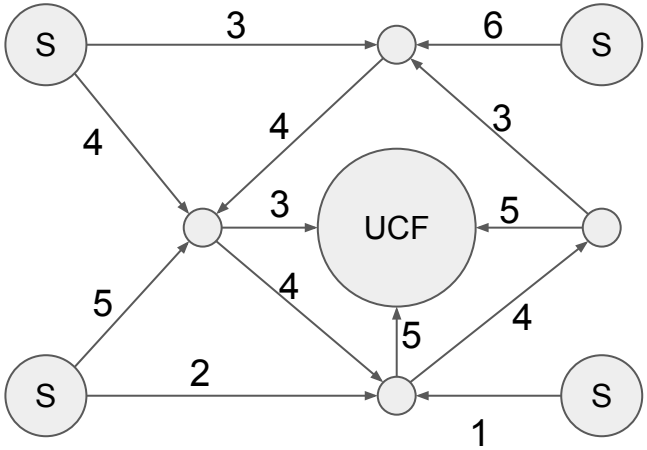
Network Flow Algorithms

Traffic

Suppose we have a large group of people that will leave certain locations (houses) in the morning by vehicle and all travel to the same location (UCF). We know the rate at which cars can travel along all roads (in cars per minute). We want to know the number of cars that can reach the location in cars per minute to determine how long it takes the location to fill up.

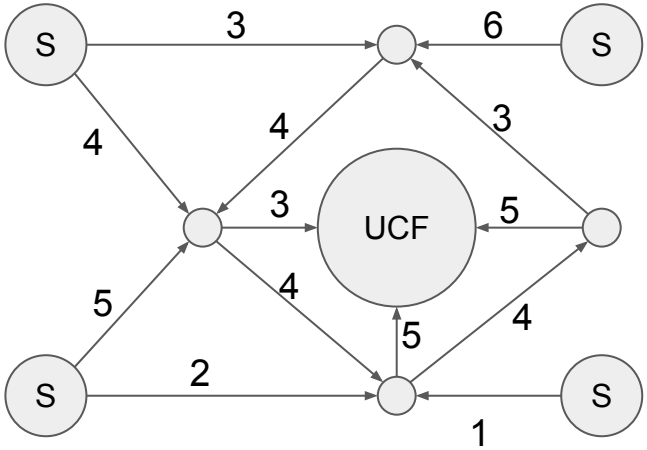
Traffic

Suppose we have a large group of people that will leave certain locations (houses) in the morning by vehicle and all travel to the same location (UCF). We know the rate at which cars can travel along all roads (in cars per minute). We want to know the number of cars that can reach the location in cars per minute to determine how long it takes the location to fill up.



Traffic

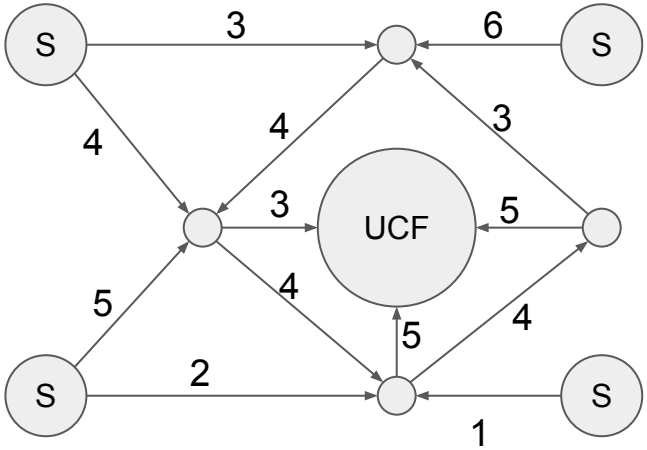
Suppose we have a large group of people that will leave certain locations (houses) in the morning by vehicle and all travel to the same location (UCF). We know the rate at which cars can travel along all roads (in cars per minute). We want to know the number of cars that can reach the location in cars per minute to determine how long it takes the location to fill up.



Ideally we could use the sum of the edges leaving S.

Traffic

Suppose we have a large group of people that will leave certain locations (houses) in the morning by vehicle and all travel to the same location (UCF). We know the rate at which cars can travel along all roads (in cars per minute). We want to know the number of cars that can reach the location in cars per minute to determine how long it takes the location to fill up.

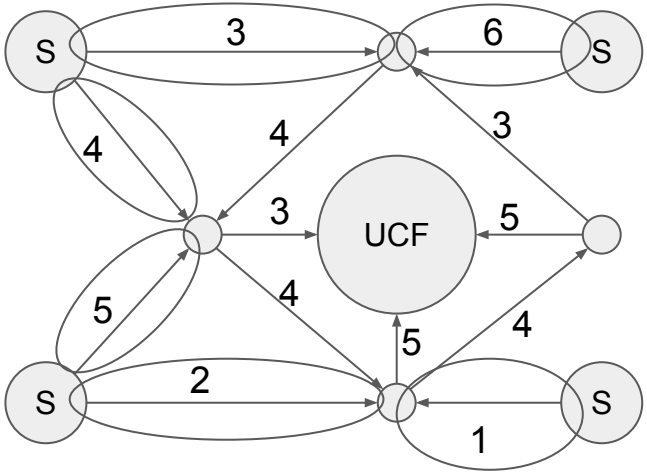


Ideally we could use the sum of the edges leaving S or the sum of the edges entering UCF.

Traffic

Suppose we have a large group of people that will leave certain locations (houses) in the morning by vehicle and all travel to the same location (UCF). We know the rate at which cars can travel along all roads (in cars per minute). We want to know the number of cars that can reach the location in cars per minute to determine how long it takes the location to fill up.

Leaving S = 21

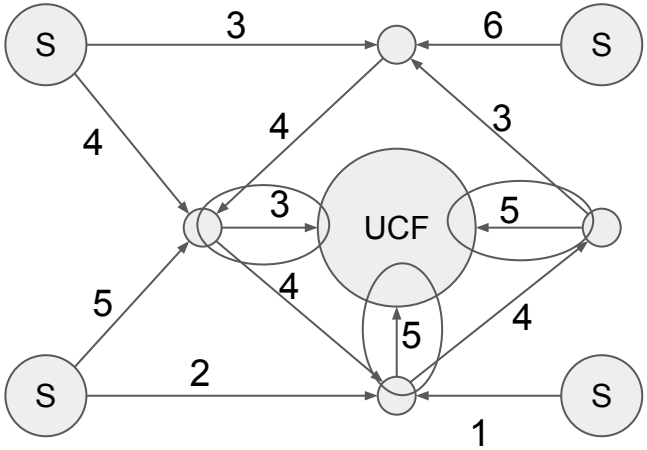


Ideally we could use the sum of the edges leaving S or the sum of the edges entering UCF.

Traffic

Suppose we have a large group of people that will leave certain locations (houses) in the morning by vehicle and all travel to the same location (UCF). We know the rate at which cars can travel along all roads (in cars per minute). We want to know the number of cars that can reach the location in cars per minute to determine how long it takes the location to fill up.

Leaving S = 21
Entering UCF = 13

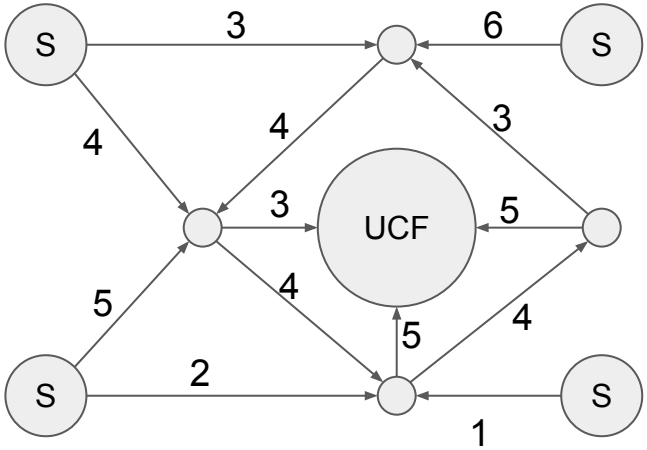


Ideally we could use the sum of the edges leaving S or the sum of the edges entering UCF.

Traffic

Suppose we have a large group of people that will leave certain locations (houses) in the morning by vehicle and all travel to the same location (UCF). We know the rate at which cars can travel along all roads (in cars per minute). We want to know the number of cars that can reach the location in cars per minute to determine how long it takes the location to fill up.

Leaving S = 21
Entering UCF = 13
13 is the bounding value.

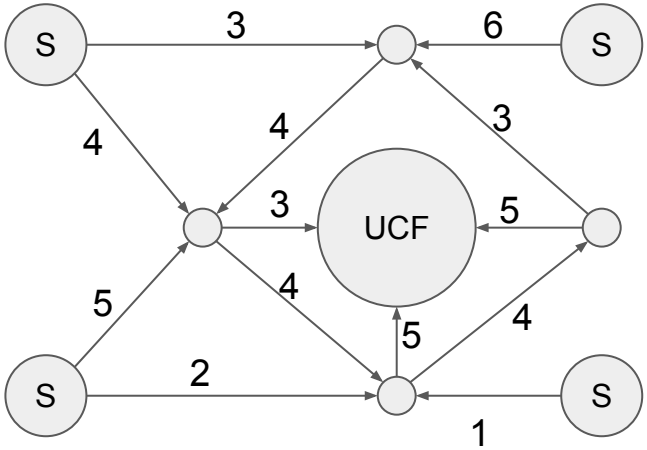


Ideally we could use the sum of the edges leaving S or the sum of the edges entering UCF.

Traffic

Suppose we have a large group of people that will leave certain locations (houses) in the morning by vehicle and all travel to the same location (UCF). We know the rate at which cars can travel along all roads (in cars per minute). We want to know the number of cars that can reach the location in cars per minute to determine how long it takes the location to fill up.

Leaving S = 21
Entering UCF = 13
13 is the bounding value. Is it correct?



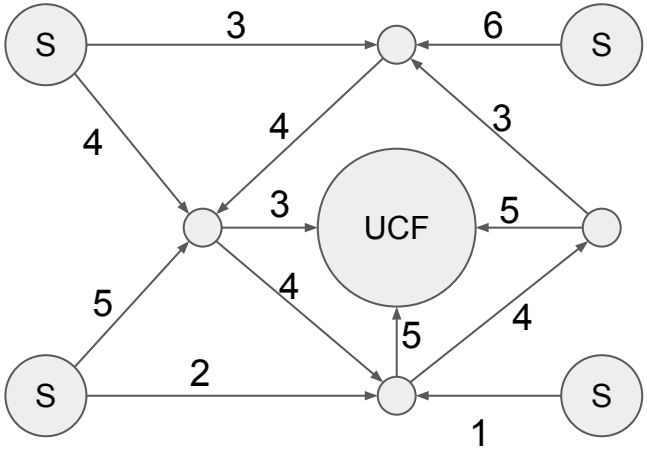
Ideally we could use the sum of the edges leaving S or the sum of the edges entering UCF.

Traffic

Suppose we have a large group of people that will leave certain locations (houses) in the morning by vehicle and all travel to the same location (UCF). We know the rate at which cars can travel along all roads (in cars per minute). We want to know the number of cars that can reach the location in cars per minute to determine how long it takes the location to fill up.

Leaving S = 21
Entering UCF = 13
13 is the bounding value. Is it correct?

The actual answer is 10.



Ideally we could use the sum of the edges leaving S or the sum of the edges entering UCF.

Solution Ideas

We could try sending one car at a time.

Solution Ideas

We could try sending one car at a time.

How do we find if a car can reach UCF?

Solution Ideas

We could try sending one car at a time.

How do we find if a car can reach UCF?

We can DFS the car from S to UCF.

Solution Ideas

We could try sending one car at a time.

How do we find if a car can reach UCF?

We can DFS the car from S to UCF.

Force as many cars along the found path as possible.

Solution Ideas

We could try sending one car at a time.

How do we find if a car can reach UCF?

We can DFS the car from S to UCF.

- Force as many cars along the found path as possible.
- Adjust the graph by “removing edges”.

Solution Ideas

We could try sending one car at a time.

How do we find if a car can reach UCF?

We can DFS the car from S to UCF.

- Force as many cars along the found path as possible.
- Adjust the graph by “removing edges”.
- If UCF is not reached, we can assume we are good.

Solution Ideas

We could try sending one car at a time.

How do we find if a car can reach UCF?

We can DFS the car from S to UCF.

- Force as many cars along the found path as possible.
- Adjust the graph by “removing edges”.
- If UCF is not reached, we can assume we are good.

In action

Solution Ideas

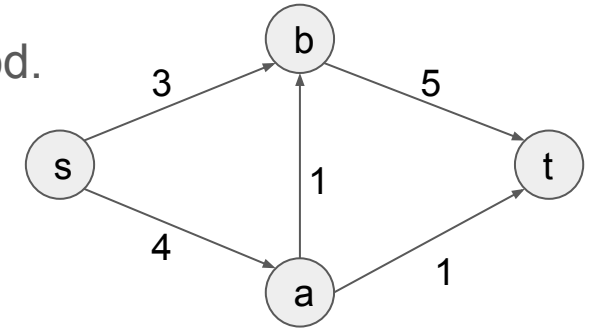
We could try sending one car at a time.

How do we find if a car can reach UCF?

We can DFS the car from S to UCF.

- Force as many cars along the found path as possible.
- Adjust the graph by “removing edges”.
- If UCF is not reached, we can assume we are good.

In action



Solution Ideas

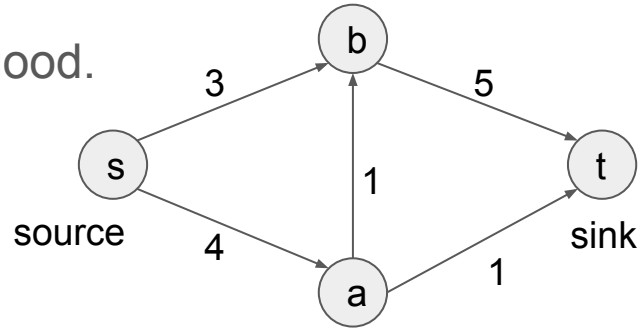
We could try sending one car at a time.

How do we find if a car can reach UCF?

We can DFS the car from S to UCF.

- Force as many cars along the found path as possible.
- Adjust the graph by “removing edges”.
- If UCF is not reached, we can assume we are good.

In action



Solution Ideas

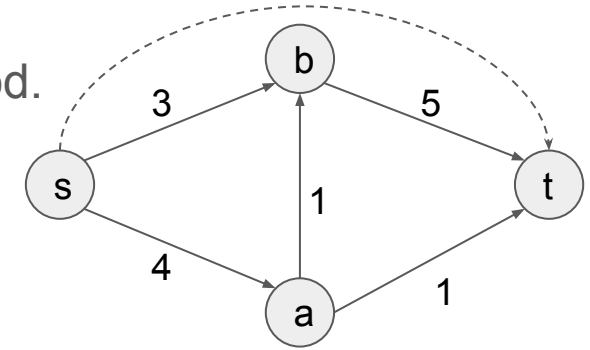
We could try sending one car at a time.

How do we find if a car can reach UCF?

We can DFS the car from S to UCF.

- Force as many cars along the found path as possible.
- Adjust the graph by “removing edges”.
- If UCF is not reached, we can assume we are good.

In action



Solution Ideas

We could try sending one car at a time.

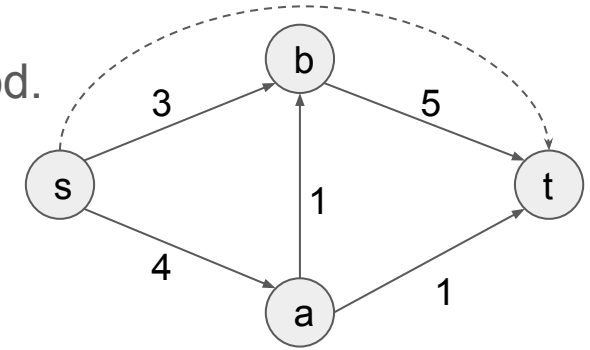
How do we find if a car can reach UCF?

We can DFS the car from S to UCF.

- Force as many cars along the found path as possible.
- Adjust the graph by “removing edges”.
- If UCF is not reached, we can assume we are good.

In action

Send 3 cars



Solution Ideas

We could try sending one car at a time.

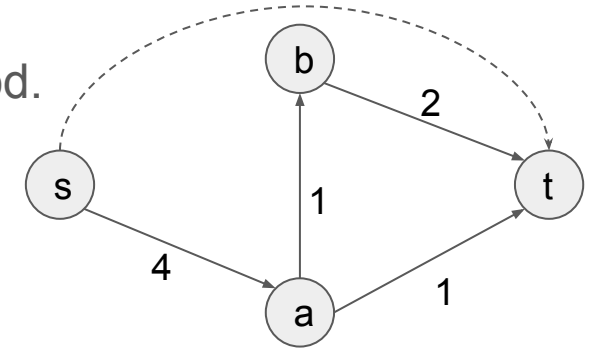
How do we find if a car can reach UCF?

We can DFS the car from S to UCF.

- Force as many cars along the found path as possible.
- Adjust the graph by “removing edges”.
- If UCF is not reached, we can assume we are good.

In action

Send 3 cars



Solution Ideas

We could try sending one car at a time.

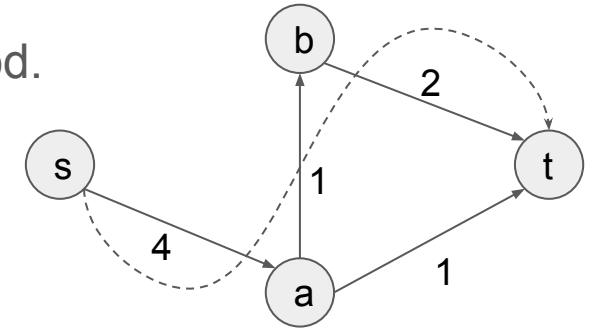
How do we find if a car can reach UCF?

We can DFS the car from S to UCF.

- Force as many cars along the found path as possible.
- Adjust the graph by “removing edges”.
- If UCF is not reached, we can assume we are good.

In action

Send 3 cars



Solution Ideas

We could try sending one car at a time.

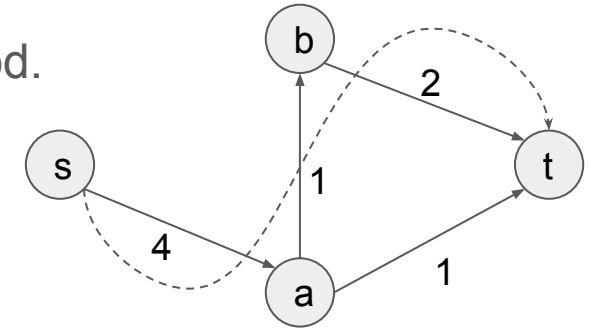
How do we find if a car can reach UCF?

We can DFS the car from S to UCF.

- Force as many cars along the found path as possible.
- Adjust the graph by “removing edges”.
- If UCF is not reached, we can assume we are good.

In action

Send 3 cars
Send 1 car



Solution Ideas

We could try sending one car at a time.

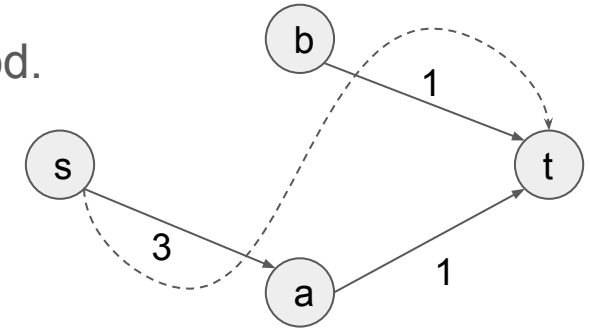
How do we find if a car can reach UCF?

We can DFS the car from S to UCF.

- Force as many cars along the found path as possible.
- Adjust the graph by “removing edges”.
- If UCF is not reached, we can assume we are good.

In action

Send 3 cars
Send 1 car



Solution Ideas

We could try sending one car at a time.

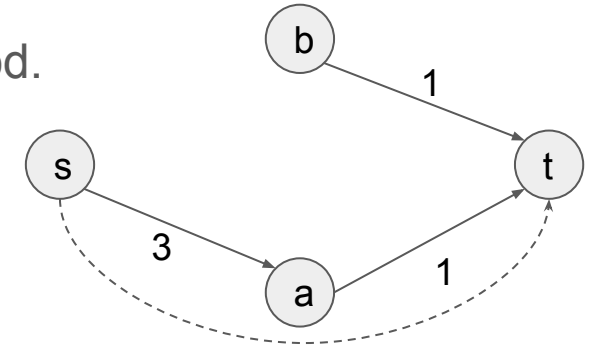
How do we find if a car can reach UCF?

We can DFS the car from S to UCF.

- Force as many cars along the found path as possible.
- Adjust the graph by “removing edges”.
- If UCF is not reached, we can assume we are good.

In action

Send 3 cars
Send 1 car



Solution Ideas

We could try sending one car at a time.

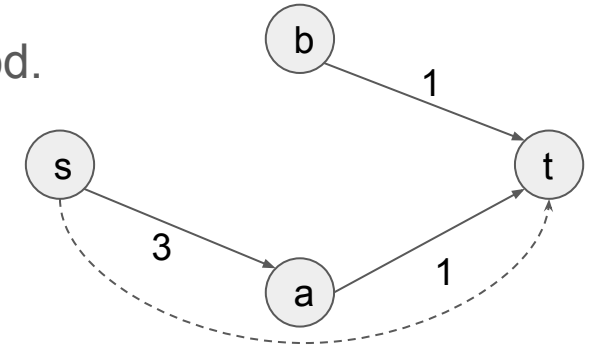
How do we find if a car can reach UCF?

We can DFS the car from S to UCF.

- Force as many cars along the found path as possible.
- Adjust the graph by “removing edges”.
- If UCF is not reached, we can assume we are good.

In action

Send 3 cars
Send 1 car
Send 1 car



Solution Ideas

We could try sending one car at a time.

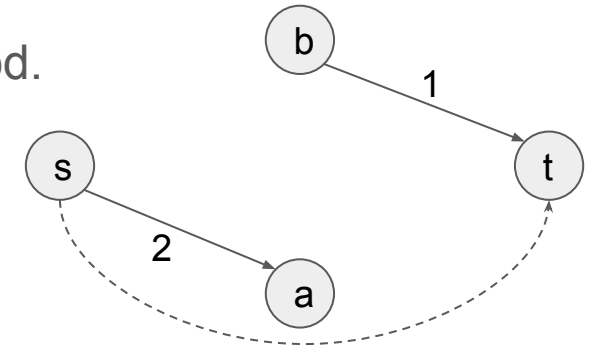
How do we find if a car can reach UCF?

We can DFS the car from S to UCF.

- Force as many cars along the found path as possible.
- Adjust the graph by “removing edges”.
- If UCF is not reached, we can assume we are good.

In action

Send 3 cars
Send 1 car
Send 1 car



Solution Ideas

We could try sending one car at a time.

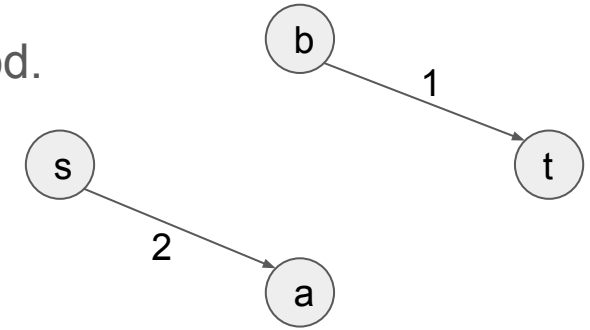
How do we find if a car can reach UCF?

We can DFS the car from S to UCF.

- Force as many cars along the found path as possible.
- Adjust the graph by “removing edges”.
- If UCF is not reached, we can assume we are good.

In action

Send 3 cars
Send 1 car
Send 1 car
No Remaining Paths

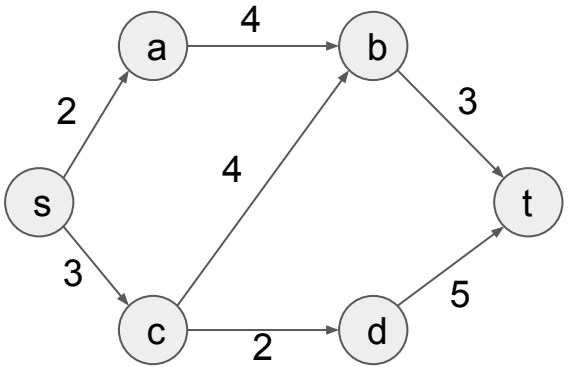


Solution Ideas (cont.)

That idea has some issues.

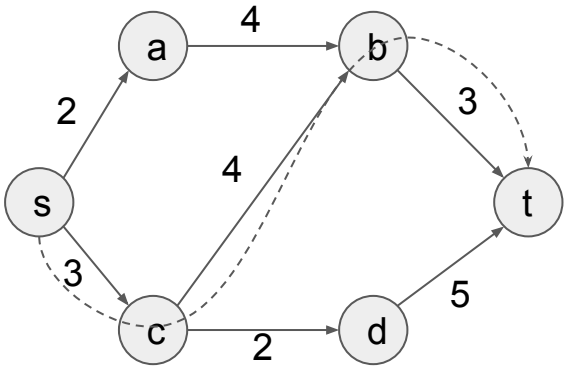
Solution Ideas (cont.)

That idea has some issues.



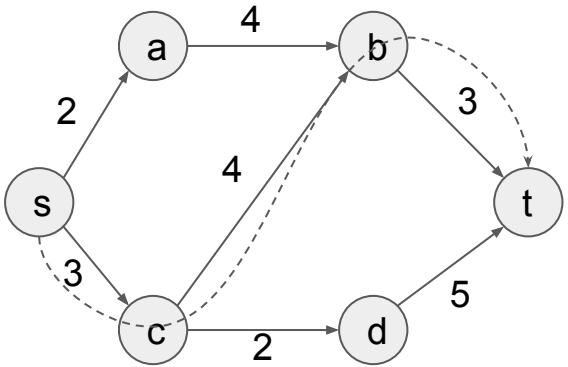
Solution Ideas (cont.)

That idea has some issues.



Solution Ideas (cont.)

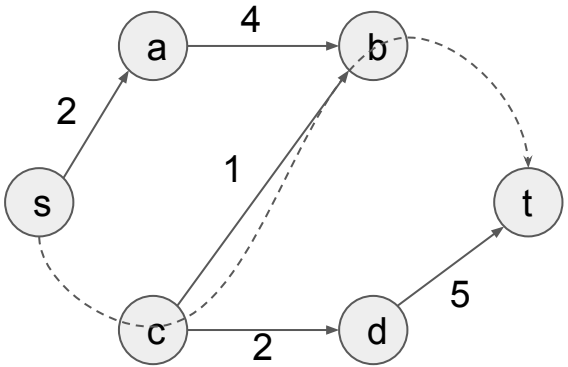
That idea has some issues.



Send 3 cars

Solution Ideas (cont.)

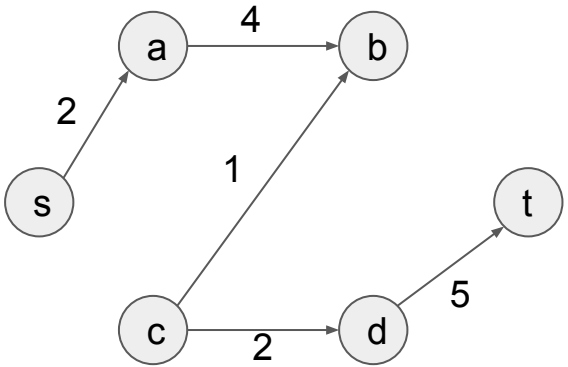
That idea has some issues.



Send 3 cars

Solution Ideas (cont.)

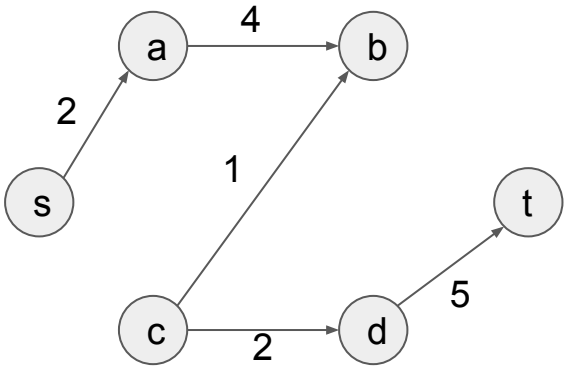
That idea has some issues.



Send 3 cars

Solution Ideas (cont.)

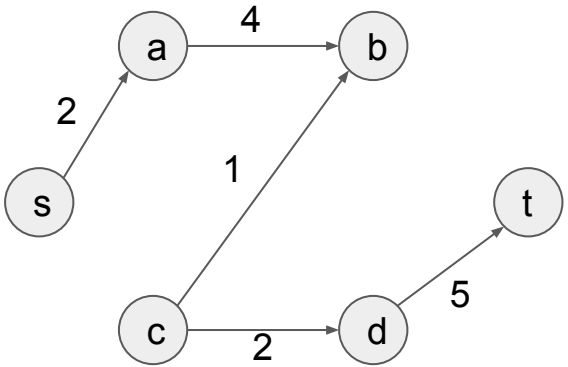
That idea has some issues.



Send 3 cars
No path.

Solution Ideas (cont.)

That idea has some issues.

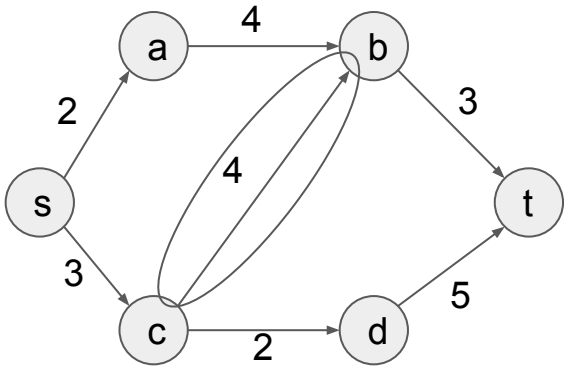


Send 3 cars
No path.
But what was the answer?

Solution Ideas (cont.)

That idea has some issues.

We made a mistake early by taking the center edge.

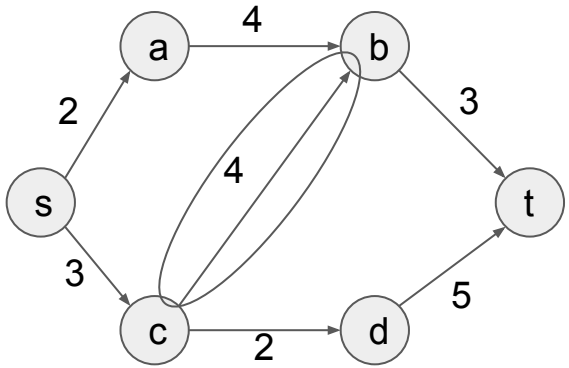


Solution Ideas (cont.)

That idea has some issues.

We made a mistake early by taking the center edge.

We need to allow ourselves to undo mistakes.



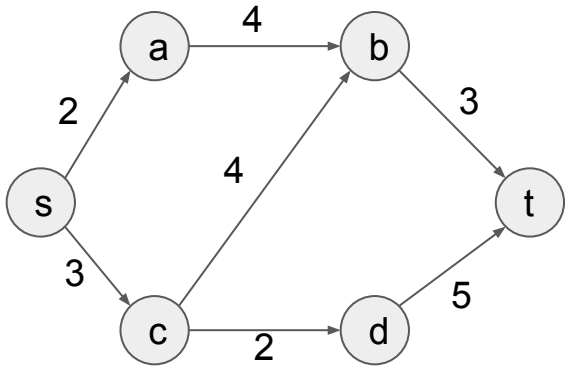
Solution Ideas (cont.)

That idea has some issues.

We made a mistake early by taking the center edge.

We need to allow ourselves to undo mistakes.

Create residual edges.



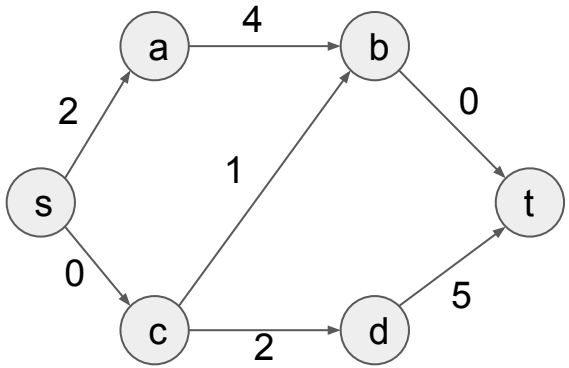
Solution Ideas (cont.)

That idea has some issues.

We made a mistake early by taking the center edge.

We need to allow ourselves to undo mistakes.

Create residual edges.



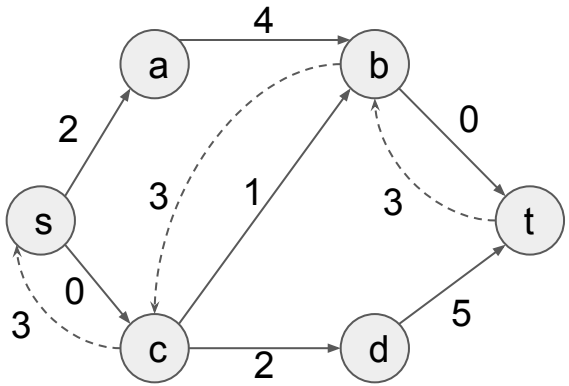
Solution Ideas (cont.)

That idea has some issues.

We made a mistake early by taking the center edge.

We need to allow ourselves to undo mistakes.

Create residual edges.



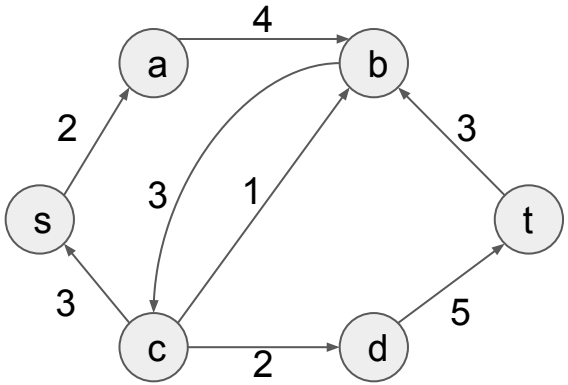
Solution Ideas (cont.)

That idea has some issues.

We made a mistake early by taking the center edge.

We need to allow ourselves to undo mistakes.

Create residual edges.



Solution Ideas (cont.)

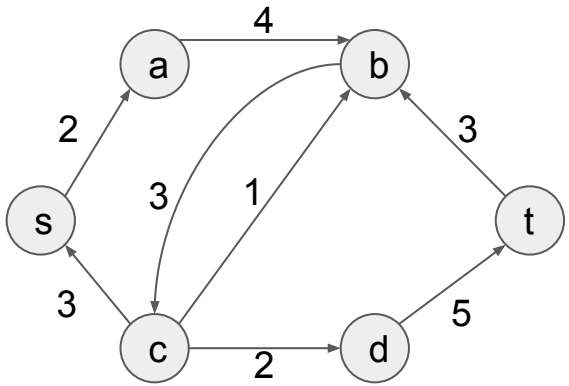
That idea has some issues.

We made a mistake early by taking the center edge.

We need to allow ourselves to undo mistakes.

Create residual edges.

Continue the algorithm.



Solution Ideas (cont.)

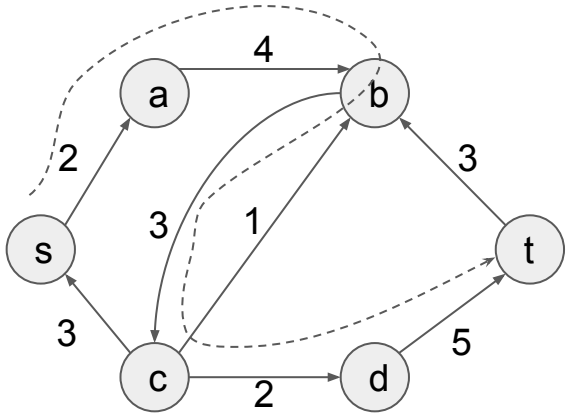
That idea has some issues.

We made a mistake early by taking the center edge.

We need to allow ourselves to undo mistakes.

Create residual edges.

Continue the algorithm.



Solution Ideas (cont.)

That idea has some issues.

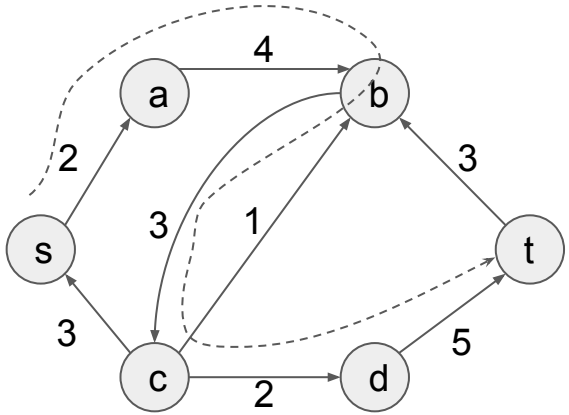
We made a mistake early by taking the center edge.

We need to allow ourselves to undo mistakes.

Create residual edges.

Continue the algorithm.

Send 2 cars
(in addition to the 3 other cars)



Solution Ideas (cont.)

That idea has some issues.

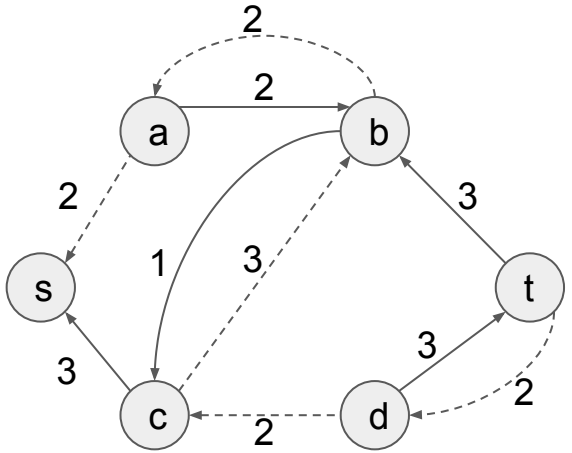
We made a mistake early by taking the center edge.

We need to allow ourselves to undo mistakes.

Create residual edges.

Continue the algorithm.

Send 2 cars
(in addition to the 3 other cars)



Solution Ideas (cont.)

That idea has some issues.

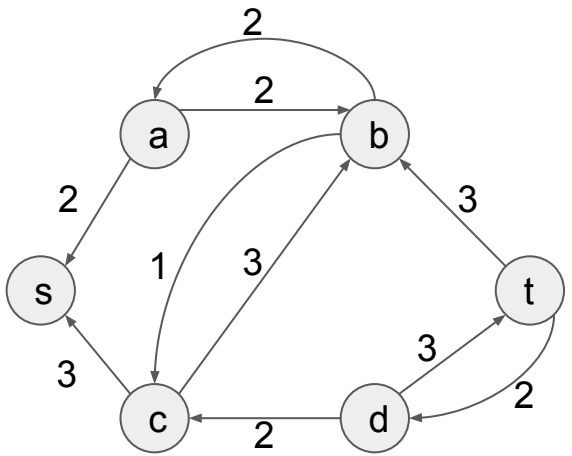
We made a mistake early by taking the center edge.

We need to allow ourselves to undo mistakes.

Create residual edges.

Continue the algorithm.

Send 2 cars
(in addition to the 3 other cars)



Solution Ideas (cont.)

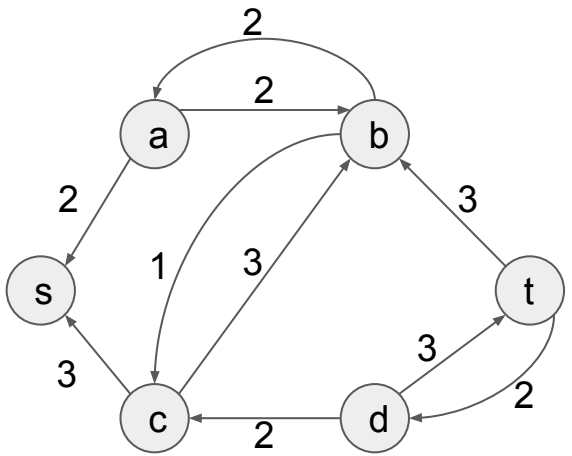
That idea has some issues.

We made a mistake early by taking the center edge.

We need to allow ourselves to undo mistakes.

Create residual edges.

Continue the algorithm.



Send 2 cars
(in addition to the 3 other cars)
What happens if we DFS now?

Solution Ideas (cont.)

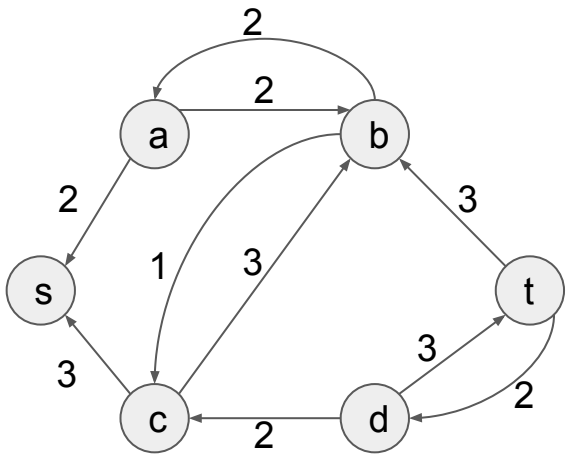
That idea has some issues.

We made a mistake early by taking the center edge.

We need to allow ourselves to undo mistakes.

Create residual edges.

Continue the algorithm.



Send 2 cars
(in addition to the 3 other cars)
What happens if we DFS now?
No path from s to t!

Correctness

By adding the reverse edges,

Correctness

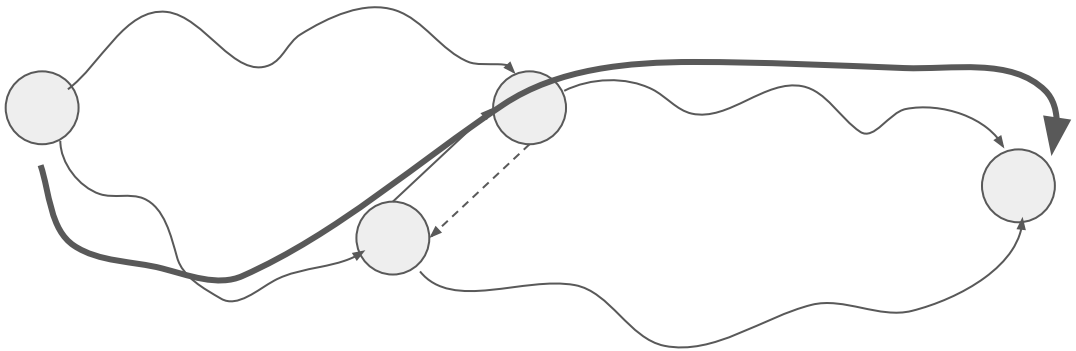
By adding the reverse edges,

We allow ourselves to pick patchwork paths together

Correctness

By adding the reverse edges,

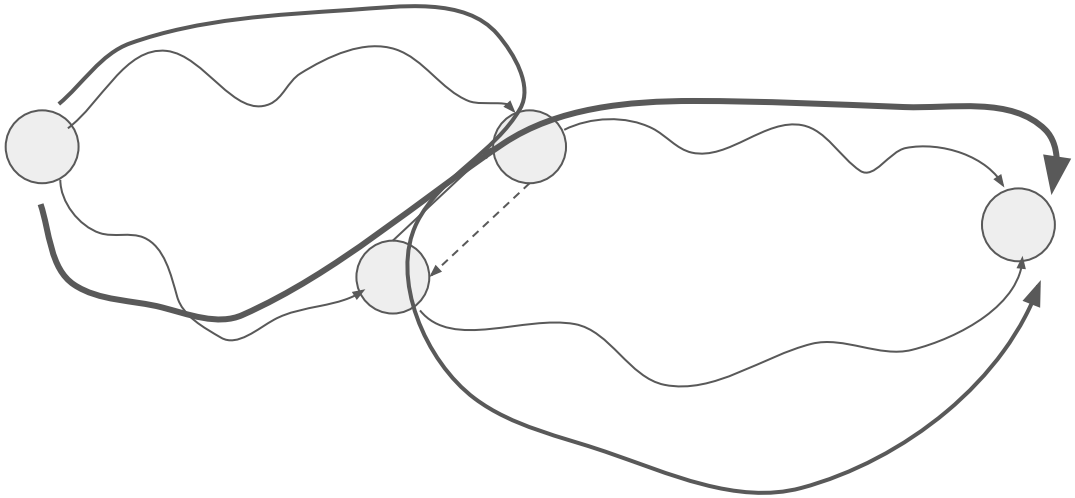
We allow ourselves to pick patchwork paths together



Correctness

By adding the reverse edges,

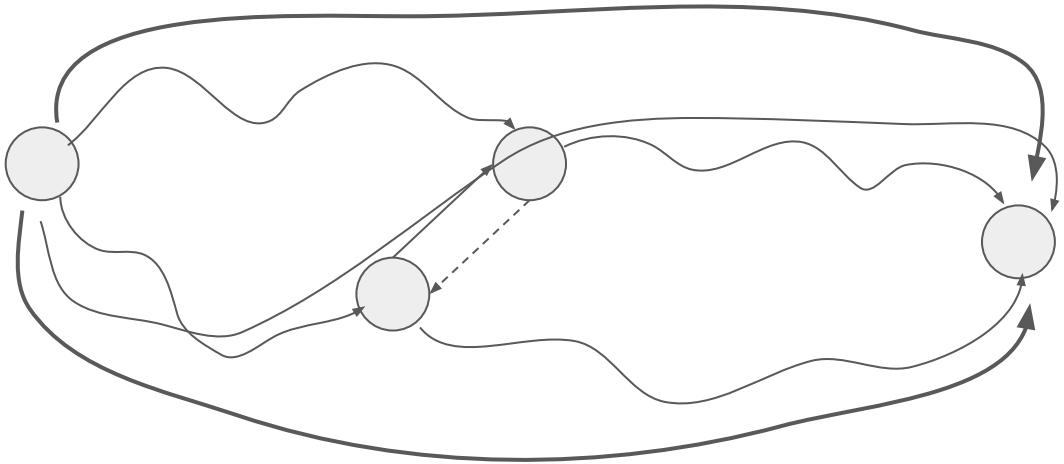
We allow ourselves to pick patchwork paths together



Correctness

By adding the reverse edges,

We allow ourselves to pick patchwork paths together



Ford-Fulkerson Algorithm (FFA)

```
Let the answer be 0
While a path exists
    Find the path using a DFS
    Increment answer by the minimum capacity along the path
    Adjust capacities along the path
    Add residual edges along the path
```

FFA Analysis

Runtime?

FFA Analysis

Runtime?

(Runtime of DFS) TIMES (Number of paths)

FFA Analysis

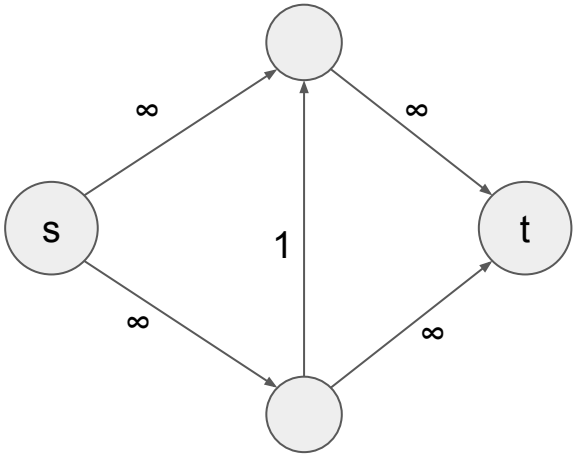
Runtime?

(Runtime of DFS) TIMES (The Answer in worst case)

FFA Analysis

Runtime?

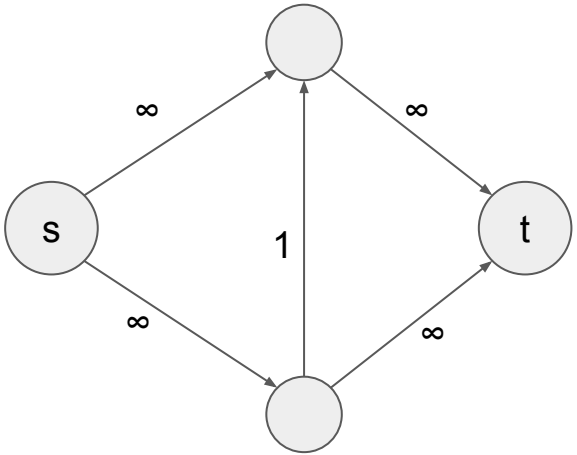
(Runtime of DFS) TIMES (The Answer in worst case)



FFA Analysis

Runtime?

(Runtime of DFS) TIMES (The Answer in worst case)

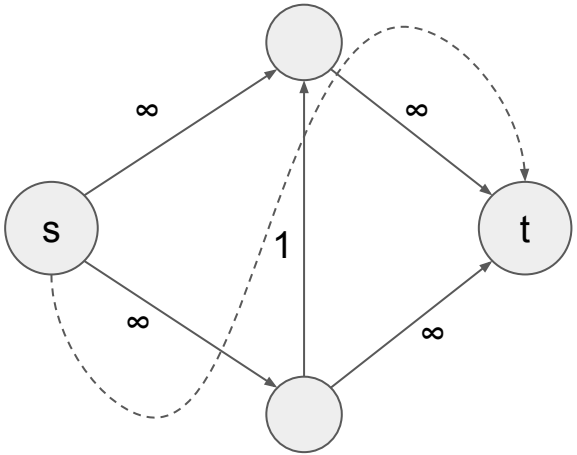


Not really ∞, but a very large value.

FFA Analysis

Runtime?

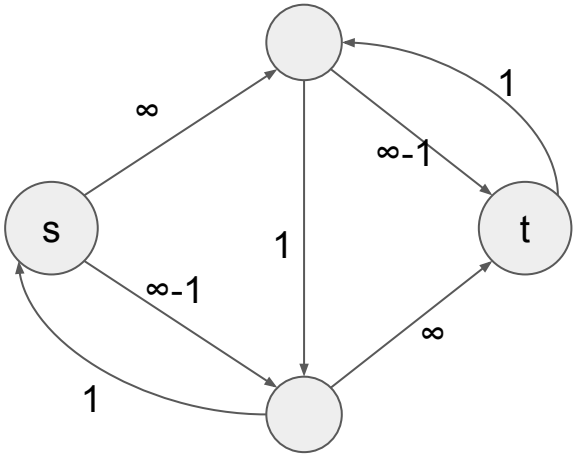
(Runtime of DFS) TIMES (The Answer in worst case)



FFA Analysis

Runtime?

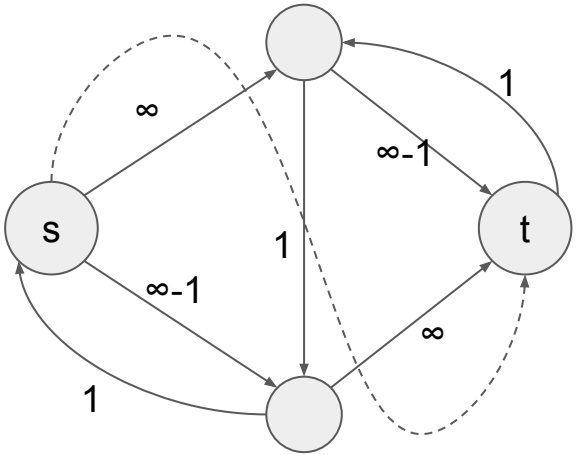
(Runtime of DFS) TIMES (The Answer in worst case)



FFA Analysis

Runtime?

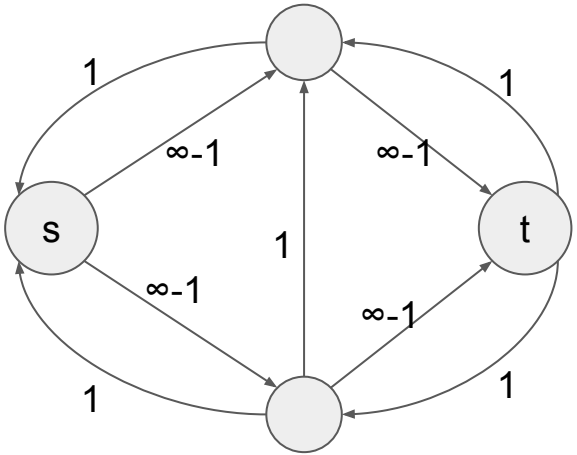
(Runtime of DFS) TIMES (The Answer in worst case)



FFA Analysis

Runtime?

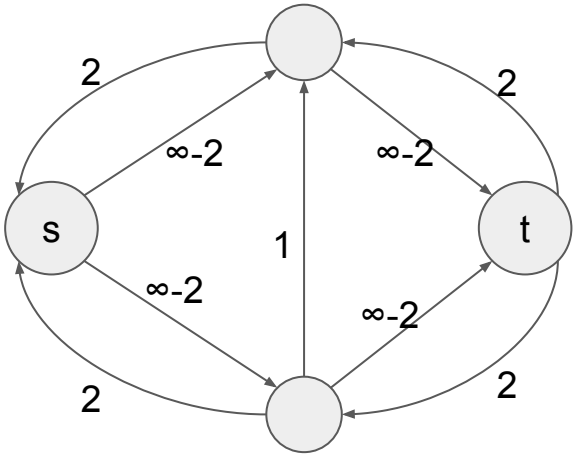
(Runtime of DFS) TIMES (The Answer in worst case)



FFA Analysis

Runtime?

(Runtime of DFS) TIMES (The Answer in worst case)

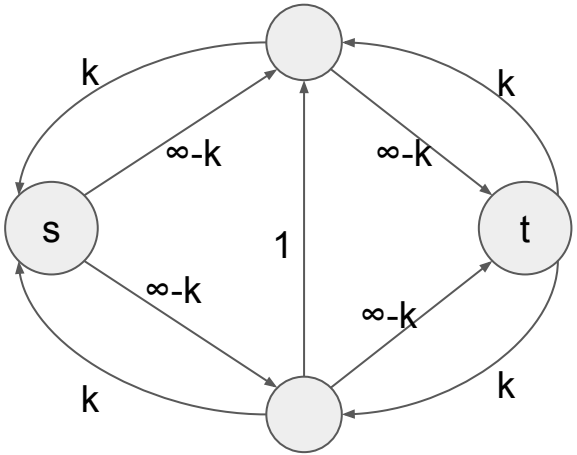


After 2 more DFS's

FFA Analysis

Runtime?

(Runtime of DFS) TIMES (The Answer in worst case)



After 2k total DFS's

FFA Analysis

Runtime?

(Runtime of DFS) TIMES (The Answer in worst case)

There are non-terminating cases with irrational edge weights.

FFA Analysis

Runtime?

(Runtime of DFS) TIMES (The Answer in worst case)

There are non-terminating cases with irrational edge weights.

We typically talk about Flow algorithms in terms of their worst case performance.

FFA Analysis

Runtime?

(Runtime of DFS) TIMES (The Answer in worst case)

There are non-terminating cases with irrational edge weights.

We typically talk about Flow algorithms in terms of their worst case performance.

FFA can be thought of as $O((|E|+|V|)(F))$, where F is the maximum flow.

Fixing FFA

To fix FFA we change our DFS to a BFS.

Fixing FFA

To fix FFA we change our DFS to a BFS.

Using the BFS makes the algorithm something called Edmonds-Karp Algorithm.

Edmonds-Karp Analysis

Push flow once takes $O(|V| + |E|)$.

Edmonds-Karp Analysis

Push flow once takes O(|V| + |E|).

For every push of flow, at least one edge has their capacity filled (saturated).

Edmonds-Karp Analysis

Push flow once takes O(|V| + |E|).
For every push of flow, at least one edge has their capacity filled (saturated).
This edge could be saturated multiple times by undoing flow.

Edmonds-Karp Analysis

Push flow once takes O(|V| + |E|).

For every push of flow, at least one edge has their capacity filled (saturated).

This edge could be saturated multiple times by undoing flow.

An edge’s saturating path length will increase after each saturation.

Edmonds-Karp Analysis

Push flow once takes O(|V| + |E|).

For every push of flow, at least one edge has their capacity filled (saturated).

This edge could be saturated multiple times by undoing flow.

An edge's saturating path length will increase after each saturation.

The path length will be at most O(|V|).

Edmonds-Karp Analysis

Push flow once takes O(|V| + |E|).

For every push of flow, at least one edge has their capacity filled (saturated).

This edge could be saturated multiple times by undoing flow.

An edge's saturating path length will increase after each saturation.

The path length will be at most O(|V|).

But each edge can go through this saturation.

Edmonds-Karp Analysis

Push flow once takes O(|V| + |E|).

For every push of flow, at least one edge has their capacity filled (saturated).

This edge could be saturated multiple times by undoing flow.

An edge's saturating path length will increase after each saturation.

The path length will be at most O(|V|).

But each edge can go through this saturation.

$O((|V|+|E|)(|E|)(|V|))$ or $O(|E|^2|V|)$

Dinitz (Dinic's) Algorithm

Instead of only pushing 1 path of flow…

Dinitz (Dinic's) Algorithm

Instead of only pushing 1 path of flow…

Push a blocking flow.

Dinitz (Dinic's) Algorithm

Instead of only pushing 1 path of flow…

Push a blocking flow.

The blocking flow provably has a worst case of $O(|E||V|^2)$

Push-Relabel

A last method that is non-trivial to explain.

Push-Relabel

A last method that is non-trivial to explain.

Original description had a runtime of $O(|V|^2|E|)$ (like Dinitz)

Push-Relabel

A last method that is non-trivial to explain.

Original description had a runtime of $O(|V|^2|E|)$ (like Dinitz)

An optimized version has $O(|V|^2|E|^{0.5})$

Tidal Flow and Benchmarking

A colleague of mine Matt Fontaine developed a flow algorithm called Tidal Flow.

Tidal Flow and Benchmarking

A colleague of mine Matt Fontaine developed a flow algorithm called Tidal Flow.

They decided to show it’s effectiveness on various graph structures.

Tidal Flow and Benchmarking

A colleague of mine Matt Fontaine developed a flow algorithm called Tidal Flow.

They decided to show it’s effectiveness on various graph structures.

It theoretically runs in $O(|V|^2|E|)$.

Tidal Flow and Benchmarking

A colleague of mine Matt Fontaine developed a flow algorithm called Tidal Flow.

They decided to show it’s effectiveness on various graph structures.

It theoretically runs in $O(|V|^2|E|)$.

https://ioinformatics.org/journal/v12_2018_25_41.pdf