How to choose augmenting paths?

▶ We need to find paths efficiently.

How to choose augmenting paths?

- ▶ We need to find paths efficiently.
- ▶ We want to guarantee a small number of iterations.

How to choose augmenting paths?

▶ We need to find paths efficiently.

▶ We want to guarantee a small number of iterations.

Several possibilities:

How to choose augmenting paths?

- We need to find paths efficiently.
- We want to guarantee a small number of iterations.

Several possibilities:

- Choose path with maximum bottleneck capacity.
- Choose path with sufficiently large bottleneck capacity.
- Choose the shortest augmenting path.

Capacity Scaling

Capacity Scaling

Intuition:

▶ Choosing a path with the highest bottleneck increases the flow as much as possible in a single step.

Capacity Scaling

Intuition:

▶ Choosing a path with the highest bottleneck increases the flow as much as possible in a single step.

▶ Don't worry about finding the exact bottleneck.

Capacity Scaling

Intuition:

- ▶ Choosing a path with the highest bottleneck increases the flow as much as possible in a single step.
- ▶ Don't worry about finding the exact bottleneck.
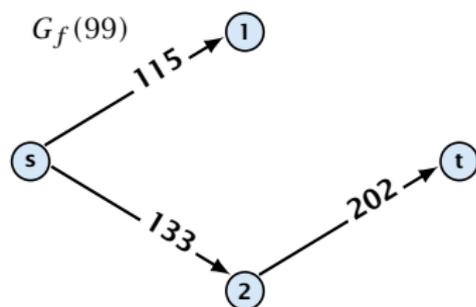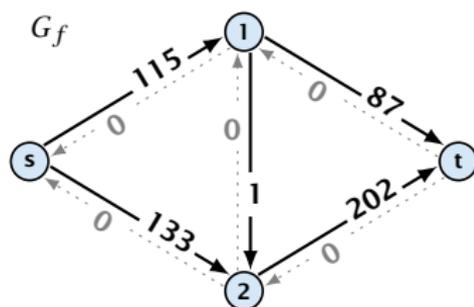- ▶ Maintain scaling parameter Δ.

Capacity Scaling

Intuition:

- ▶ Choosing a path with the highest bottleneck increases the flow as much as possible in a single step.
- ▶ Don't worry about finding the exact bottleneck.
- ▶ Maintain scaling parameter Δ.
- ▶ $G_f(\Delta)$ is a sub-graph of the residual graph G_f that contains only edges with capacity at least Δ.

Capacity Scaling

Intuition:

- ▶ Choosing a path with the highest bottleneck increases the flow as much as possible in a single step.
- ▶ Don't worry about finding the exact bottleneck.
- ▶ Maintain scaling parameter Δ.
- ▶ $G_f(\Delta)$ is a sub-graph of the residual graph G_f that contains only edges with capacity at least Δ.

Capacity Scaling

Algorithm 45 maxflow(G, s, t, c)

1: **foreach** $e \in E$ **do** $f_e \leftarrow 0$;
2: $\Delta \leftarrow 2^{\lceil \log_2 C \rceil}$
3: **while** $\Delta \geq 1$ **do**
4:     $G_f(\Delta) \leftarrow \Delta$-residual graph
5:     **while** there is augmenting path P in $G_f(\Delta)$ **do**
6:         $f \leftarrow$ augment(f, c, P)
7:         update($G_f(\Delta)$)
8:     $\Delta \leftarrow \Delta/2$
9: **return** f

Capacity Scaling

Capacity Scaling

Assumption:
All capacities are integers between 1 and C.

Capacity Scaling

Assumption:
All capacities are integers between 1 and C.

Invariant:
All flows and capacities are/remain integral throughout the algorithm.

Capacity Scaling

Assumption:
All capacities are integers between 1 and C.

Invariant:
All flows and capacities are/remain integral throughout the algorithm.

Correctness:
The algorithm computes a maxflow:

- ▶ because of integrality we have $G_f(1) = G_f$

Capacity Scaling

Assumption:
All capacities are integers between 1 and C.

Invariant:
All flows and capacities are/remain integral throughout the algorithm.

Correctness:
The algorithm computes a maxflow:

- ▶ because of integrality we have $G_f(1) = G_f$
- ▶ therefore after the last phase there are no augmenting paths anymore

Capacity Scaling

Assumption:
All capacities are integers between 1 and C.

Invariant:
All flows and capacities are/remain integral throughout the algorithm.

Correctness:
The algorithm computes a maxflow:

- ▶ because of integrality we have $G_f(1) = G_f$
- ▶ therefore after the last phase there are no augmenting paths anymore
- ▶ this means we have a maximum flow.

Capacity Scaling

Capacity Scaling

Lemma 1

There are $\lceil \log C \rceil$ iterations over Δ.

Proof: obvious.

Capacity Scaling

Lemma 1
There are $\lceil \log C \rceil$ iterations over Δ.

Proof: obvious.

Lemma 2
Let f be the flow at the end of a Δ-phase. Then the maximum flow is smaller than $\mathrm{val}(f) + m\Delta$.

Proof: less obvious, but simple:

Capacity Scaling

Lemma 1
There are $\lceil \log C \rceil$ iterations over Δ.

Proof: obvious.

Lemma 2
Let f be the flow at the end of a Δ-phase. Then the maximum flow is smaller than $\mathrm{val}(f) + m\Delta$.

Proof: less obvious, but simple:

▶ There must exist an s-t cut in $G_f(\Delta)$ of zero capacity.

Capacity Scaling

Lemma 1
There are $\lceil \log C \rceil$ iterations over Δ.

Proof: obvious.

Lemma 2
Let f be the flow at the end of a Δ-phase. Then the maximum flow is smaller than $\mathrm{val}(f) + m\Delta$.

Proof: less obvious, but simple:

- There must exist an s-t cut in $G_f(\Delta)$ of zero capacity.
- In G_f this cut can have capacity at most $m\Delta$.

Capacity Scaling

Lemma 1

There are $\lceil \log C \rceil$ iterations over Δ.

Proof: obvious.

Lemma 2

Let f be the flow at the end of a Δ-phase. Then the maximum flow is smaller than $\mathrm{val}(f) + m\Delta$.

Proof: less obvious, but simple:

▶ There must exist an s-t cut in $G_f(\Delta)$ of zero capacity.

▶ In G_f this cut can have capacity at most $m\Delta$.

▶ This gives me an upper bound on the flow that I can still add.

Capacity Scaling

Capacity Scaling

Lemma 3

There are at most $2m$ augmentations per scaling-phase.

Capacity Scaling

Lemma 3

There are at most $2m$ augmentations per scaling-phase.

Proof:

▶ Let f be the flow at the end of the previous phase.

Capacity Scaling

Lemma 3

There are at most $2m$ augmentations per scaling-phase.

Proof:

- ▶ Let f be the flow at the end of the previous phase.
- ▶ $\text{val}(f^*) \leq \text{val}(f) + 2m\Delta$

Capacity Scaling

Lemma 3

There are at most $2m$ augmentations per scaling-phase.

Proof:

- Let f be the flow at the end of the previous phase.
- $\text{val}(f^*) \leq \text{val}(f) + 2m\Delta$
- Each augmentation increases flow by Δ.

Capacity Scaling

Lemma 3

There are at most $2m$ augmentations per scaling-phase.

Proof:

- ▶ Let f be the flow at the end of the previous phase.
- ▶ $\text{val}(f^*) \leq \text{val}(f) + 2m\Delta$
- ▶ Each augmentation increases flow by Δ.

Theorem 4

We need $\mathcal{O}(m \log C)$ augmentations. The algorithm can be implemented in time $\mathcal{O}(m^2 \log C)$.