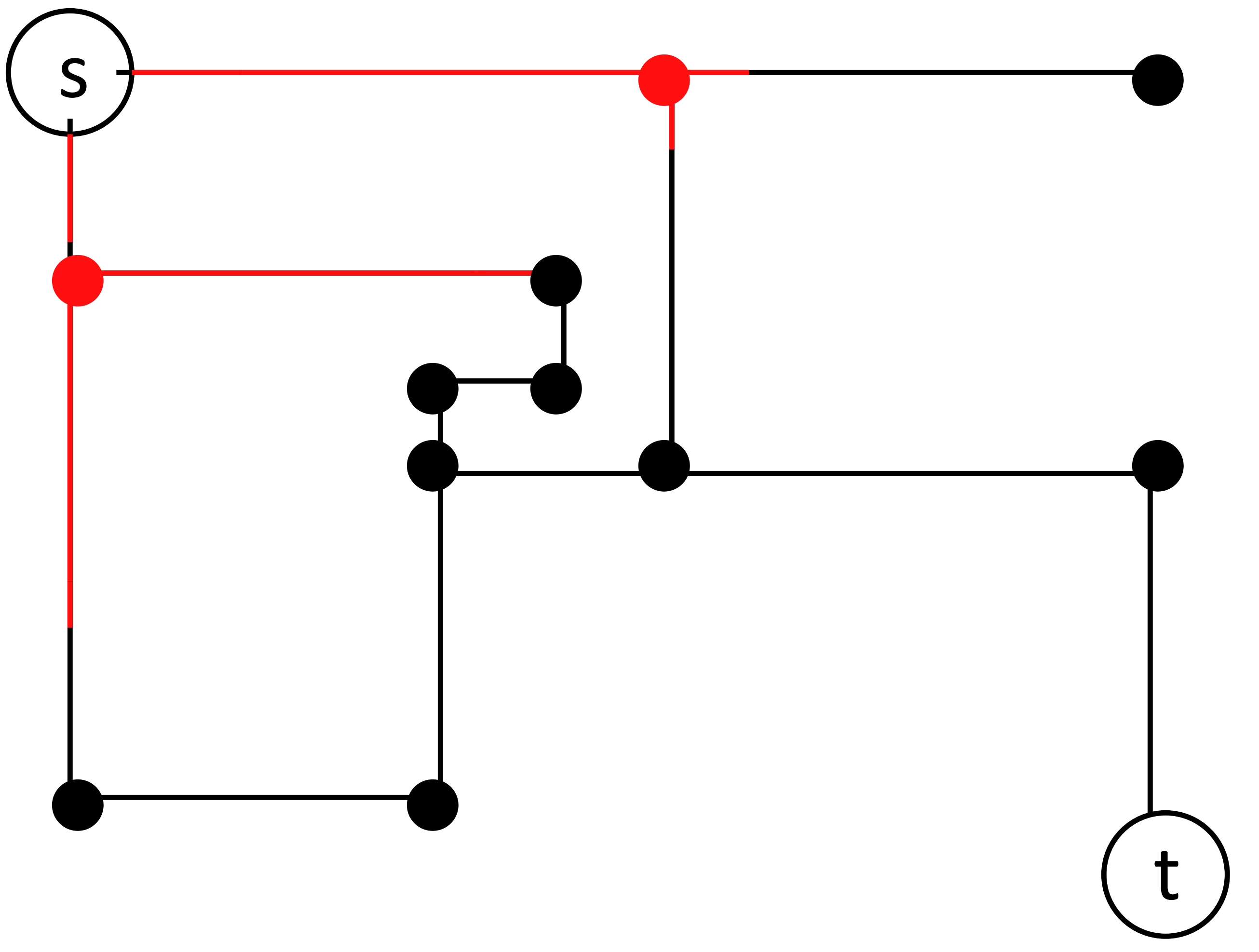
Ooze



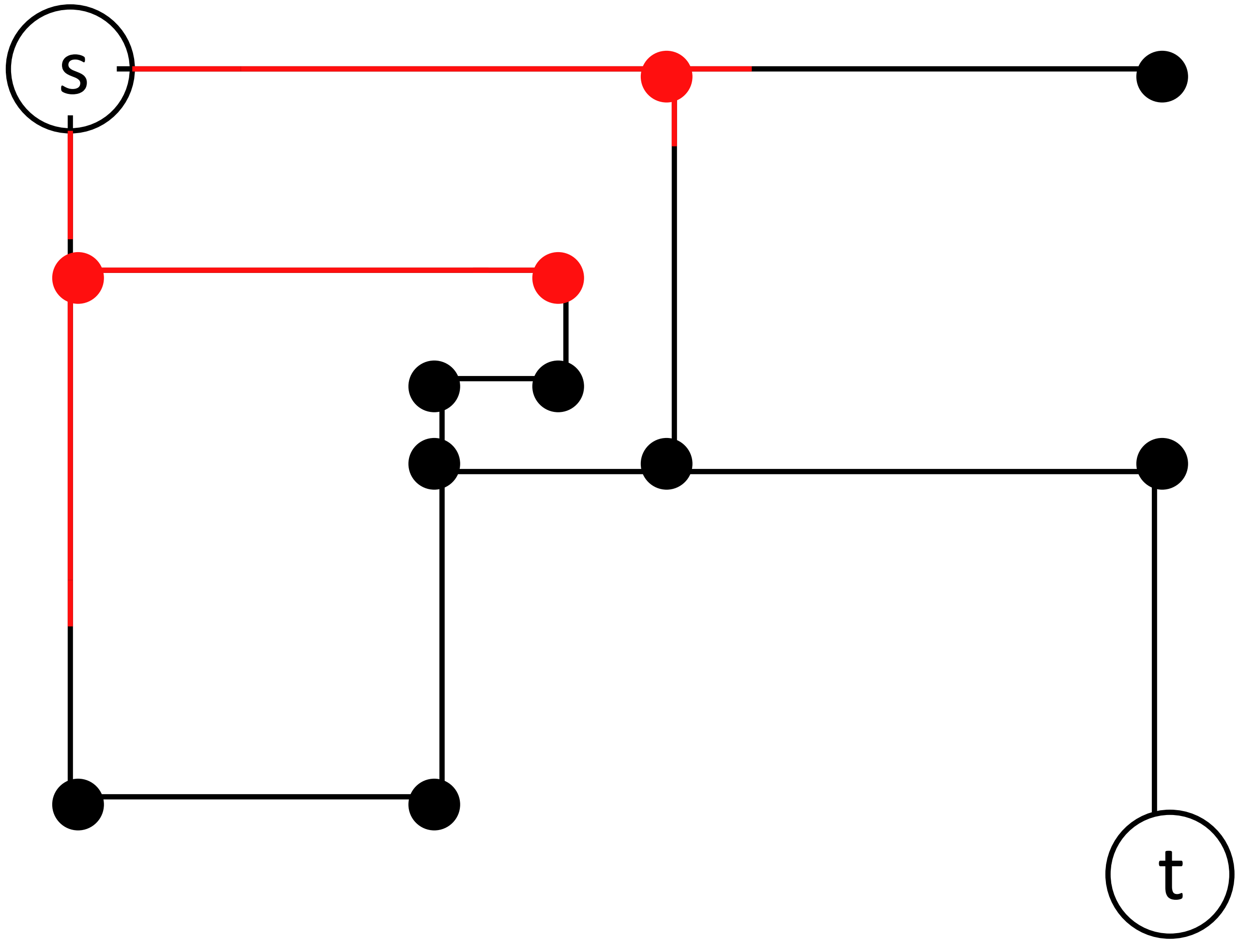
Ooze



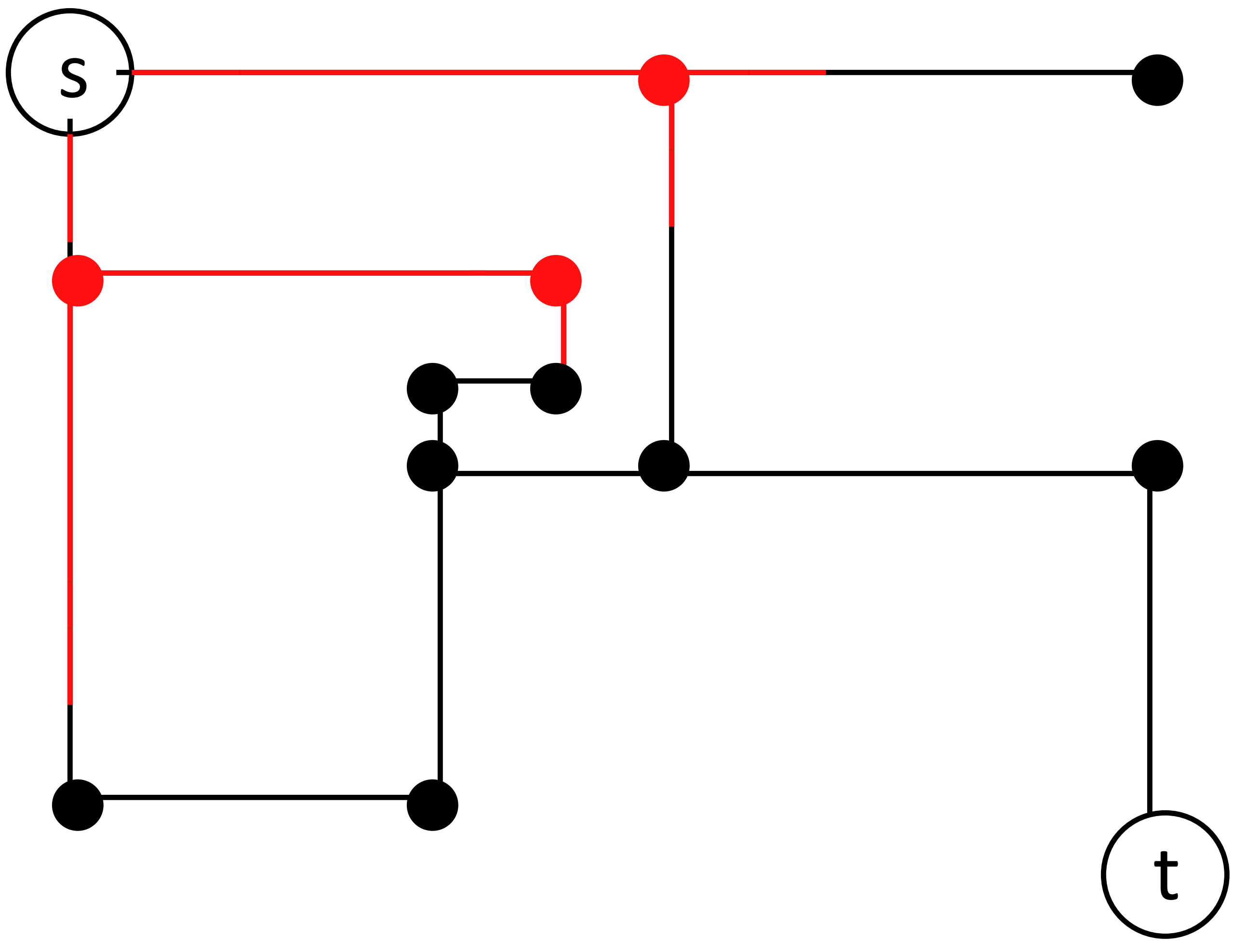
Ooze



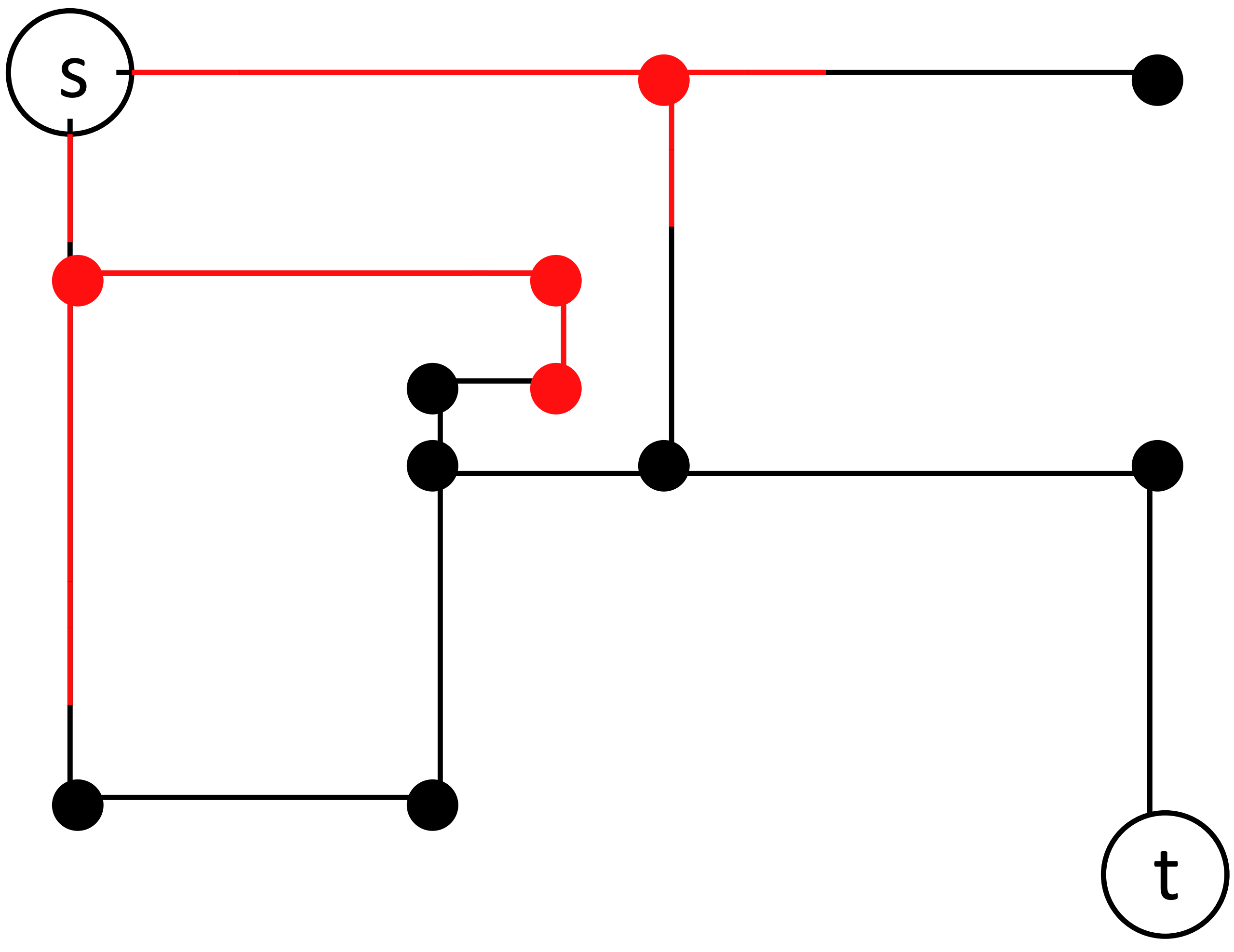
Ooze



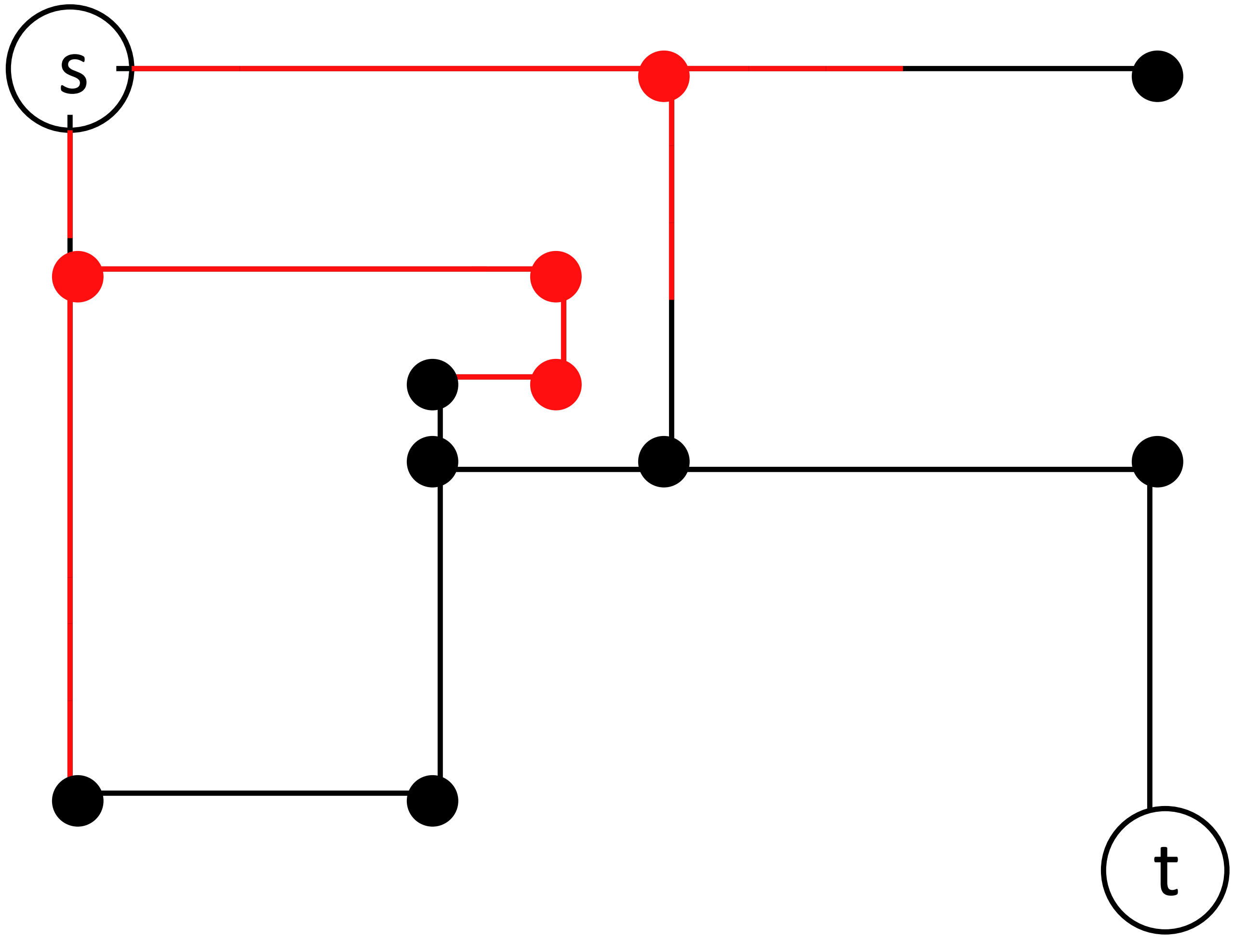
Ooze



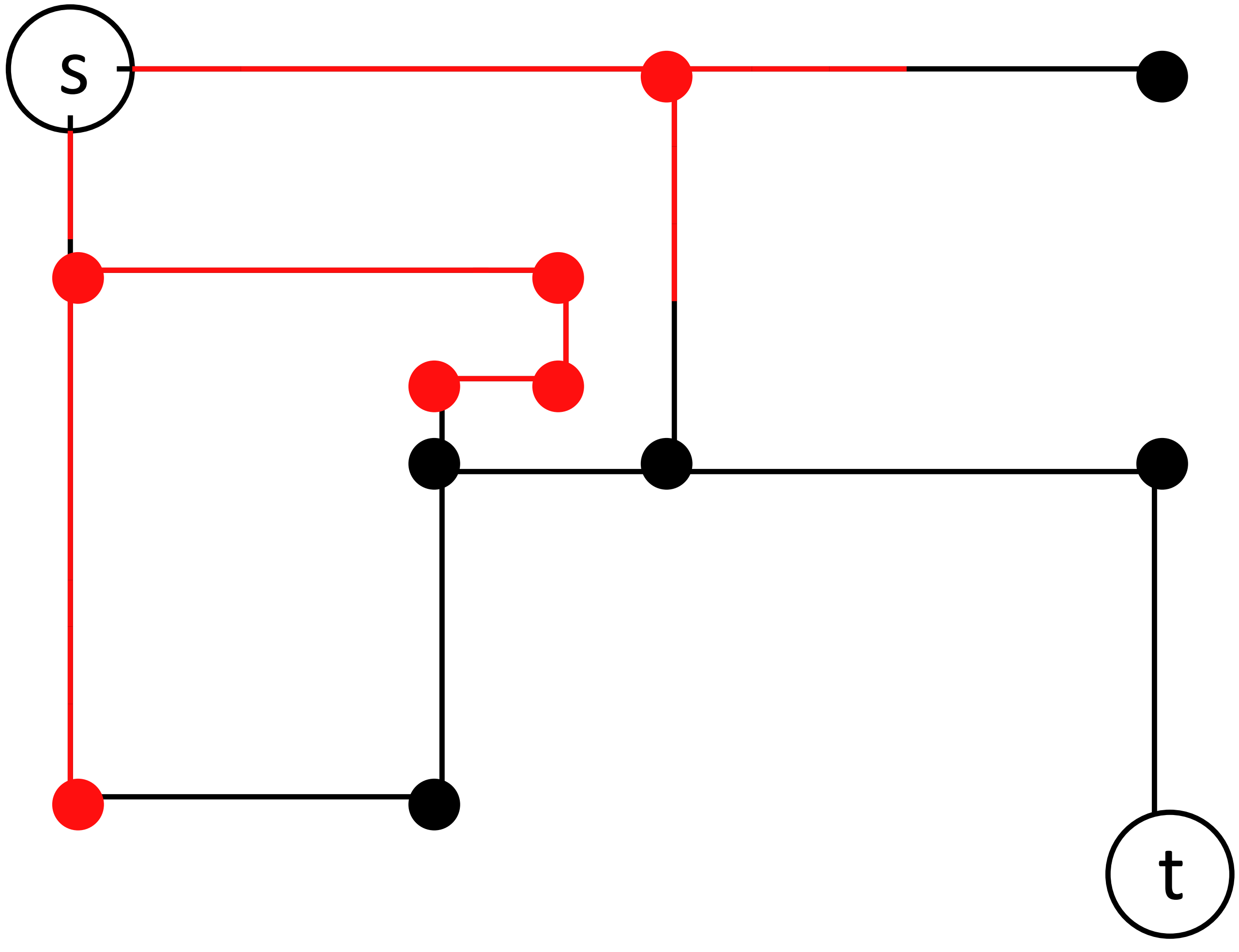
Ooze



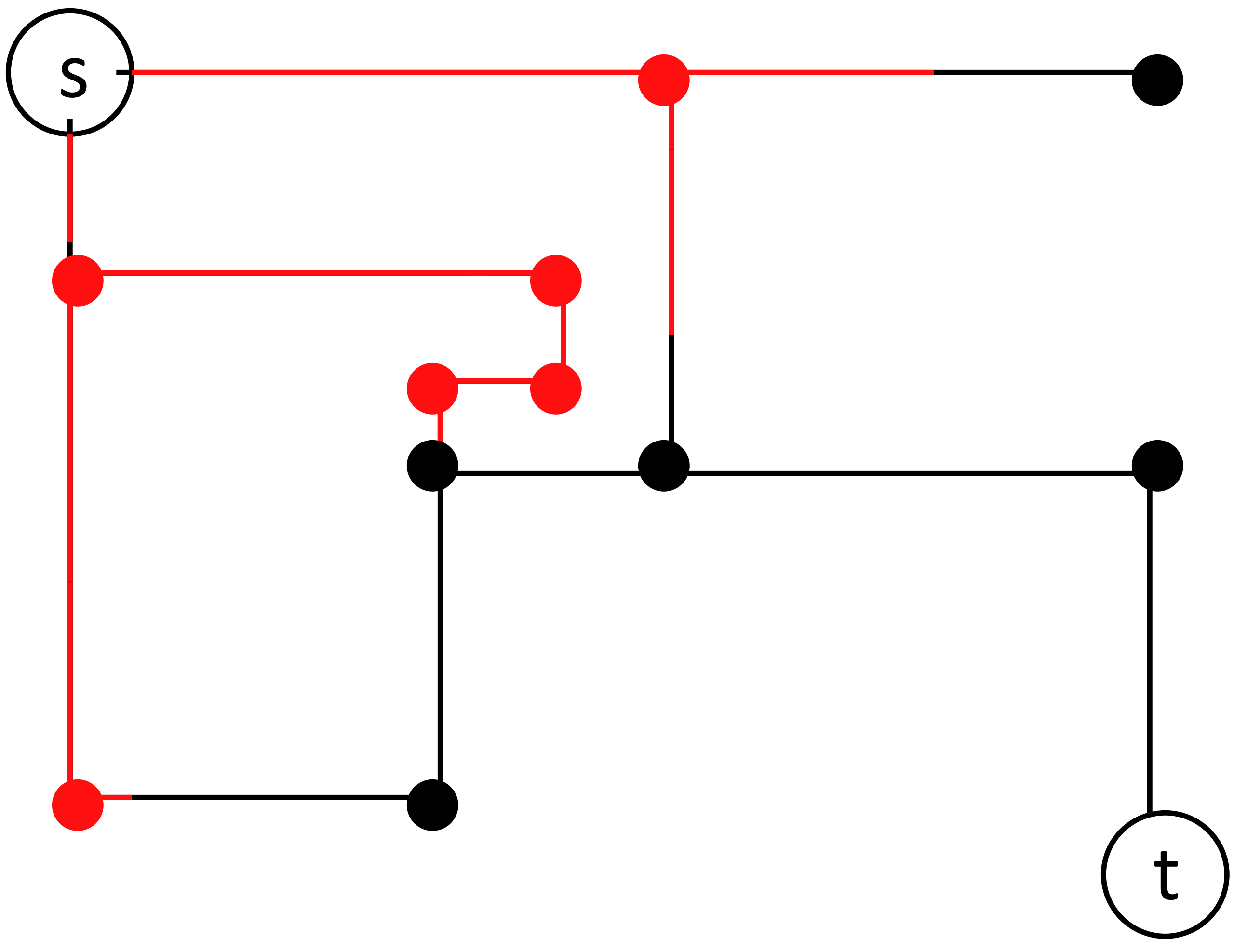
Ooze



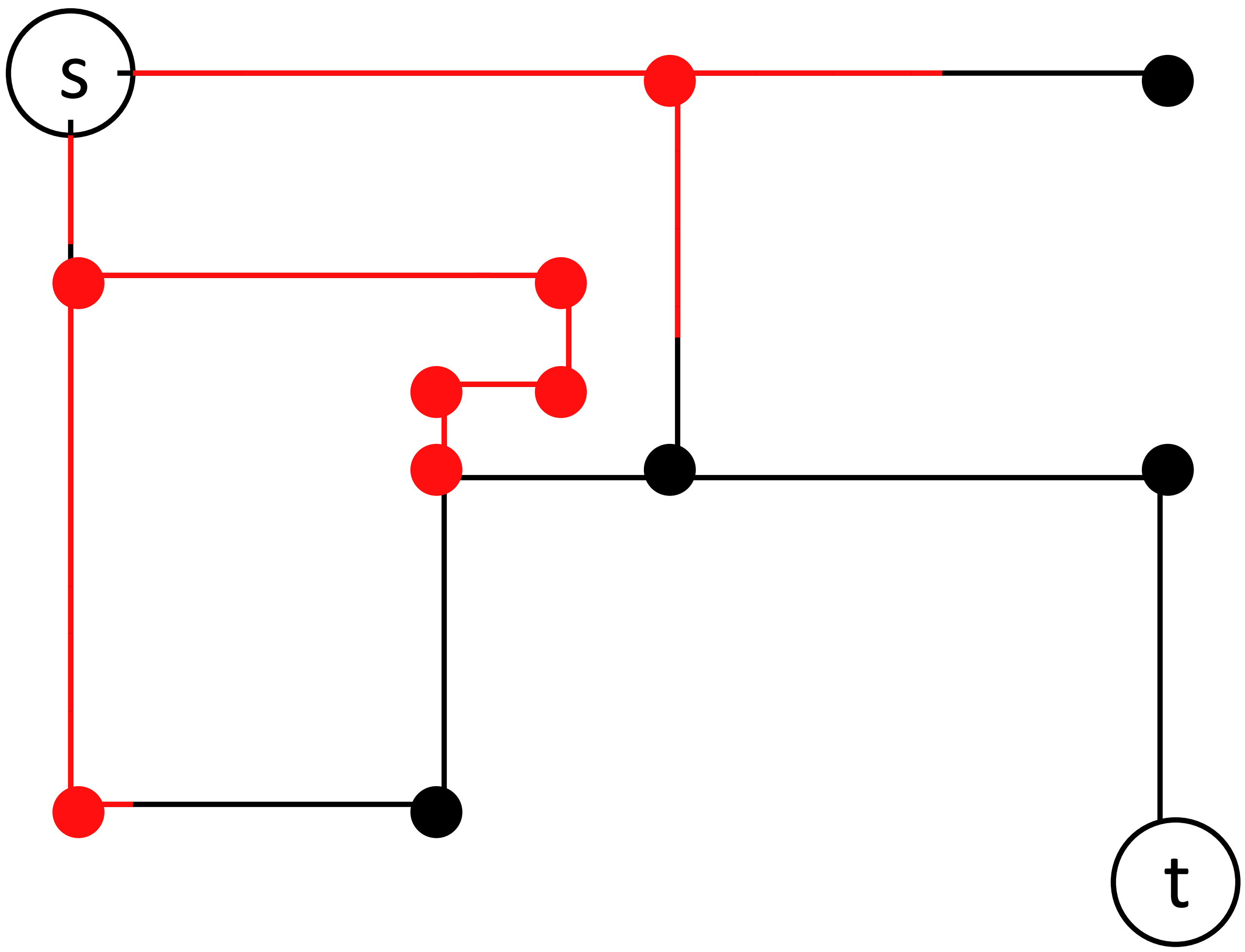
Ooze



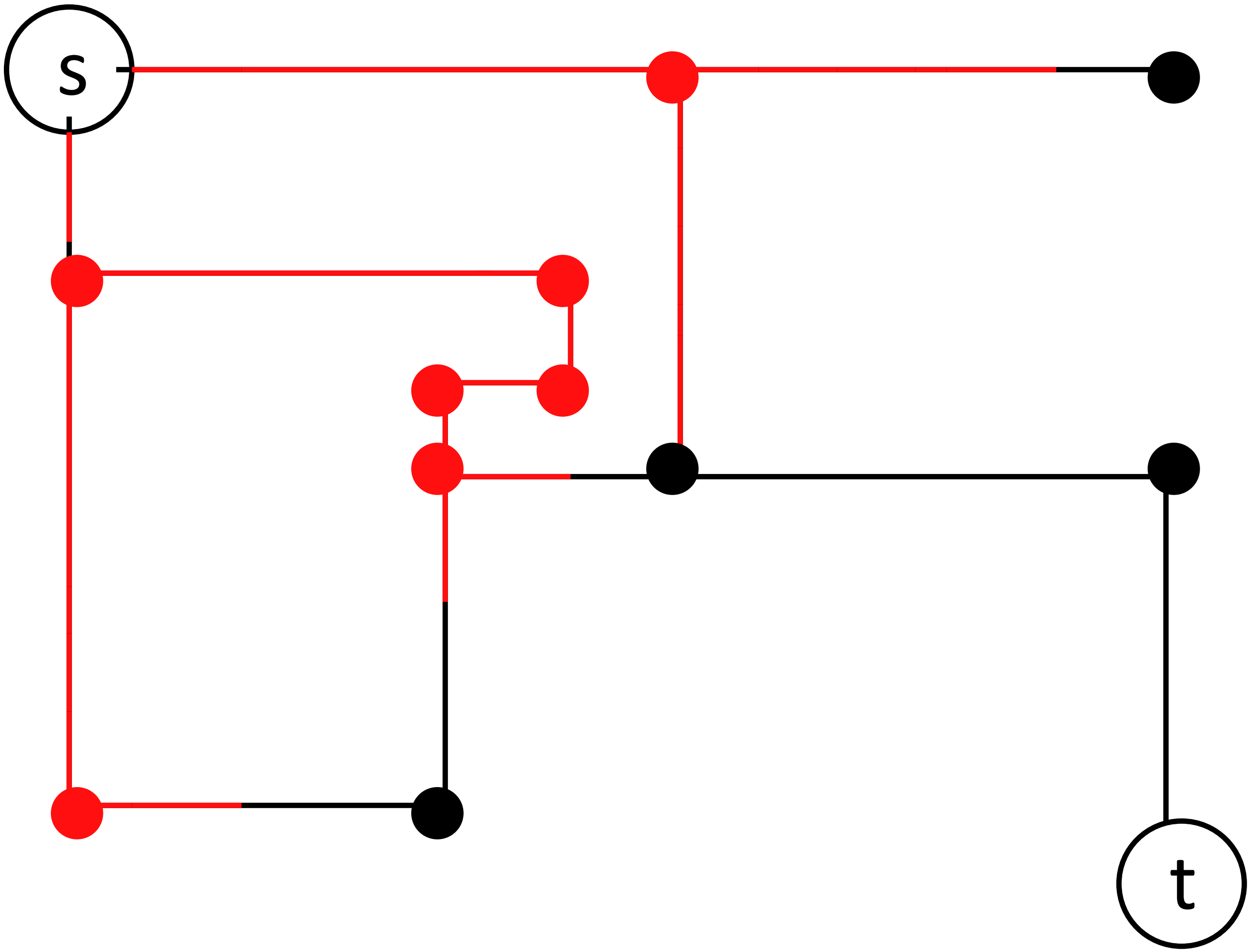
Ooze



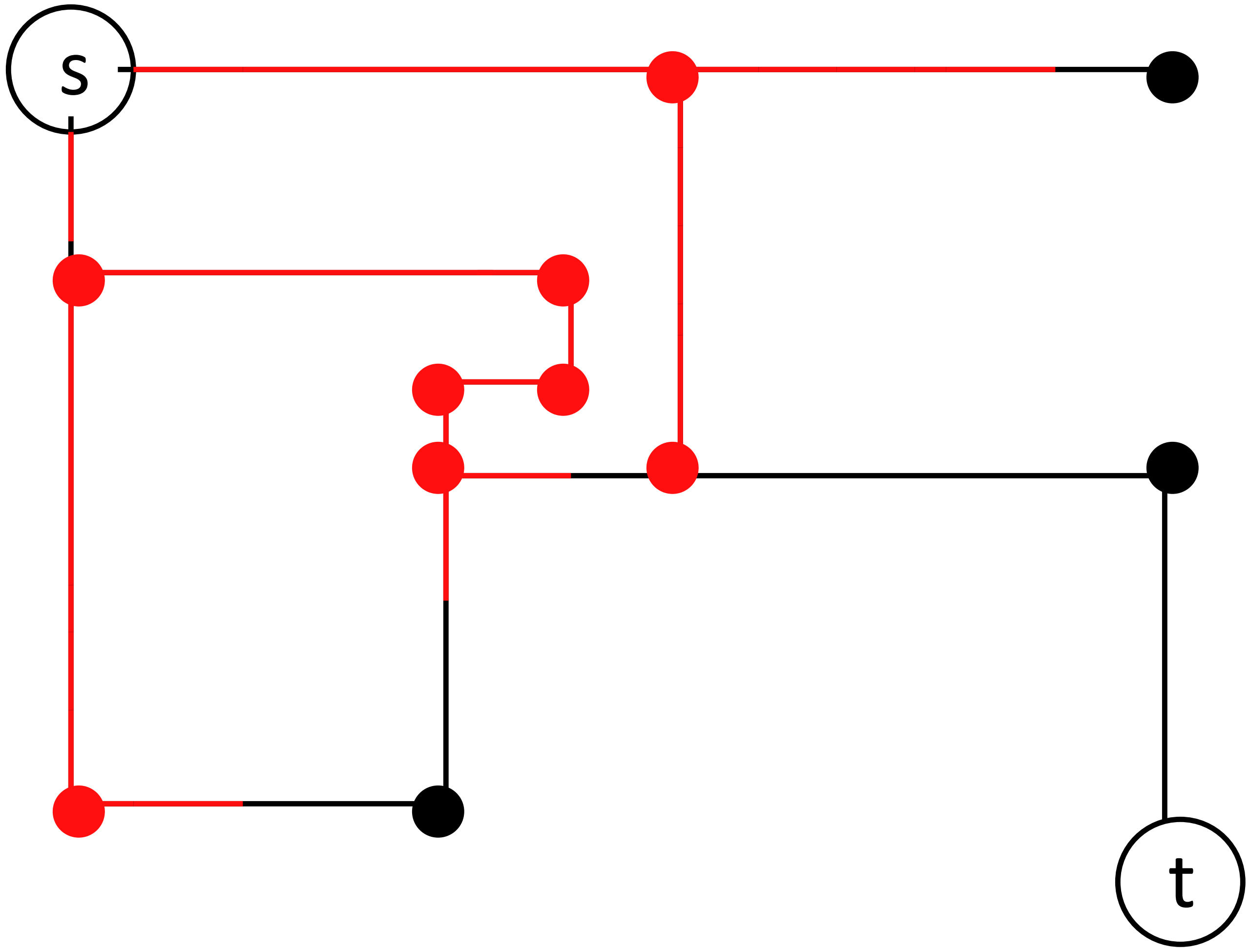
Ooze



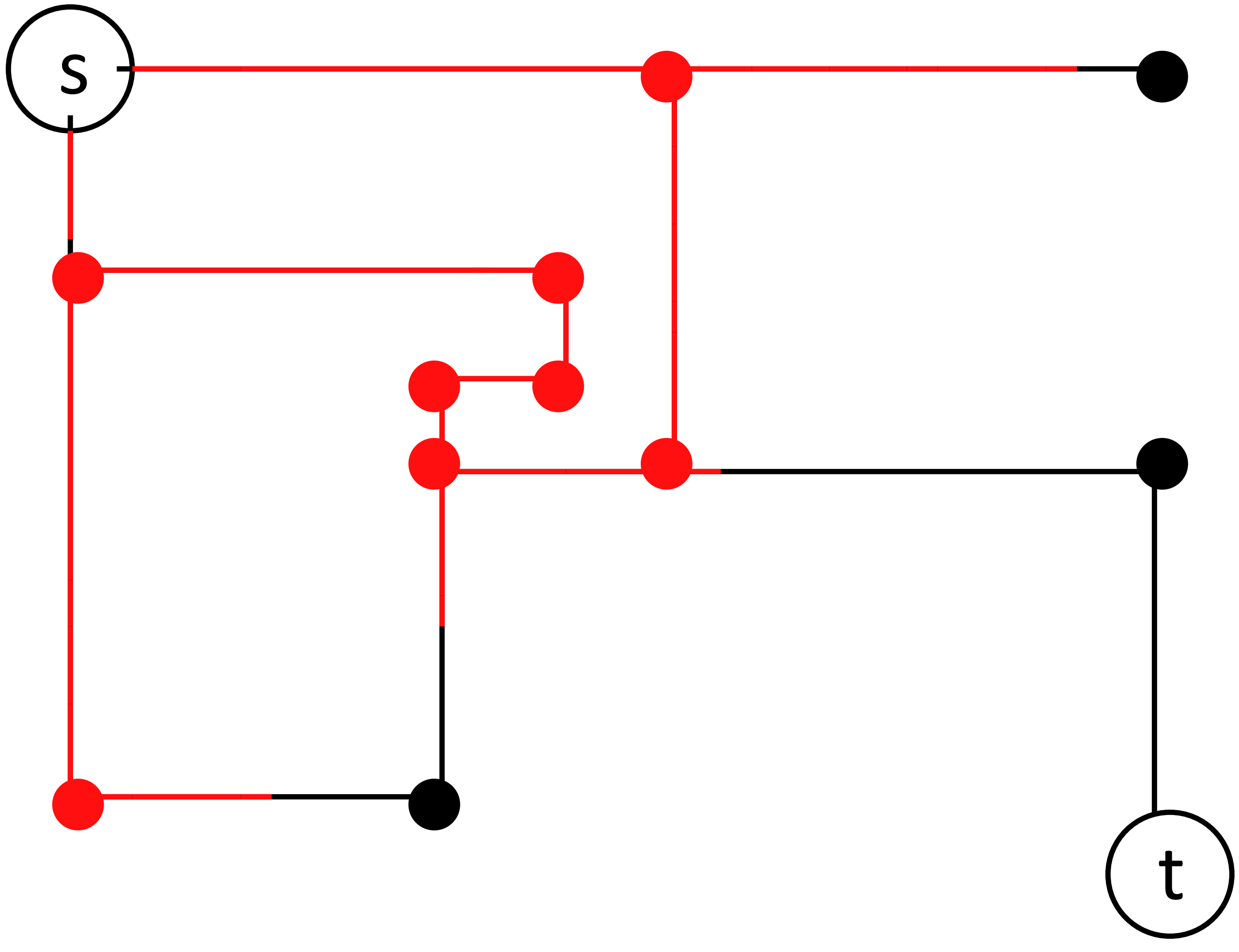
Ooze



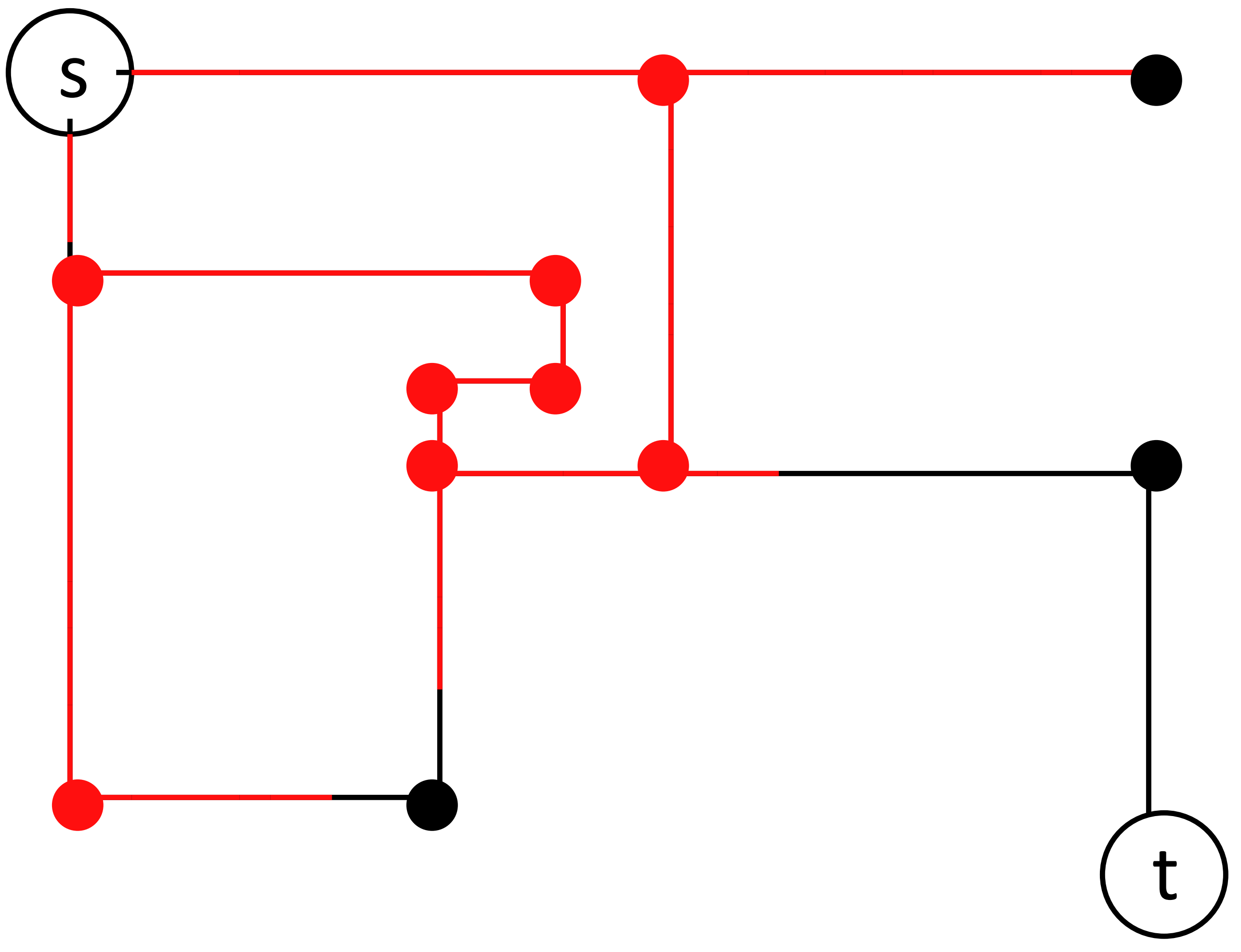
Ooze



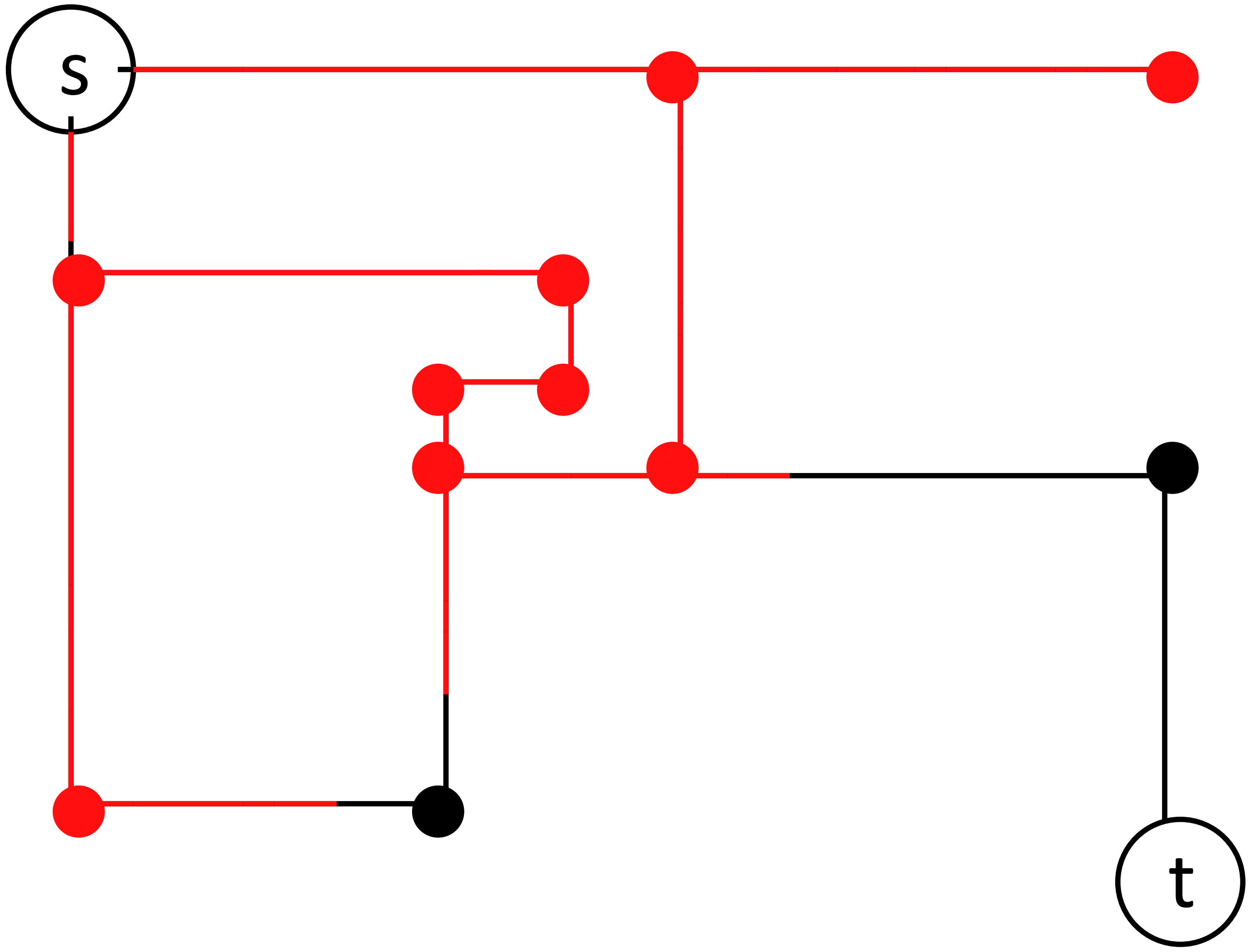
Ooze



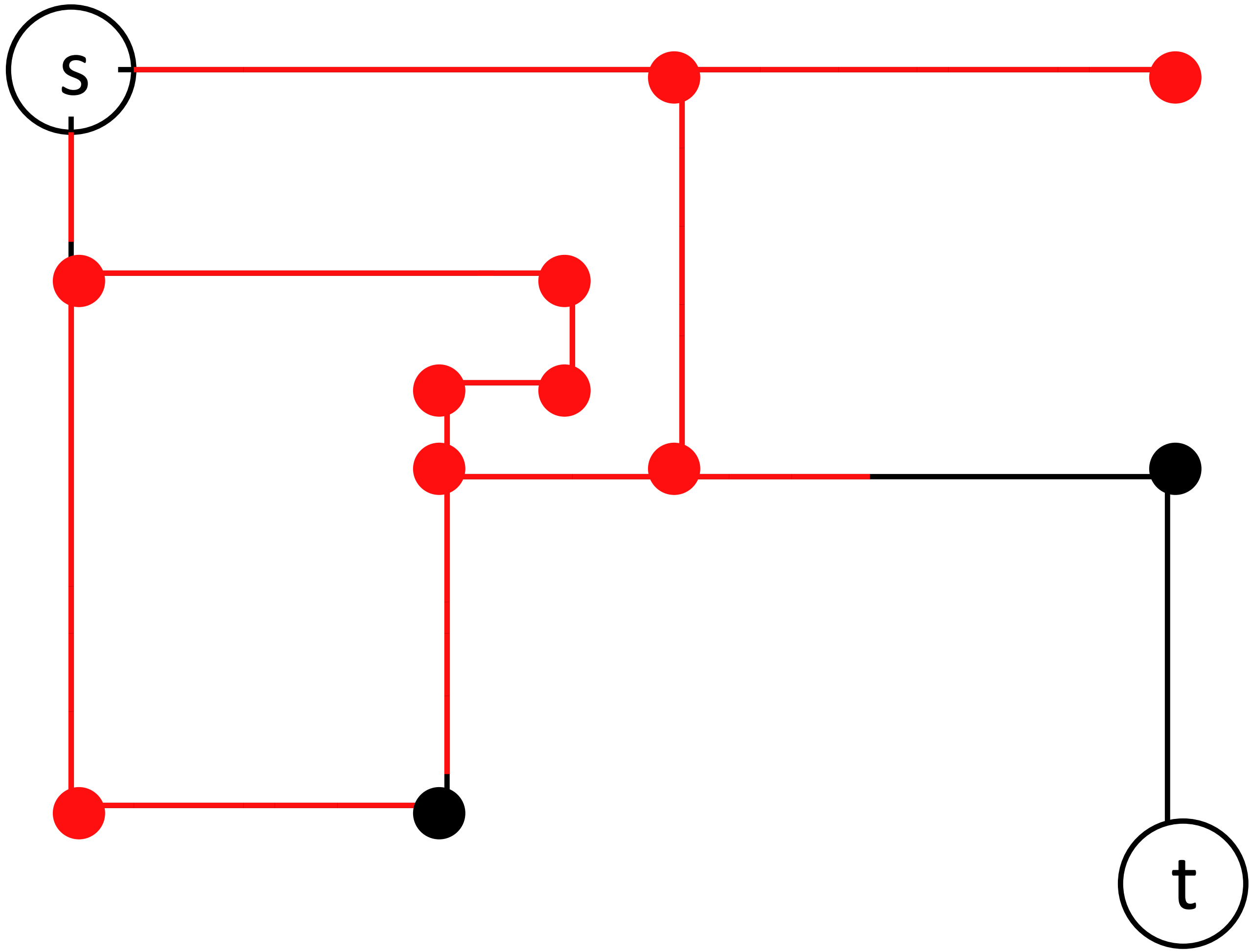
Ooze



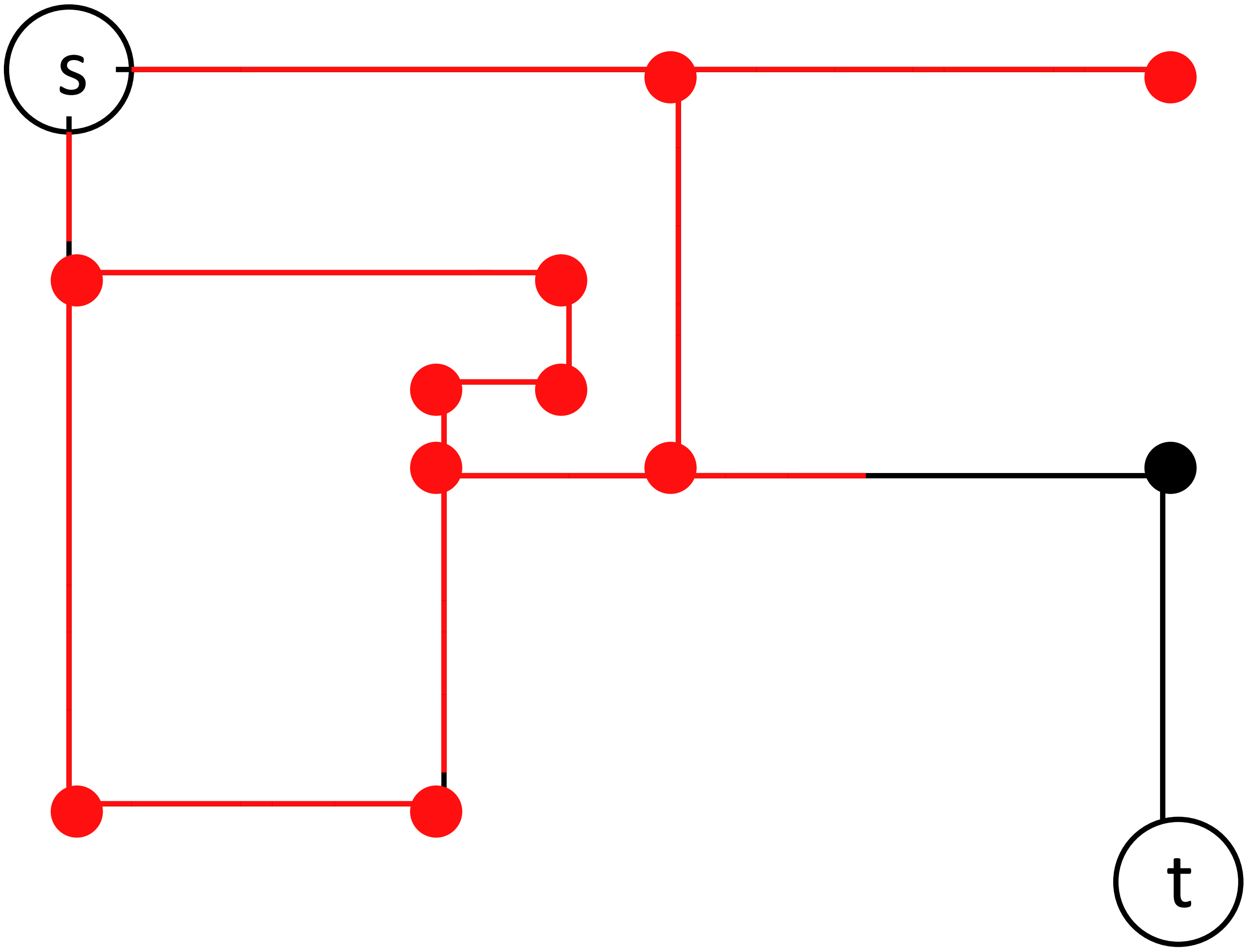
Ooze



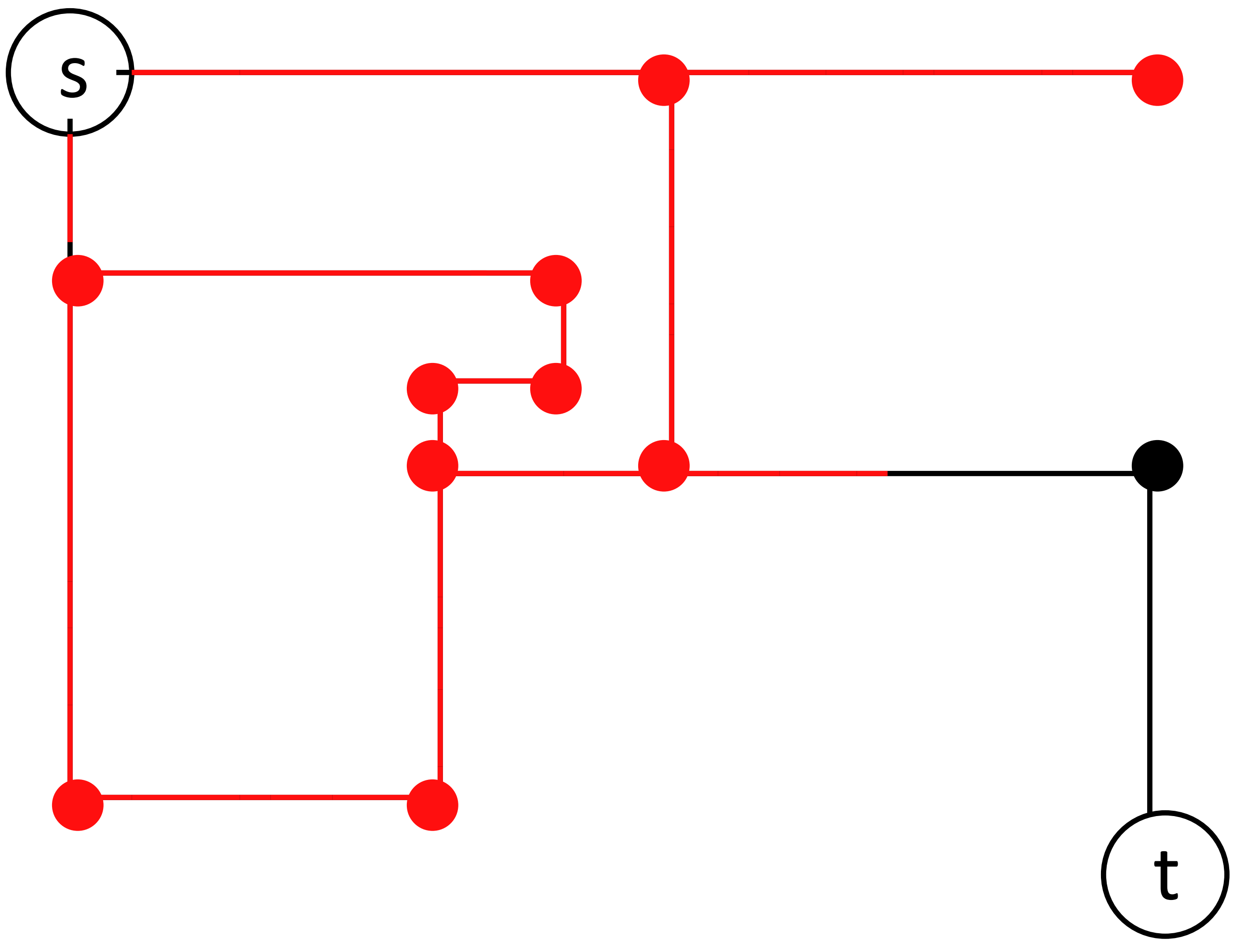
Ooze



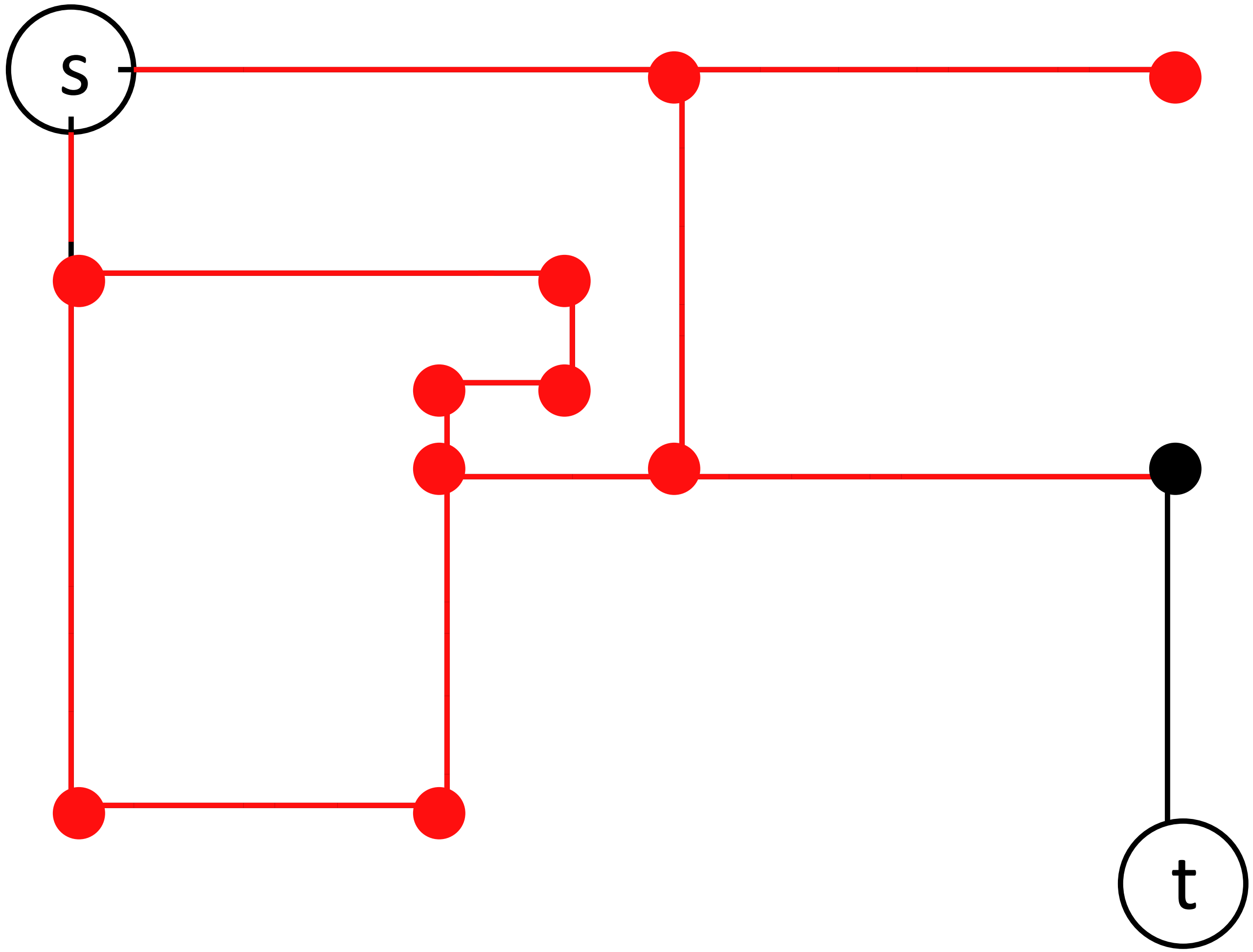
Ooze



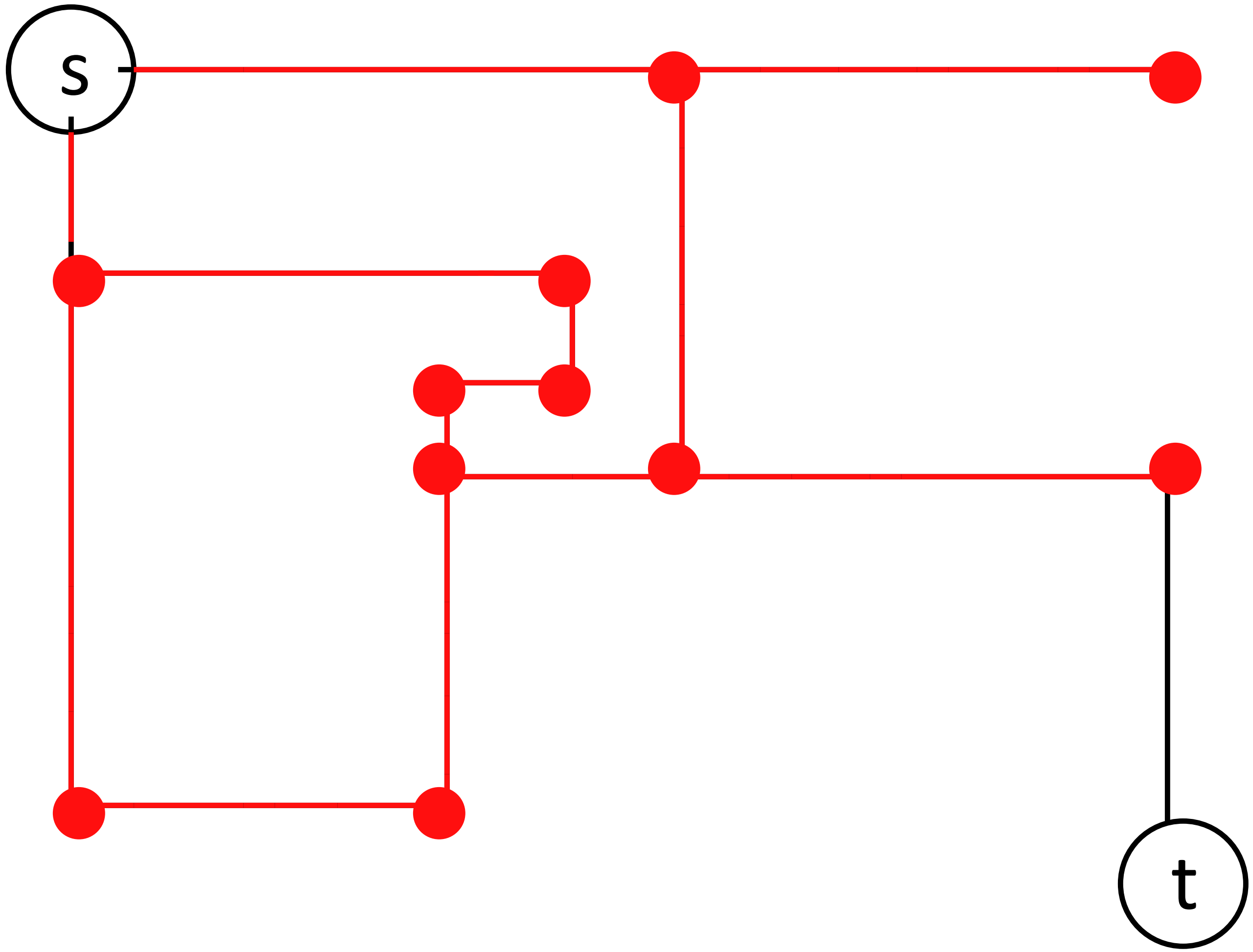
Ooze



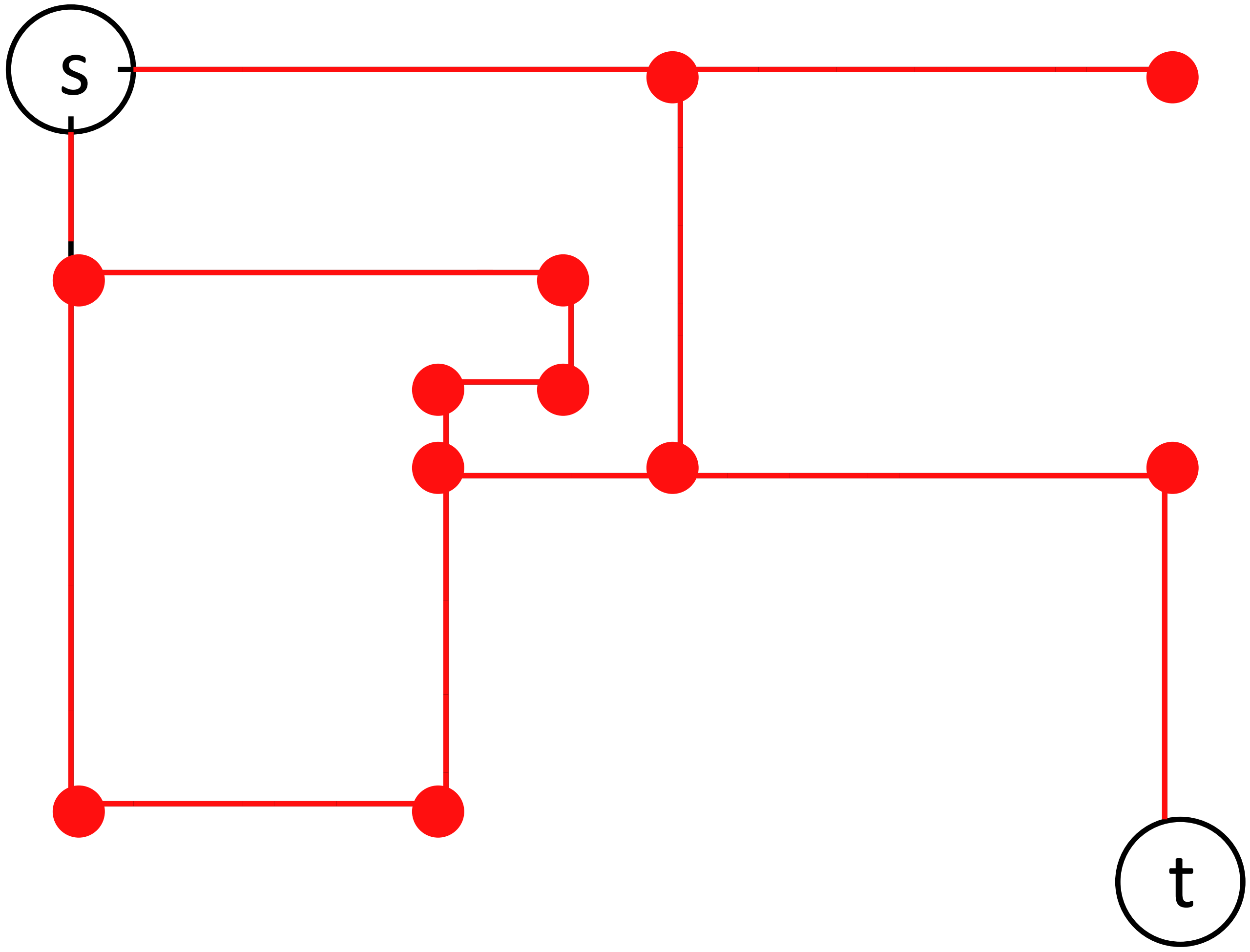
Ooze



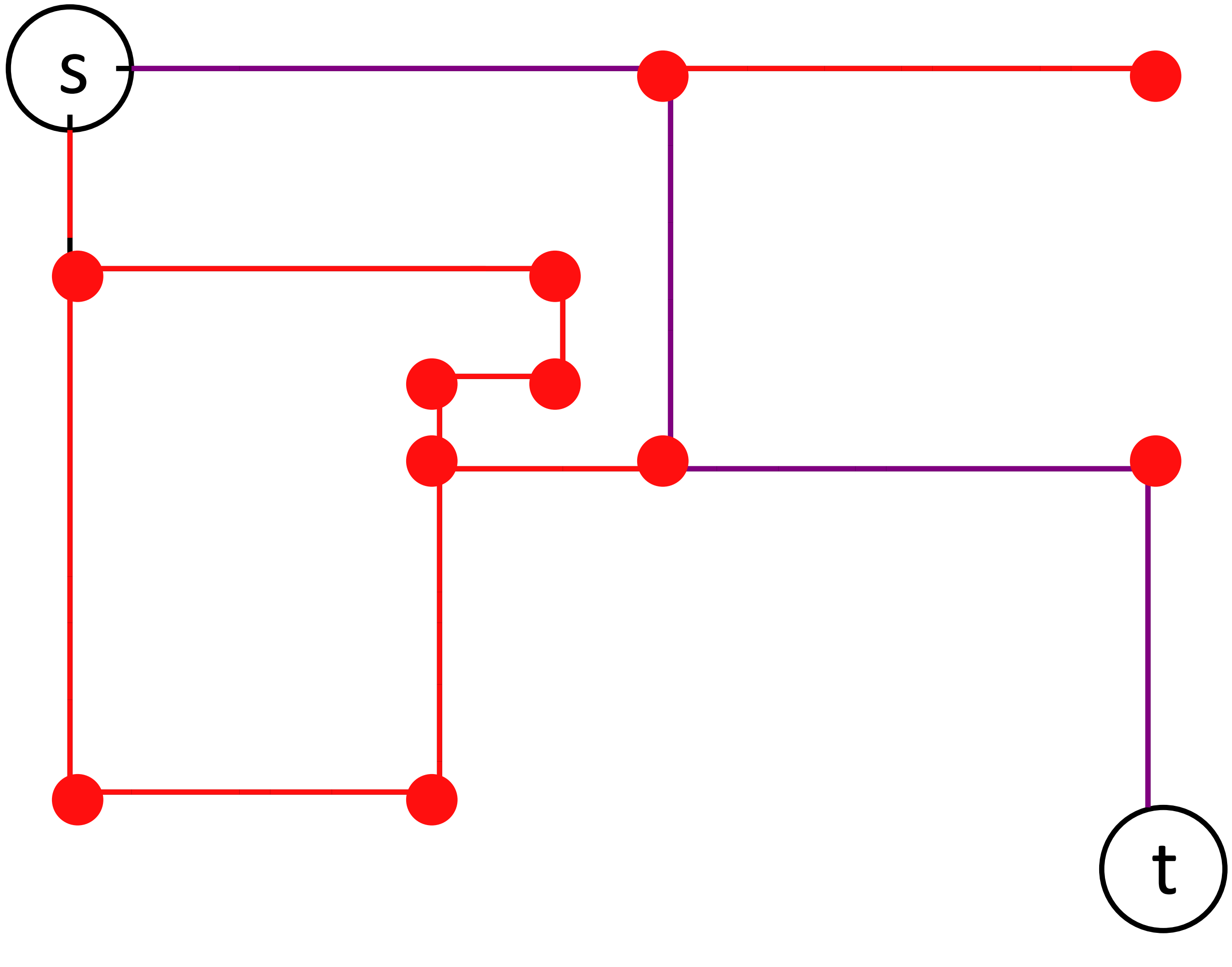
Ooze



Ooze



Ooze



Algorithm

- How do we make this an algorithm?

Algorithm

- How do we make this an algorithm?
- Simulate the above process keep track of when new vertices are discovered.

Algorithm

- How do we make this an algorithm?
- Simulate the above process keep track of when new vertices are discovered.
- If v is discovered at time t(v), the ooze from v will reach neighbor w at time $t(v)+\ell(v,w)$.

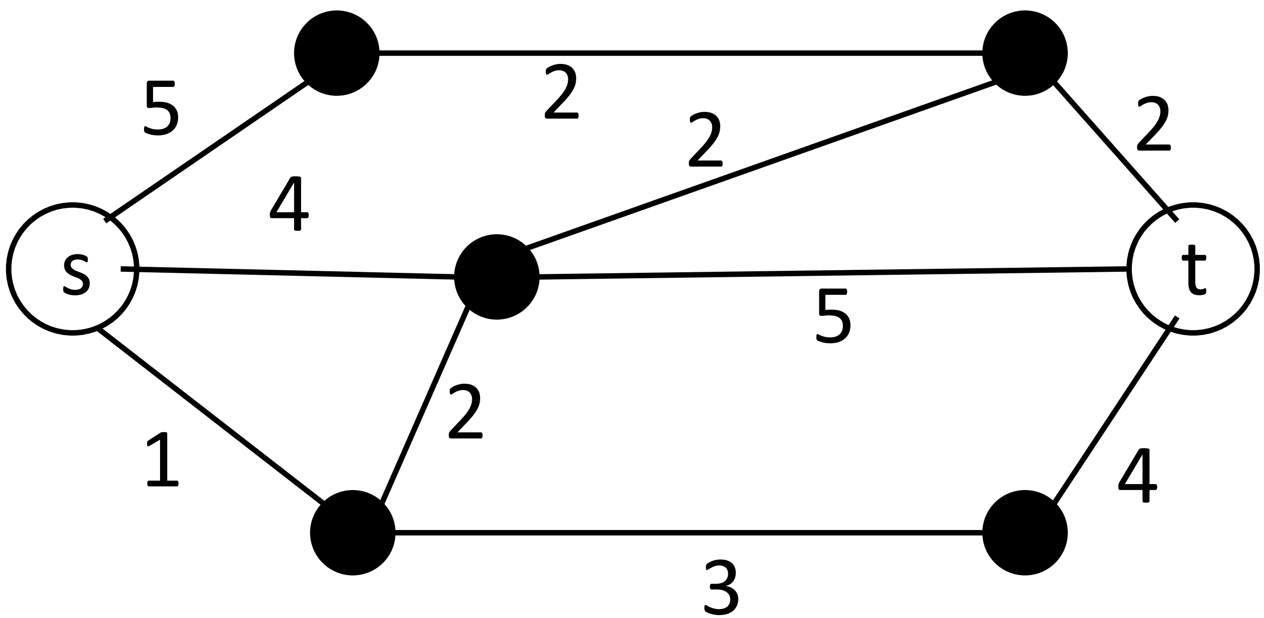
Algorithm

- How do we make this an algorithm?
- Simulate the above process keep track of when new vertices are discovered.
- If v is discovered at time t(v), the ooze from v will reach neighbor w at time $t(v)+\ell(v,w)$.
- Next vertex to be discovered is one with minimal $t(v)+ \ell(v,w)$.

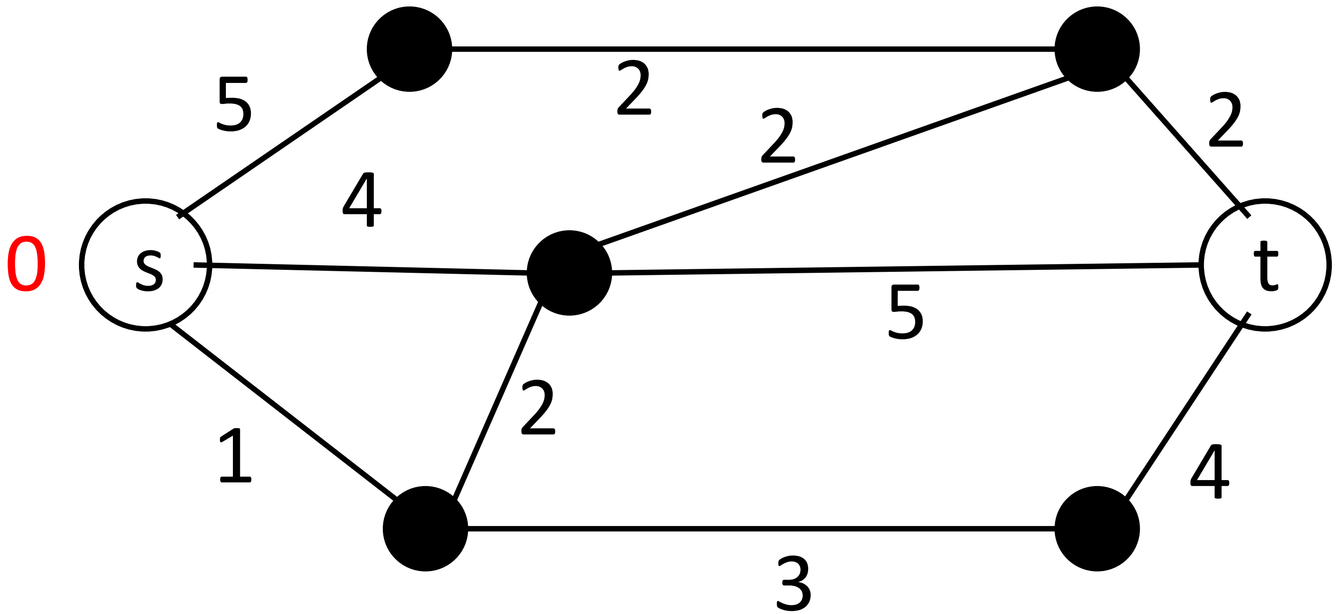
Algorithm

```
Distances(G,s,ℓ)
  dist(s) ← 0
  While(not all distances found)
    Find minimum over (v,w) ∈ E
      with v discovered w not
      of dist(v)+ℓ(v,w)
    dist(w) ← dist(v)+ℓ(v,w)
    prev(w) ← v
```

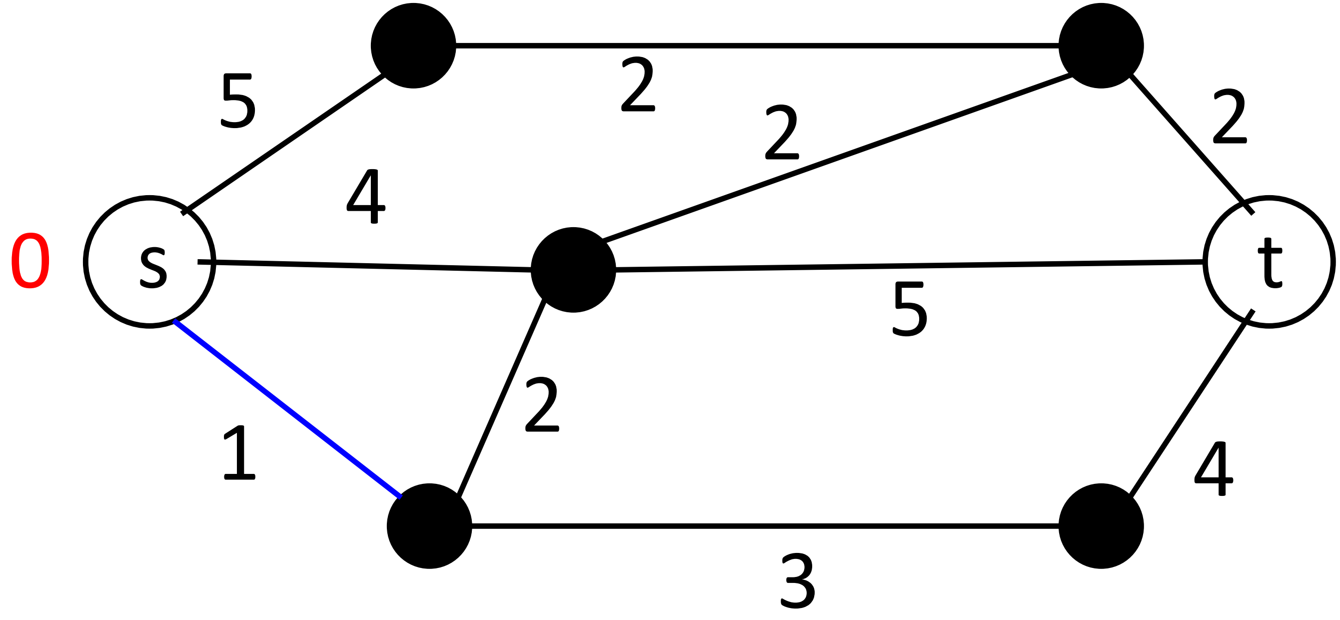
Example



Example

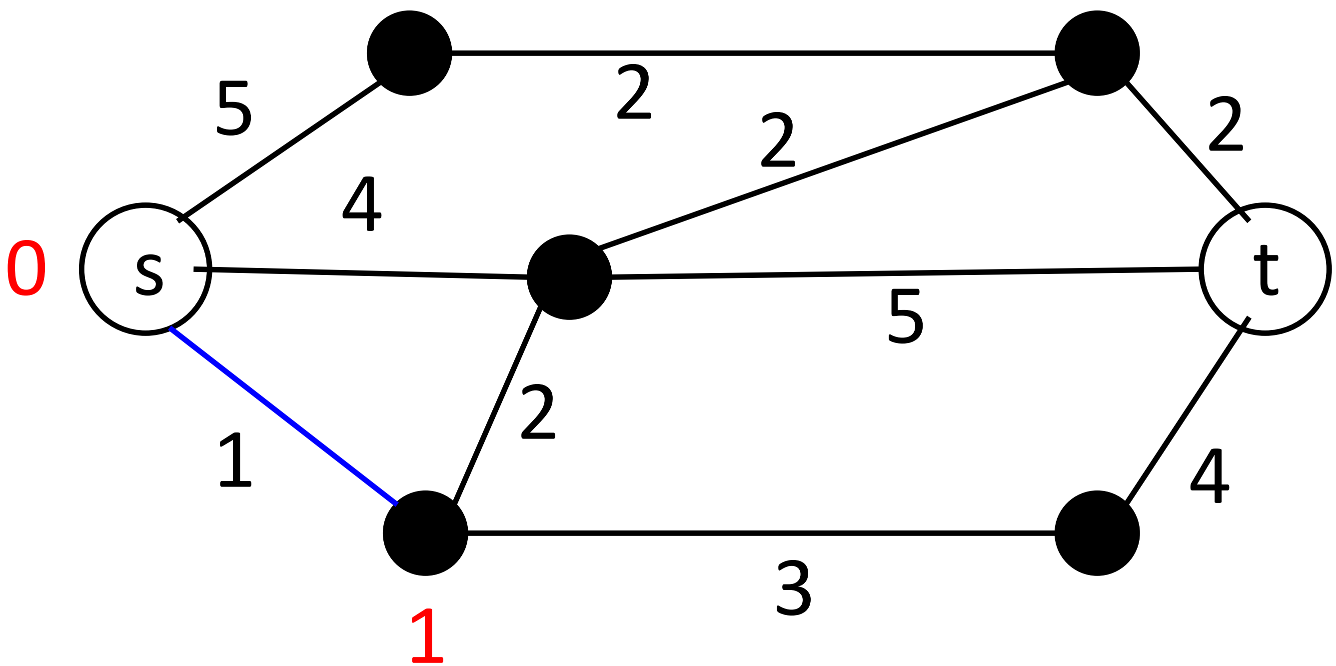


Example

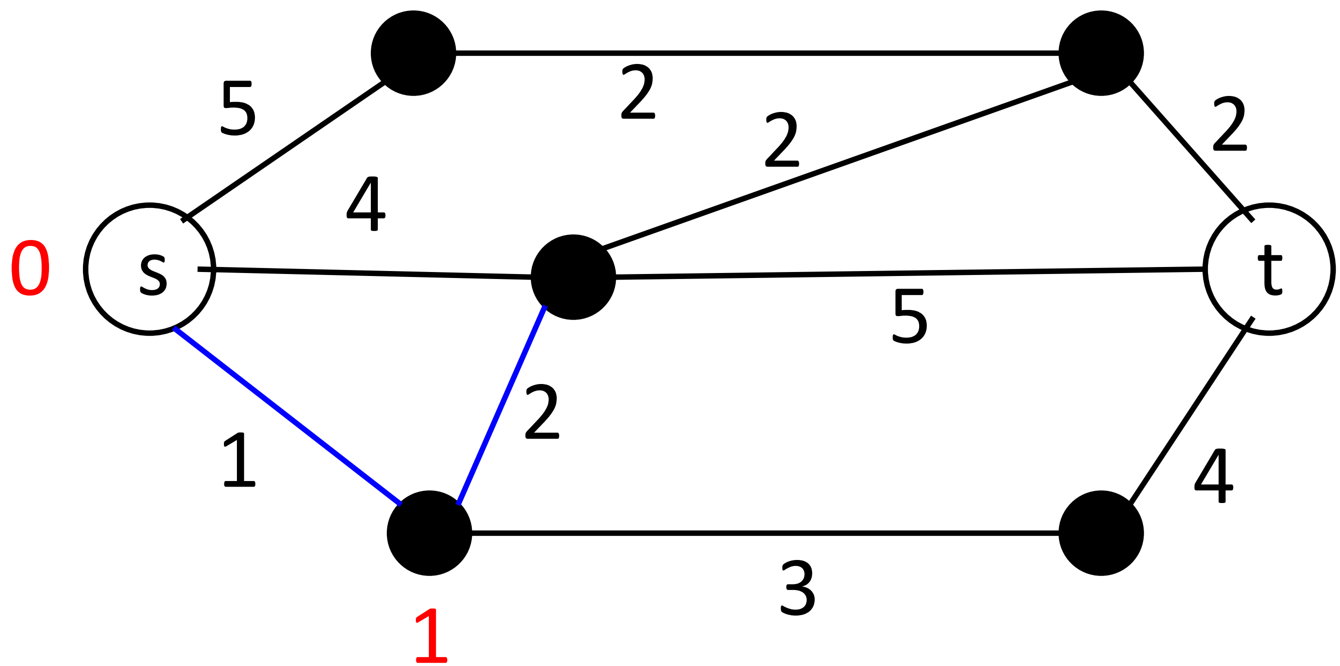


0+1=1

Example

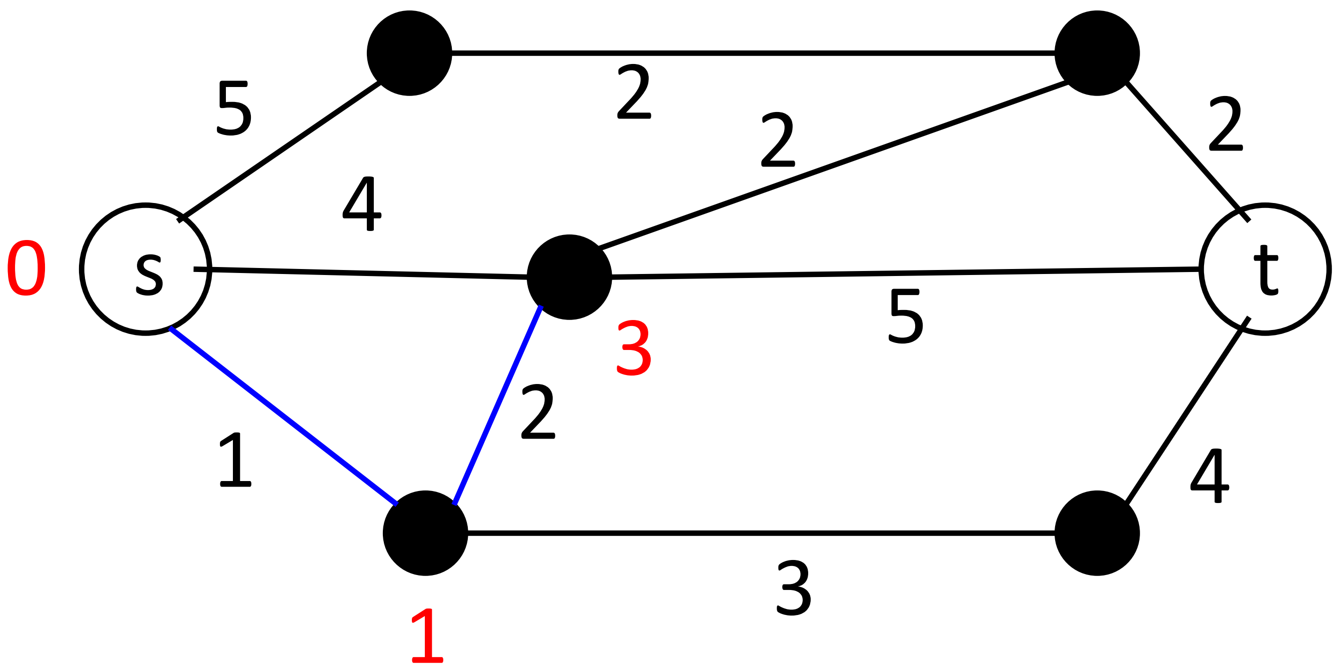


Example

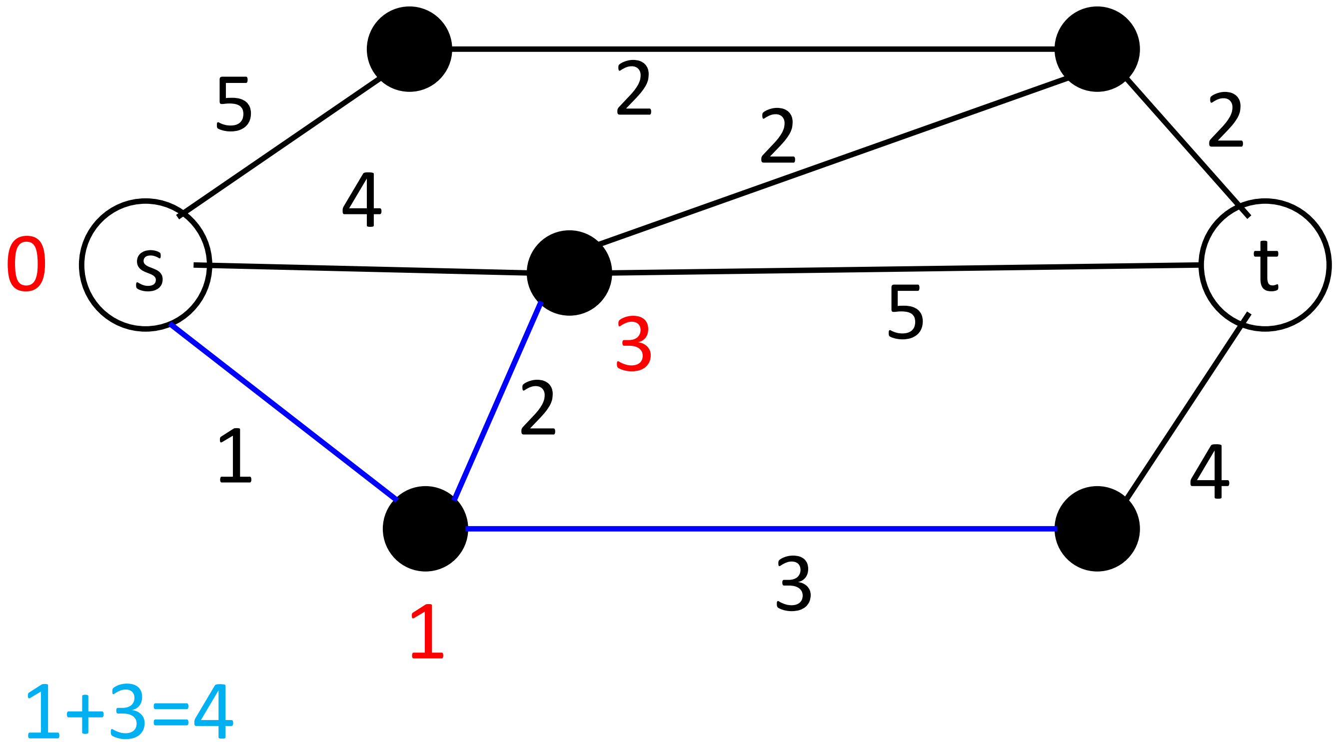


1+2=3

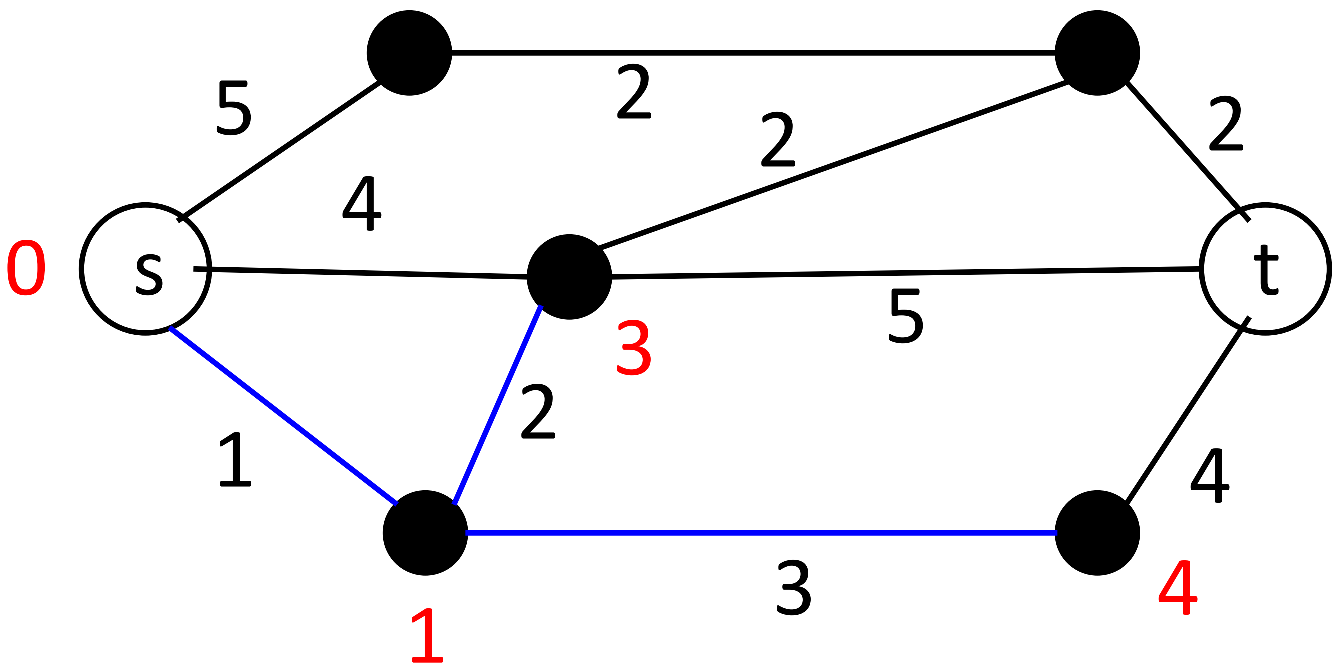
Example



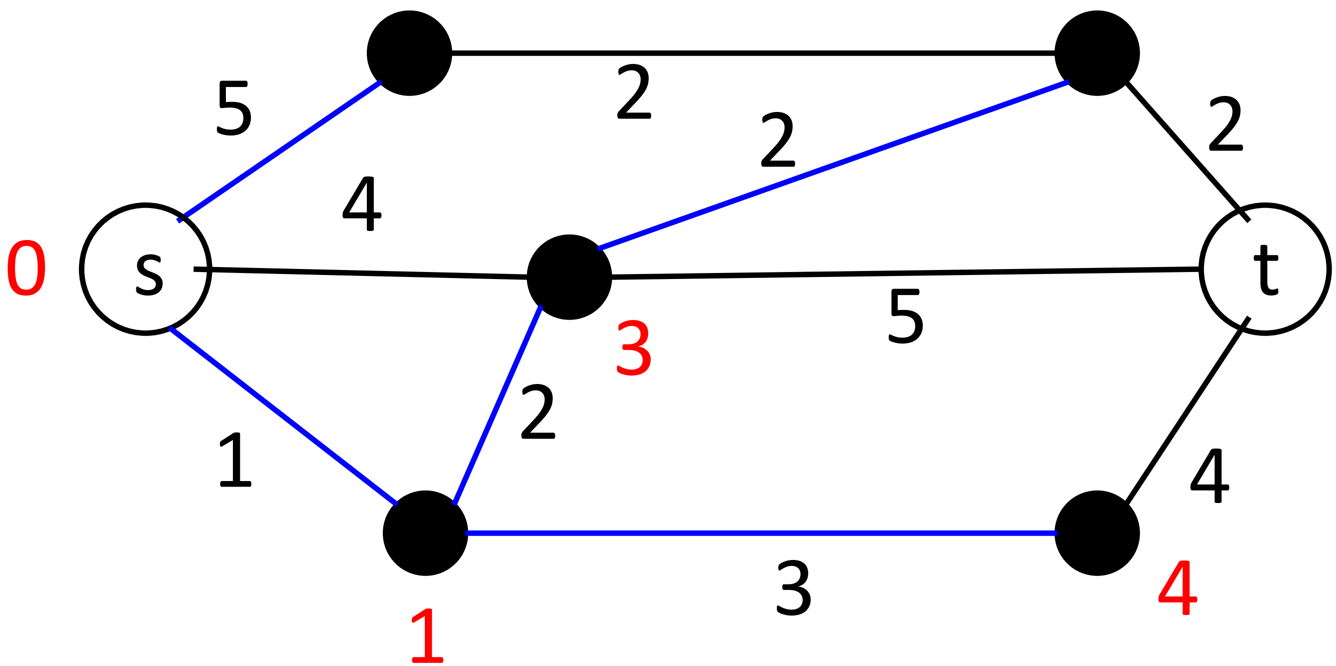
Example



Example



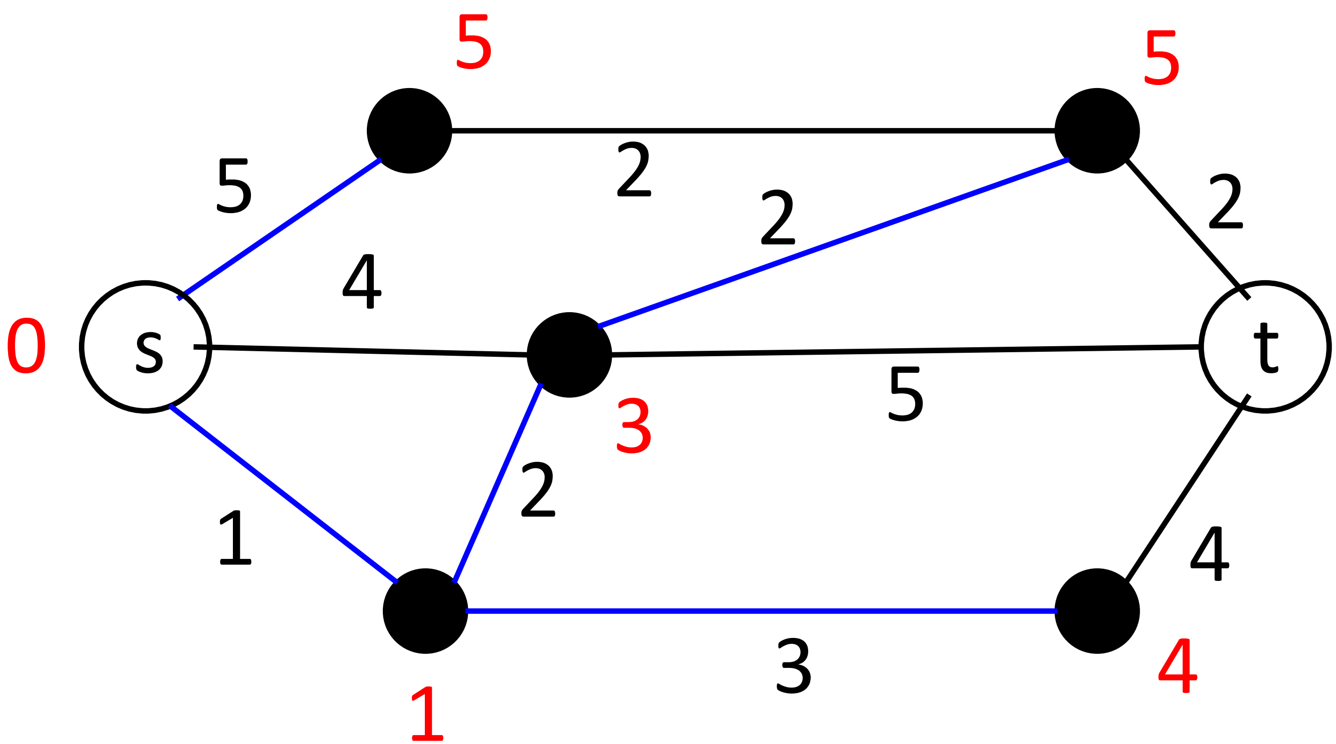
Example



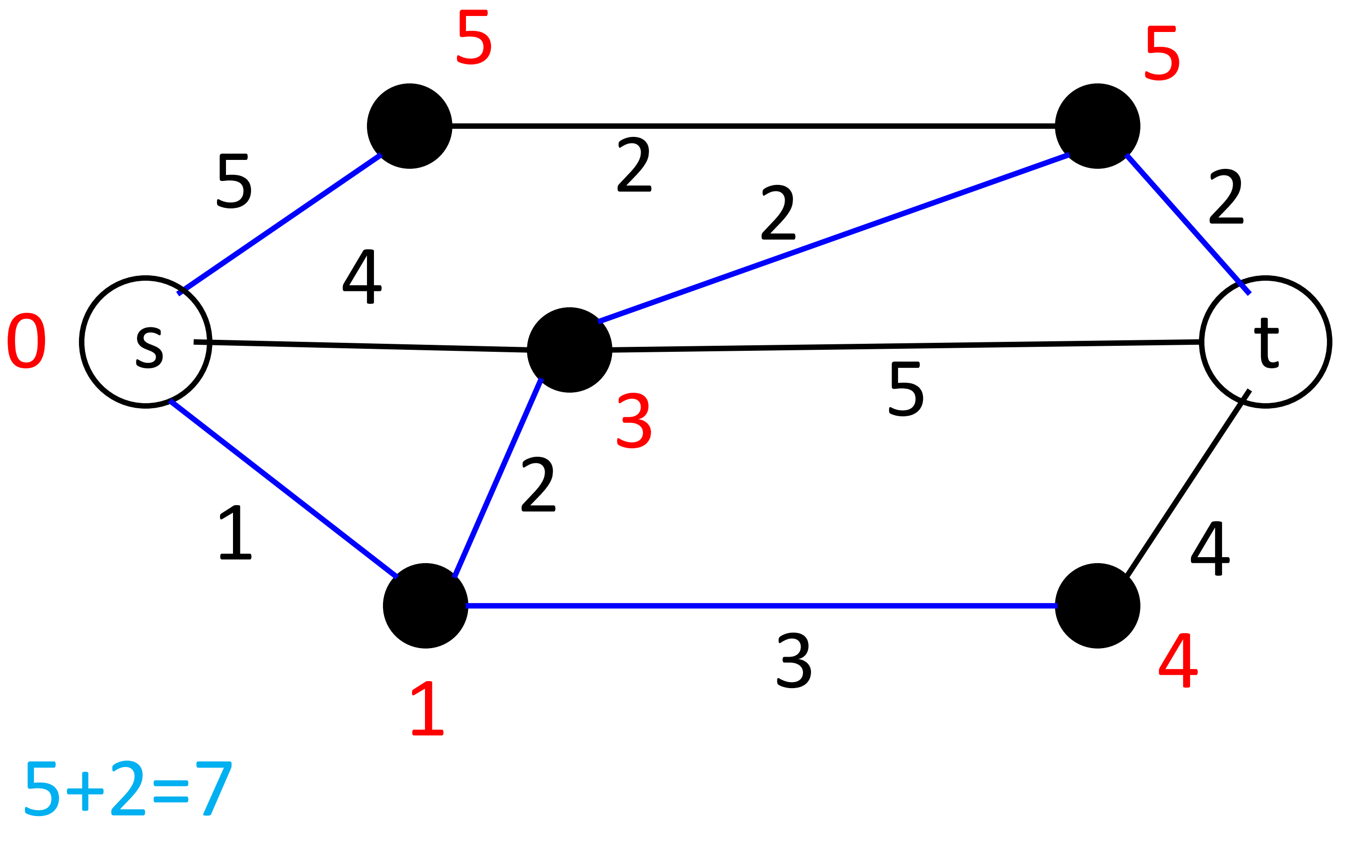
2+3=5

0+5=5

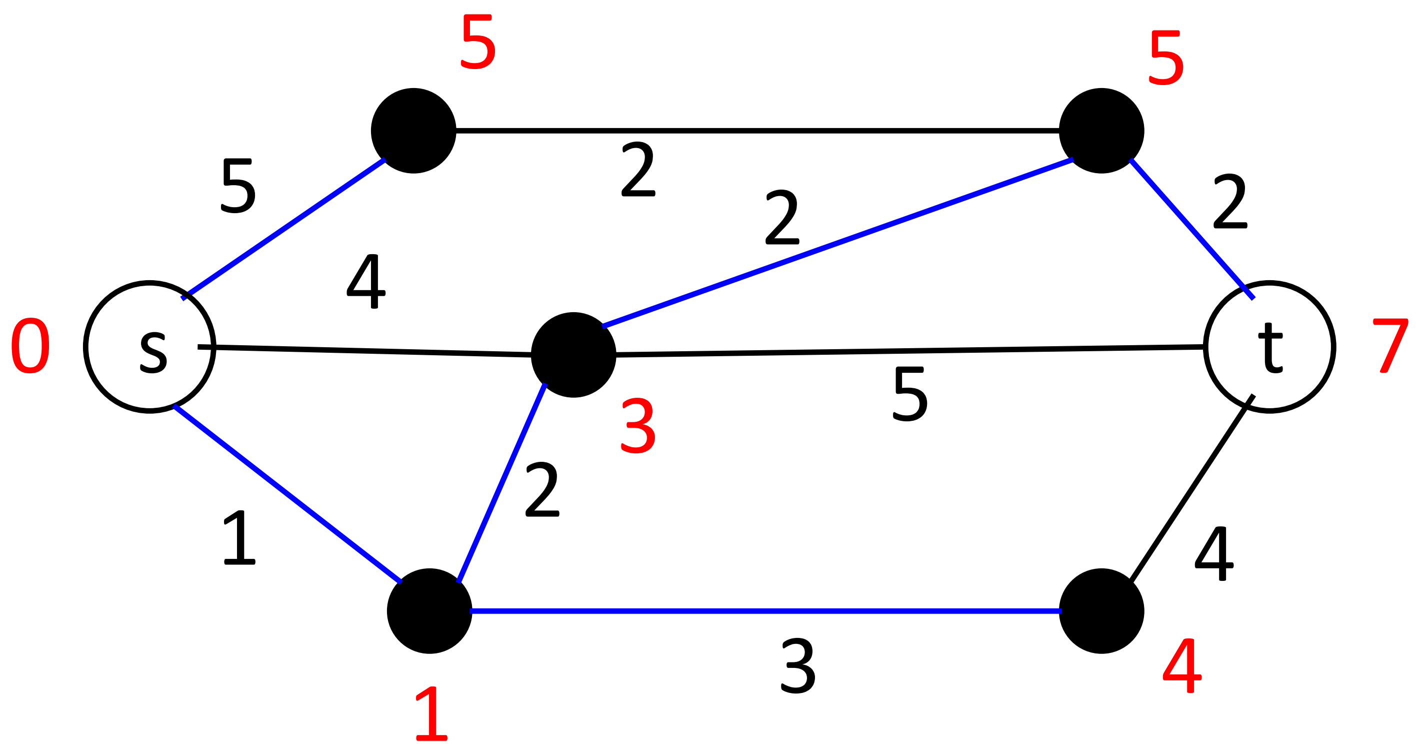
Example



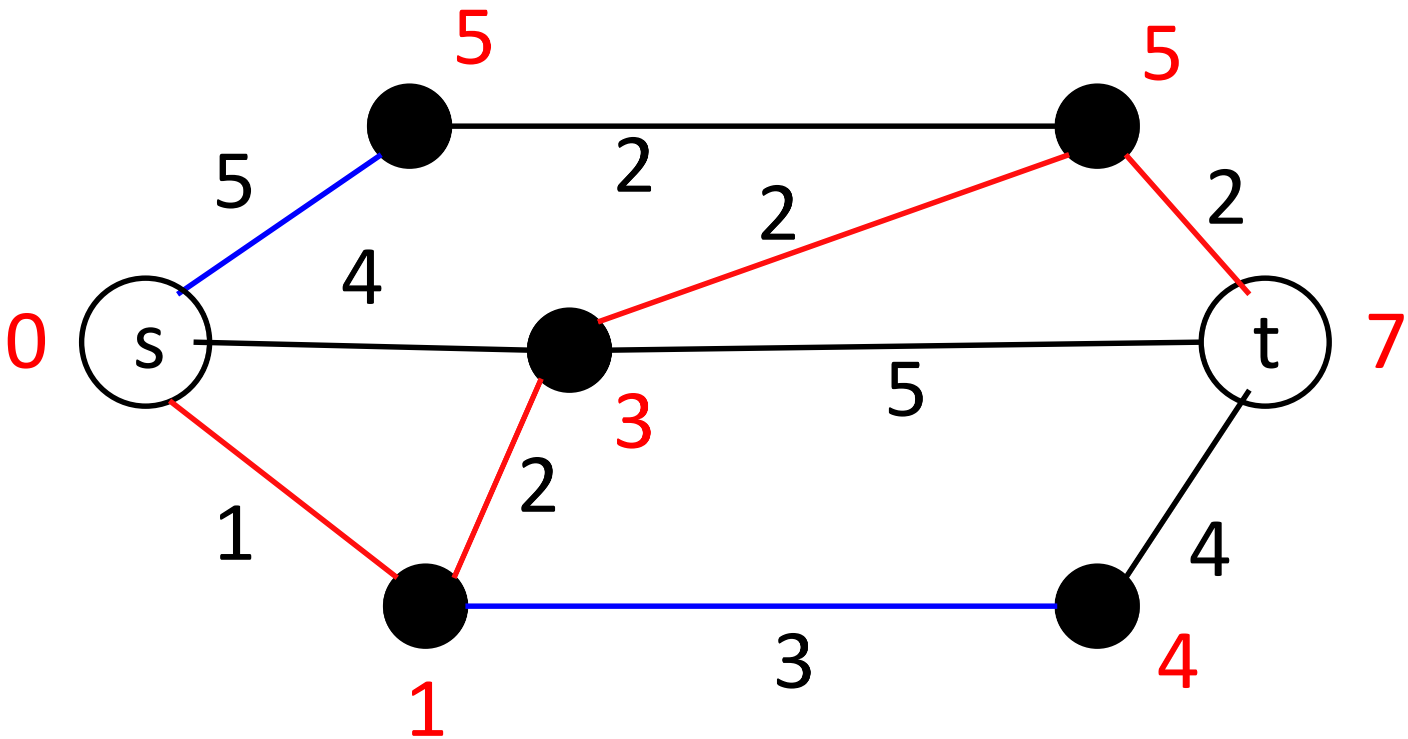
Example



Example



Example



Why does this work?

Claim: Whenever the algorithm assigns a distance to a vertex v that is the length of the shortest path from s to v.

Why does this work?

Claim: Whenever the algorithm assigns a distance to a vertex v that is the length of the shortest path from s to v.

Proof by Induction:

- dist(s) = 0 [the empty path has length 0]

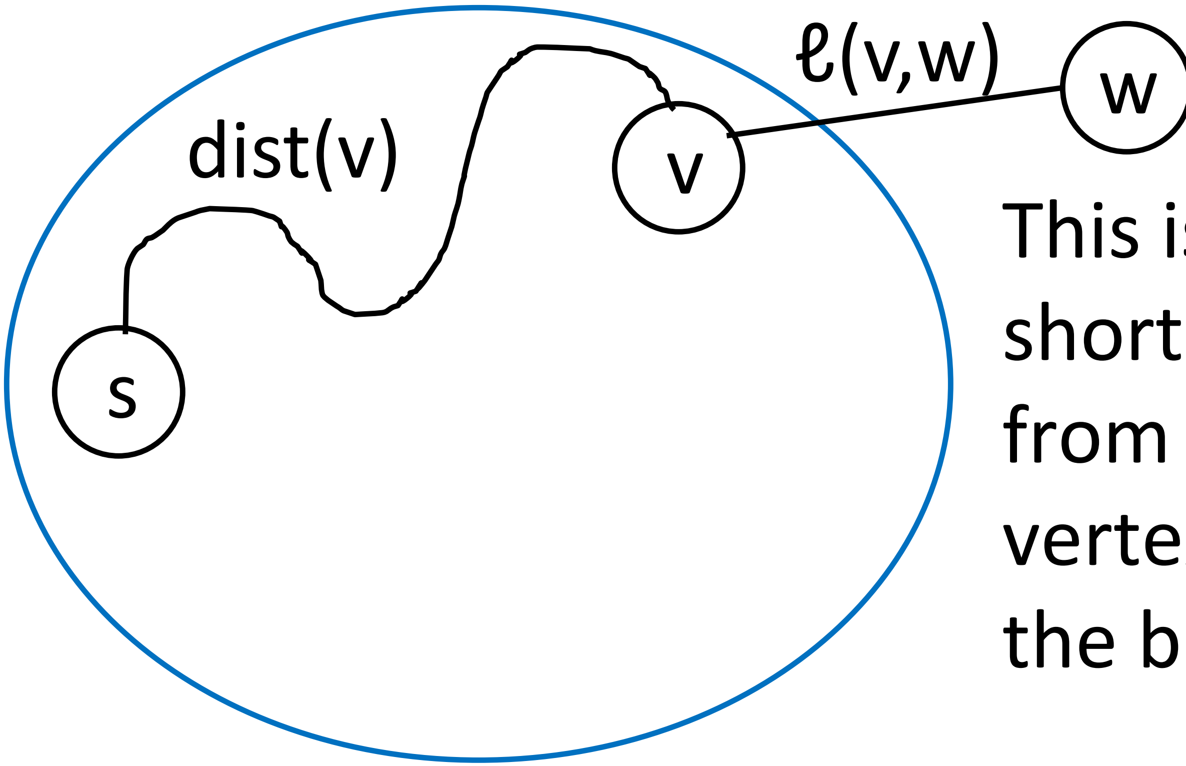
Why does this work?

<u>Claim:</u> Whenever the algorithm assigns a distance to a vertex v that is the length of the shortest path from s to v.

<u>Proof by Induction:</u>

- dist(s) = 0 [the empty path has length 0]
- When assigning distance to w, assume that all previously assigned distances are correct.

Inductive Step



This is the shortest path from s to *any* vertex outside the bubble.

Correctly Assigned Distances

Question: Runtime

What is the runtime of this algorithm?

A) $O(|V|+|E|)$
B) $O(|V|\log|V|+|E|)$
C) $O(|V||E|)$
D) $O(|E|^2)$

```
Distances(G,s,ℓ)
  dist(s) ← 0
  While(not all distances found)
    Find minimum over (v,w) ∈ E
      with v discovered w not
      of dist(v)+ℓ(v,w)
    dist(w) ← dist(v)+ℓ(v,w)
    prev(w) ← v
```

Question: Runtime

What is the runtime of this algorithm?

A) $O(|V|+|E|)$
B) $O(|V|\log|V|+|E|)$
C) $O(|V||E|)$
D) $O(|E|^2)$

```
Distances(G,s,ℓ)
  dist(s) ← 0
  While(not all distances found)
    Find minimum over (v,w) ∈ E
      with v discovered w not
      of dist(v)+ℓ(v,w)
    dist(w) ← dist(v)+ℓ(v,w)
    prev(w) ← v
```

Runtime

```
Distances(G,s,ℓ)
  dist(s) ← 0
  While(not all distances found)
    Find minimum over (v,w) ∈ E
      with v discovered w not
      of dist(v)+ℓ(v,w)
    dist(w) ← dist(v)+ℓ(v,w)
    prev(w) ← v
```

Runtime

```
Distances(G,s,ℓ)
  dist(s) ← 0
  While(not all distances found)
    Find minimum over (v,w) ∈ E
      with v discovered w not
      of dist(v)+ℓ(v,w)
    dist(w) ← dist(v)+ℓ(v,w)
    prev(w) ← v
```

O(|V|) iterations

Runtime

```
Distances(G,s,ℓ)
  dist(s) ← 0
  While(not all distances found)
    Find minimum over (v,w) ∈ E
      with v discovered w not
      of dist(v)+ℓ(v,w)
    dist(w) ← dist(v)+ℓ(v,w)
    prev(w) ← v
```

O(|V|) iterations

O(|E|) edges

Runtime

O(|V|) iterations

```
Distances(G,s,ℓ)
  dist(s) ← 0
  While(not all distances found)
    Find minimum over (v,w) ∈ E
      with v discovered w not
      of dist(v)+ℓ(v,w)
    dist(w) ← dist(v)+ℓ(v,w)
    prev(w) ← v
```

O(|E|) edges

Runtime O(|V||E|)

Runtime

- This is too slow.

Runtime

- This is too slow.
- **Problem:** Every iteration we have to check every edge.

Runtime

- This is too slow.
- **Problem:** Every iteration we have to check every edge.
- **Idea:** Most of the comparison doesn't change much iteration to iteration. Use to save time.

Runtime

- This is too slow.
- **Problem:** Every iteration we have to check every edge.
- **Idea:** Most of the comparison doesn't change much iteration to iteration. Use to save time.
- **Specifically:** Record for each w best value of dist(v,w)+ ℓ(v,w).

Attempt II

```
Distances(G,s,ℓ)
  For v ∈ V
    dist(v) ← ∞, done(v) ← false
  dist(s) ← 0
  While(not all vertices done)
    Find v not done with minimum dist(v)
    done(v) ← true
    For (v,w) ∈ E
      If dist(v)+ℓ(v,w) < dist(w)
        dist(w) ← dist(v)+ℓ(v,w)
        prev(w) ← v
```

See if better path to w & update dist(w)

Attempt II

```
Distances(G,s,ℓ)
  For v ∈ V
    dist(v) ← ∞, done(v) ← false
  dist(s) ← 0
  While(not all vertices done)
    Find v not done with minimum dist(v)
    done(v) ← true
    For (v,w) ∈ E
      If dist(v)+ℓ(v,w) < dist(w)
        dist(w) ← dist(v)+ℓ(v,w)
        prev(w) ← v
```

Attempt II

```
Distances(G,s,ℓ)
  For v ∈ V
    dist(v) ← ∞, done(v) ← false
  dist(s) ← 0
  While(not all vertices done)
    Find v not done with minimum dist(v)
    done(v) ← true
    For (v,w) ∈ E
      If dist(v)+ℓ(v,w) < dist(w)
        dist(w) ← dist(v)+ℓ(v,w)
        prev(w) ← v
```

O(|V|)

Attempt II

```
Distances(G,s,ℓ)
  For v ∈ V
    dist(v) ← ∞, done(v) ← false
  dist(s) ← 0
  While(not all vertices done)
    Find v not done with minimum dist(v)
    done(v) ← true
    For (v,w) ∈ E
      If dist(v)+ℓ(v,w) < dist(w)
        dist(w) ← dist(v)+ℓ(v,w)
        prev(w) ← v
```

O(|V|)

O(|V|) iterations

Attempt II

```
Distances(G,s,ℓ)
  For v ∈ V
    dist(v) ← ∞, done(v) ← false
  dist(s) ← 0
  While(not all vertices done)
    Find v not done with minimum dist(v)
    done(v) ← true
    For (v,w) ∈ E
      If dist(v)+ℓ(v,w) < dist(w)
        dist(w) ← dist(v)+ℓ(v,w)
        prev(w) ← v
```

O(|V|)

O(|V|) iterations

O(|V|)

Attempt II

```
Distances(G,s,ℓ)
   For v ∈ V
      dist(v) ← ∞, done(v) ← false
   dist(s) ← 0
   While(not all vertices done)
      Find v not done with minimum dist(v)
      done(v) ← true
      For (v,w) ∈ E
         If dist(v)+ℓ(v,w) < dist(w)
            dist(w) ← dist(v)+ℓ(v,w)
            prev(w) ← v
```

O(|V|)

O(|V|) iterations

O(|V|)

O(|E|) total

Attempt II

Runtime: $O(|V|^2+|E|)$

```
Distances(G,s,ℓ)
  For v ∈ V
    dist(v) ← ∞, done(v) ← false
  dist(s) ← 0
  While(not all vertices done)
    Find v not done with minimum dist(v)
    done(v) ← true
    For (v,w) ∈ E
      If dist(v)+ℓ(v,w) < dist(w)
        dist(w) ← dist(v)+ℓ(v,w)
        prev(w) ← v
```

$O(|V|)$

$O(|V|)$ iterations

$O(|V|)$

$O(|E|)$ total

Still too Slow

- Repeatedly ask for smallest vertex
  - Even though not much is changing from round to round, the algorithm is computing the minimum from scratch every time

Still too Slow

- Repeatedly ask for smallest vertex
  - Even though not much is changing from round to round, the algorithm is computing the minimum from scratch every time
- Use a data structure!
  - Data structures help answer a bunch of similar questions faster than answering each question individually

Still too Slow

- Repeatedly ask for smallest vertex
  - Even though not much is changing from round to round, the algorithm is computing the minimum from scratch every time
- Use a data structure!
  - Data structures help answer a bunch of similar questions faster than answering each question individually
- For this kind of question, want a priority queue.

Priority Queue

A Priority Queue is a datastructure that stores elements sorted by a key value.

Priority Queue

A Priority Queue is a datastructure that stores elements sorted by a key value.

Operations:

- Insert – adds a new element to the PQ.
- DecreaseKey – Changes the key of an element of the PQ to a specified *smaller* value.
- DeleteMin – Finds the element with the smallest key and removes it from the PQ.