# 7.3 Choosing Good Augmenting Paths

# Ford-Fulkerson: Exponential Number of Augmentations

Q. Is generic Ford-Fulkerson algorithm polynomial in input size?

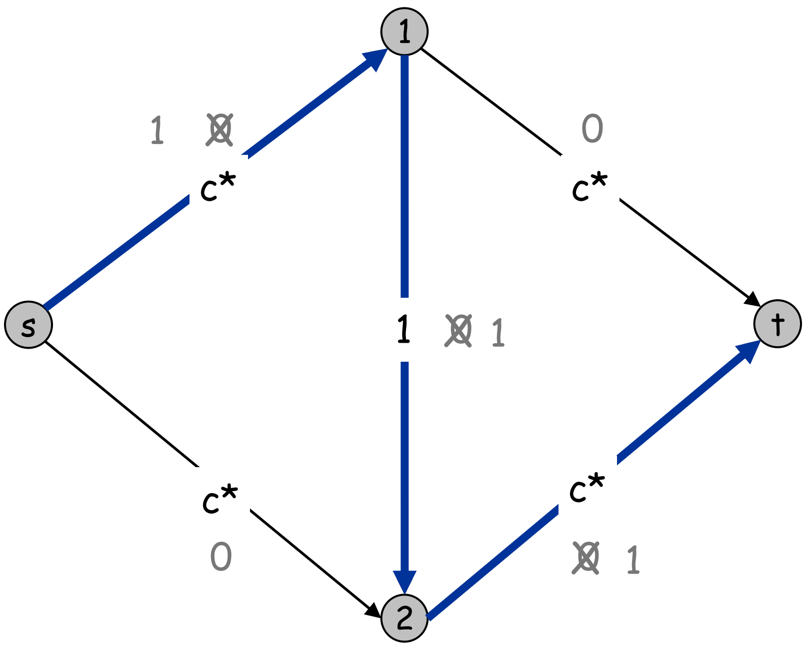
m, n, and log $c^*$

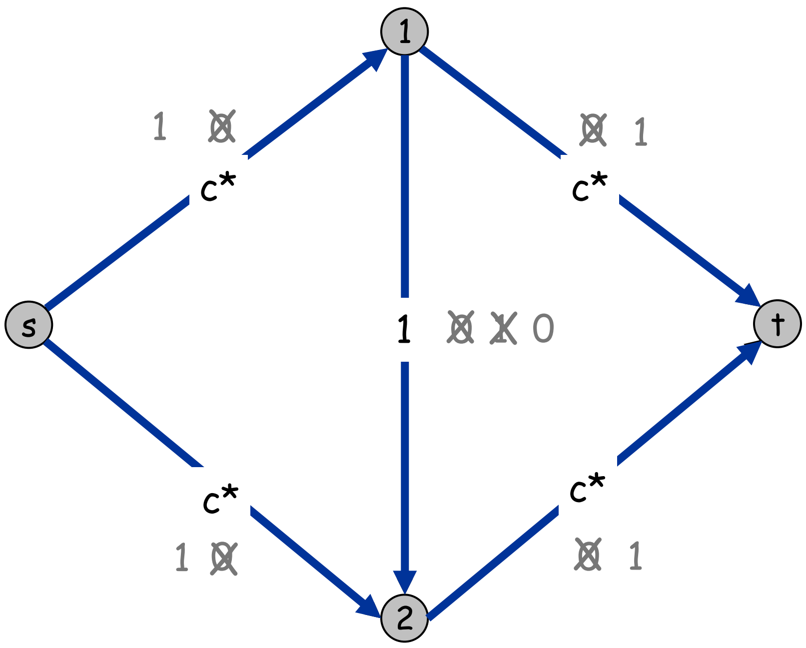
# Ford-Fulkerson: Exponential Number of Augmentations

Q. Is generic Ford-Fulkerson algorithm polynomial in input size?

m, n, and log c*

A. No. If max capacity is c*, then algorithm can take nc* iterations.



# Choosing Good Augmenting Paths

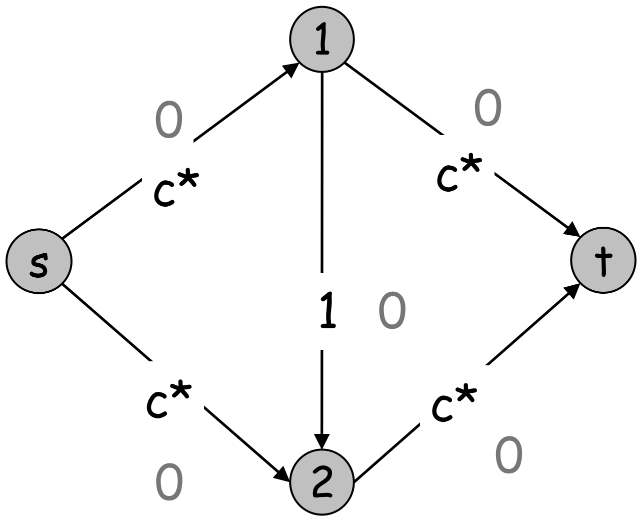
Use care when selecting augmenting paths.

- Some choices lead to exponential algorithms.
- Clever choices lead to polynomial algorithms.
- If capacities are irrational, algorithm not guaranteed to terminate!

Goal: choose augmenting paths so that:

- Can find augmenting paths efficiently.
- Few iterations.

Q. How to choose "good" augmenting paths?



# Choosing Good Augmenting Paths

Use care when selecting augmenting paths.

- Some choices lead to exponential algorithms.
- Clever choices lead to polynomial algorithms.
- If capacities are irrational, algorithm not guaranteed to terminate!

Goal: choose augmenting paths so that:

- Can find augmenting paths efficiently.
- Few iterations.

Choose augmenting paths with: [Edmonds-Karp 1972, Dinitz 1970]

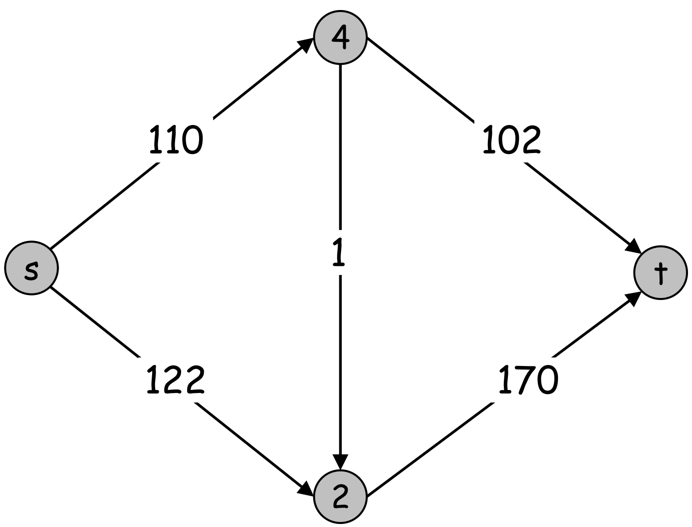
- Max bottleneck capacity.
- Sufficiently large bottleneck capacity.
- Fewest number of edges.

TU Delft

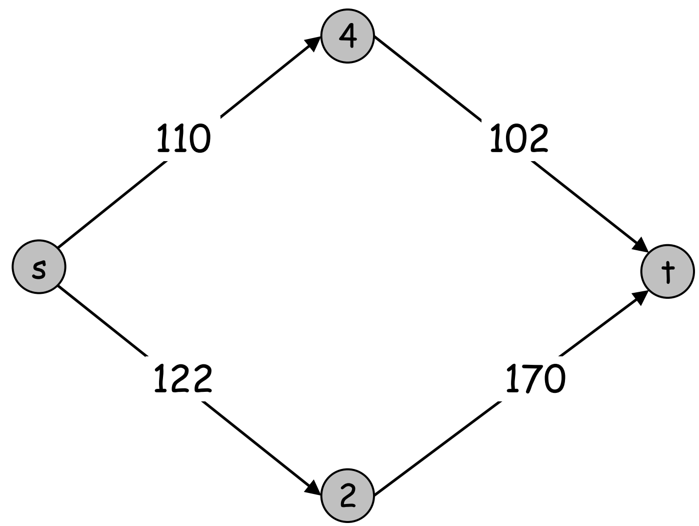
# Capacity Scaling

Intuition. Choosing path with sufficiently high bottleneck capacity

- Maintain scaling parameter $\Delta$.
- Find flow in the subgraph $G_f(\Delta)$ of the residual graph consisting of only arcs with capacity at least $\Delta$.




$G_f$




$G_f(64)$

# Capacity Scaling

```
Scaling-Max-Flow(G, s, t, c) {
   foreach e ∈ E  f(e) ← 0
   Δ ← smallest power of 2 greater than or equal to c*
   G_f ← residual graph

   while (Δ ≥ 1) {
      G_f(Δ) ← Δ-residual graph
      while (there exists augmenting path P in G_f(Δ)) {
         f ← augment(f, c, P)
         update G_f(Δ)
      }
      Δ ← Δ / 2
   }
   return f
}
```

Q. Why is this algorithm correct? (Why does it terminates with a max flow?)

# Capacity Scaling: Correctness

Assumption. All edge capacities are integers between 1 and c*.

Integrality invariant. All flow and residual capacity values are integral.

Q. Why is this algorithm correct? (Why does it terminates with a max flow?)

## Capacity Scaling: Correctness

Assumption. All edge capacities are integers between 1 and c*.

Integrality invariant. All flow and residual capacity values are integral.

Correctness. If the algorithm terminates, then f is a max flow.

Pf.

- By integrality invariant, when $\Delta = 1 \Rightarrow G_f(\Delta) = G_f$.
- Upon termination of $\Delta = 1$ phase, there are no augmenting paths. ▪

TU Delft

# Capacity Scaling: Running Time

```
Scaling-Max-Flow(G, s, t, c) {
   foreach e ∈ E  f(e) ← 0
   Δ ← smallest power of 2 greater than or equal to c*
   G_f ← residual graph

   while (Δ ≥ 1) {
      G_f(Δ) ← Δ-residual graph
      while (there exists augmenting path P in G_f(Δ)) {
         f ← augment(f, c, P)
         update G_f(Δ)
      }
      Δ ← Δ / 2
   }
   return f
}
```

Q. How many scaling phases? (= the outer loop)

Q. How many augmentations per scaling phase?

# Capacity Scaling: Running Time

Q. How many scaling phases? (= the outer loop)
A. The outer while loop repeats $1 + \lceil \log_2 c^* \rceil$ times.
Pf. Initially $c^* \le \Delta < 2c^*$. $\Delta$ decreases by a factor of 2 each iteration. ▪

Q. How many augmentations per scaling phase?
A. ?
Q. How much is an increase in flow for *one augmentation* in a $\Delta$-phase?
A. Each augmentation in a $\Delta$-phase increases v(f') by at least $\Delta$.

Q. What is the maximum increase in flow in a *whole* $\Delta$-phase?
A. Previous phase with $2\Delta$ "missed" less than $2\Delta$ for each edge, so maximum increase at most $m2\Delta$.
So at most $m2\Delta/\Delta = 2m$ augmentations per scaling phase.

Theorem. The scaling max-flow algorithm finds a max flow in $O(m \log c^*)$ augmentations. It can be implemented to run in $O(m^2 \log c^*)$ time.

TU Delft