

# Repetition: Primal Dual for Set Cover

## Primal Relaxation:

$$\begin{array}{ll} \min & \sum_{i=1}^k w_i x_i \\ \text{s.t.} & \forall u \in U \quad \sum_{i:u \in S_i} x_i \geq 1 \\ & \forall i \in \{1, \dots, k\} \quad x_i \geq 0 \end{array}$$

## Dual Formulation:

$$\begin{array}{ll} \max & \sum_{u \in U} y_u \\ \text{s.t.} & \forall i \in \{1, \dots, k\} \quad \sum_{u:u \in S_i} y_u \leq w_i \\ & y_u \geq 0 \end{array}$$

# Repetition: Primal Dual for Set Cover

## Primal Relaxation:

$$\begin{array}{ll} \min & \sum_{i=1}^k w_i x_i \\ \text{s.t.} & \forall u \in U \quad \sum_{i:u \in S_i} x_i \geq 1 \\ & \forall i \in \{1, \dots, k\} \quad x_i \geq 0 \end{array}$$

## Dual Formulation:

$$\begin{array}{ll} \max & \sum_{u \in U} y_u \\ \text{s.t.} & \forall i \in \{1, \dots, k\} \quad \sum_{u:u \in S_i} y_u \leq w_i \\ & y_u \geq 0 \end{array}$$

# Repetition: Primal Dual for Set Cover

## Algorithm:

- ▶ Start with  $y = 0$  (feasible dual solution).  
Start with  $x = 0$  (integral primal solution that may be infeasible).
- ▶ While  $x$  not feasible

# Repetition: Primal Dual for Set Cover

## Algorithm:

- ▶ Start with  $y = 0$  (feasible dual solution).  
Start with  $x = 0$  (integral primal solution that may be infeasible).
- ▶ While  $x$  not feasible
  - ▶ Identify an element  $e$  that is not covered in current primal integral solution.
  - ▶ Increase dual variable  $y_e$  until a dual constraint becomes tight (maybe increase by 0!).
  - ▶ If this is the constraint for set  $S_j$  set  $x_j = 1$  (add this set to your solution).

# Repetition: Primal Dual for Set Cover

## Algorithm:

- ▶ Start with  $y = 0$  (feasible dual solution).  
Start with  $x = 0$  (integral primal solution that may be infeasible).
- ▶ While  $x$  not feasible
  - ▶ Identify an element  $e$  that is not covered in current primal integral solution.
  - ▶ Increase dual variable  $y_e$  until a dual constraint becomes tight (maybe increase by 0!).
  - ▶ If this is the constraint for set  $S_j$  set  $x_j = 1$  (add this set to your solution).

# Repetition: Primal Dual for Set Cover

## Algorithm:

- ▶ Start with  $y = 0$  (feasible dual solution).  
Start with  $x = 0$  (integral primal solution that may be infeasible).
- ▶ While  $x$  not feasible
  - ▶ Identify an element  $e$  that is not covered in current primal integral solution.
  - ▶ Increase dual variable  $y_e$  until a dual constraint becomes tight (maybe increase by 0!).
  - ▶ If this is the constraint for set  $S_j$  set  $x_j = 1$  (add this set to your solution).

# Repetition: Primal Dual for Set Cover

## Algorithm:

- ▶ Start with  $y = 0$  (feasible dual solution).  
Start with  $x = 0$  (integral primal solution that may be infeasible).
- ▶ While  $x$  not feasible
  - ▶ Identify an element  $e$  that is not covered in current primal integral solution.
  - ▶ Increase dual variable  $y_e$  until a dual constraint becomes tight (maybe increase by 0!).
  - ▶ If this is the constraint for set  $S_j$  set  $x_j = 1$  (add this set to your solution).

# Repetition: Primal Dual for Set Cover

**Analysis:**



# Repetition: Primal Dual for Set Cover

## Analysis:

- ▶ For every set  $S_j$  with  $x_j = 1$  we have

$$\sum_{e \in S_j} y_e = w_j$$

# Repetition: Primal Dual for Set Cover

## Analysis:

- ▶ For every set  $S_j$  with  $x_j = 1$  we have

$$\sum_{e \in S_j} y_e = w_j$$

- ▶ Hence our cost is

# Repetition: Primal Dual for Set Cover

## Analysis:

- ▶ For every set  $S_j$  with  $x_j = 1$  we have

$$\sum_{e \in S_j} y_e = w_j$$

- ▶ Hence our cost is

$$\sum_j w_j x_j$$

# Repetition: Primal Dual for Set Cover

## Analysis:

- ▶ For every set  $S_j$  with  $x_j = 1$  we have

$$\sum_{e \in S_j} y_e = w_j$$

- ▶ Hence our cost is

$$\sum_j w_j x_j = \sum_j \sum_{e \in S_j} y_e$$

# Repetition: Primal Dual for Set Cover

## Analysis:

- ▶ For every set  $S_j$  with  $x_j = 1$  we have

$$\sum_{e \in S_j} y_e = w_j$$

- ▶ Hence our cost is

$$\sum_j w_j x_j = \sum_j \sum_{e \in S_j} y_e = \sum_e |\{j : e \in S_j\}| \cdot y_e$$

# Repetition: Primal Dual for Set Cover

## Analysis:

- ▶ For every set  $S_j$  with  $x_j = 1$  we have

$$\sum_{e \in S_j} y_e = w_j$$

- ▶ Hence our cost is

$$\begin{aligned} \sum_j w_j x_j &= \sum_j \sum_{e \in S_j} y_e = \sum_e |\{j : e \in S_j\}| \cdot y_e \\ &\leq f \cdot \sum_e y_e \leq f \cdot \text{OPT} \end{aligned}$$

Note that the constructed pair of primal and dual solution fulfills **primal slackness conditions**.

Note that the constructed pair of primal and dual solution fulfills **primal slackness conditions**.

This means

$$x_j > 0 \Rightarrow \sum_{e \in S_j} y_e = w_j$$



Note that the constructed pair of primal and dual solution fulfills **primal slackness conditions**.

This means

$$x_j > 0 \Rightarrow \sum_{e \in S_j} y_e = w_j$$

If we would also fulfill **dual slackness conditions**

$$y_e > 0 \Rightarrow \sum_{j: e \in S_j} x_j = 1$$

then the solution would be **optimal!!!!**

We don't fulfill these constraint but we fulfill an approximate version:

We don't fulfill these constraint but we fulfill an approximate version:

$$y_e > 0 \Rightarrow 1 \leq \sum_{j:e \in S_j} x_j \leq f$$

We don't fulfill these constraint but we fulfill an approximate version:

$$y_e > 0 \Rightarrow 1 \leq \sum_{j:e \in S_j} x_j \leq f$$

This is sufficient to show that the solution is an  $f$ -approximation.

Suppose we have a primal/dual pair

$$\begin{array}{ll} \min & \sum_j c_j x_j \\ \text{s.t.} & \forall i \quad \sum_j a_{ij} x_j \geq b_i \\ & \forall j \quad x_j \geq 0 \end{array}$$

$$\begin{array}{ll} \max & \sum_i b_i y_i \\ \text{s.t.} & \forall j \quad \sum_i a_{ij} y_i \leq c_j \\ & \forall i \quad y_i \geq 0 \end{array}$$

Suppose we have a primal/dual pair

$$\begin{array}{ll} \min & \sum_j c_j x_j \\ \text{s.t.} & \forall i \quad \sum_j a_{ij} x_j \geq b_i \\ & \forall j \quad x_j \geq 0 \end{array}$$

$$\begin{array}{ll} \max & \sum_i b_i y_i \\ \text{s.t.} & \forall j \quad \sum_i a_{ij} y_i \leq c_j \\ & \forall i \quad y_i \geq 0 \end{array}$$

and solutions that fulfill approximate slackness conditions:

$$x_j > 0 \Rightarrow \sum_i a_{ij} y_i \geq \frac{1}{\alpha} c_j$$

$$y_i > 0 \Rightarrow \sum_j a_{ij} x_j \leq \beta b_i$$

Then

$$\sum_j c_j x_j$$

Then

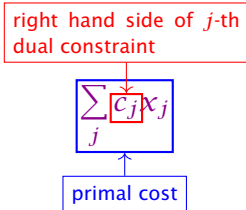
$$\sum_j c_j x_j$$

↑

primal cost



Then



Then

$$\boxed{\sum_j c_j x_j} \leq \alpha \sum_j \left( \sum_i a_{ij} y_i \right) x_j$$

↑  
primal cost

Then

$$\boxed{\sum_j c_j x_j} \leq \alpha \sum_j \left( \sum_i a_{ij} y_i \right) x_j$$

↑

$$\boxed{\text{primal cost}} = \alpha \sum_i \left( \sum_j a_{ij} x_j \right) y_i$$

Then

$$\begin{aligned} \boxed{\sum_j c_j x_j} &\leq \alpha \sum_j \left( \sum_i a_{ij} y_i \right) x_j \\ \uparrow \\ \boxed{\text{primal cost}} &= \alpha \sum_i \left( \sum_j a_{ij} x_j \right) y_i \\ &\leq \alpha \beta \cdot \sum_i b_i y_i \end{aligned}$$

Then

$$\begin{aligned} \boxed{\sum_j c_j x_j} &\leq \alpha \sum_j \left( \sum_i a_{ij} y_i \right) x_j \\ \text{primal cost} &= \alpha \sum_i \left( \sum_j a_{ij} x_j \right) y_i \\ &\leq \alpha \beta \cdot \boxed{\sum_i b_i y_i} \\ &\quad \uparrow \\ &\quad \text{dual objective} \end{aligned}$$

# Feedback Vertex Set for Undirected Graphs

- ▶ Given a graph  $G = (V, E)$  and non-negative weights  $w_v \geq 0$  for vertex  $v \in V$ .

# Feedback Vertex Set for Undirected Graphs

- ▶ Given a graph  $G = (V, E)$  and non-negative weights  $w_v \geq 0$  for vertex  $v \in V$ .
- ▶ Choose a minimum cost subset of vertices s.t. every cycle contains at least one vertex.

We can encode this as an instance of Set Cover

- ▶ Each vertex can be viewed as a set that contains some cycles.



We can encode this as an instance of Set Cover

- ▶ Each vertex can be viewed as a set that contains some cycles.
- ▶ However, this encoding gives a Set Cover instance of non-polynomial size.

We can encode this as an instance of Set Cover

- ▶ Each vertex can be viewed as a set that contains some cycles.
- ▶ However, this encoding gives a Set Cover instance of non-polynomial size.
- ▶ The  $O(\log n)$ -approximation for Set Cover does not help us to get a good solution.

Let  $\mathcal{C}$  denote the set of all cycles (where a cycle is identified by its set of vertices)

Let  $\mathcal{C}$  denote the set of all cycles (where a cycle is identified by its set of vertices)

### Primal Relaxation:

$$\begin{array}{ll} \min & \sum_v w_v x_v \\ \text{s.t.} & \forall C \in \mathcal{C} \quad \sum_{v \in C} x_v \geq 1 \\ & \forall v \quad x_v \geq 0 \end{array}$$

### Dual Formulation:

$$\begin{array}{ll} \max & \sum_{C \in \mathcal{C}} y_C \\ \text{s.t.} & \forall v \in V \quad \sum_{C: v \in C} y_C \leq w_v \\ & \forall C \quad y_C \geq 0 \end{array}$$

If we perform the previous dual technique for Set Cover we get the following:

- ▶ Start with  $x = 0$  and  $y = 0$

If we perform the previous dual technique for Set Cover we get the following:

- ▶ Start with  $x = 0$  and  $y = 0$
- ▶ While there is a cycle  $C$  that is not covered (does not contain a chosen vertex).

If we perform the previous dual technique for Set Cover we get the following:

- ▶ Start with  $x = 0$  and  $y = 0$
- ▶ While there is a cycle  $C$  that is not covered (does not contain a chosen vertex).
  - ▶ Increase  $y_C$  until dual constraint for some vertex  $v$  becomes tight.

If we perform the previous dual technique for Set Cover we get the following:

- ▶ Start with  $x = 0$  and  $y = 0$
- ▶ While there is a cycle  $C$  that is not covered (does not contain a chosen vertex).
  - ▶ Increase  $y_C$  until dual constraint for some vertex  $v$  becomes tight.
  - ▶ set  $x_v = 1$ .



Then

$$\sum_v w_v x_v$$

Then

$$\sum_v w_v x_v = \sum_v \sum_{C:v \in C} y_C x_v$$

Then

$$\begin{aligned}\sum_v w_v x_v &= \sum_v \sum_{C:v \in C} y_C x_v \\ &= \sum_{v \in S} \sum_{C:v \in C} y_C\end{aligned}$$

where  $S$  is the set of vertices we choose.

Then

$$\begin{aligned}\sum_v w_v x_v &= \sum_v \sum_{C:v \in C} y_C x_v \\ &= \sum_{v \in S} \sum_{C:v \in C} y_C \\ &= \sum_C |S \cap C| \cdot y_C\end{aligned}$$

where  $S$  is the set of vertices we choose.

Then

$$\begin{aligned}\sum_v w_v x_v &= \sum_v \sum_{C:v \in C} y_C x_v \\ &= \sum_{v \in S} \sum_{C:v \in C} y_C \\ &= \sum_C |S \cap C| \cdot y_C\end{aligned}$$

where  $S$  is the set of vertices we choose.

If every cycle is short we get a good approximation ratio, but this is unrealistic.

### Algorithm 1 FeedbackVertexSet

- 1:  $y \leftarrow 0$
- 2:  $x \leftarrow 0$
- 3: **while** exists cycle  $C$  in  $G$  **do**
- 4:     increase  $y_C$  until there is  $v \in C$  s.t.  $\sum_{C:v \in C} y_C = w_v$
- 5:      $x_v = 1$
- 6:     remove  $v$  from  $G$
- 7:     repeatedly remove vertices of degree 1 from  $G$

**Idea:**

Always choose a short cycle that is not covered. If we always find a cycle of length at most  $\alpha$  we get an  $\alpha$ -approximation.

**Idea:**

Always choose a short cycle that is not covered. If we always find a cycle of length at most  $\alpha$  we get an  $\alpha$ -approximation.

**Observation:**

For any path  $P$  of vertices of degree 2 in  $G$  the algorithm chooses at most one vertex from  $P$ .



**Observation:**

If we always choose a cycle for which the number of vertices of degree at least 3 is at most  $\alpha$  we get a  $2\alpha$ -approximation.

## Observation:

If we always choose a cycle for which the number of vertices of degree at least 3 is at most  $\alpha$  we get a  $2\alpha$ -approximation.

## Theorem 92

*In any graph with no vertices of degree 1, there always exists a cycle that has at most  $\mathcal{O}(\log n)$  vertices of degree 3 or more. We can find such a cycle in linear time.*

This means we have

$$y_C > 0 \Rightarrow |S \cap C| \leq \mathcal{O}(\log n) .$$

# Primal Dual for Shortest Path

Given a graph  $G = (V, E)$  with two nodes  $s, t \in V$  and edge-weights  $c : E \rightarrow \mathbb{R}^+$  find a shortest path between  $s$  and  $t$  w.r.t. edge-weights  $c$ .

$$\begin{array}{ll} \min & \sum_e c(e)x_e \\ \text{s.t.} & \forall S \in \mathcal{S} \quad \sum_{e \in \delta(S)} x_e \geq 1 \\ & \forall e \in E \quad x_e \in \{0, 1\} \end{array}$$

Here  $\delta(S)$  denotes the set of edges with exactly one end-point in  $S$ , and  $\mathcal{S} = \{S \subseteq V : s \in S, t \notin S\}$ .

# Primal Dual for Shortest Path

Given a graph  $G = (V, E)$  with two nodes  $s, t \in V$  and edge-weights  $c : E \rightarrow \mathbb{R}^+$  find a shortest path between  $s$  and  $t$  w.r.t. edge-weights  $c$ .

$$\begin{array}{ll} \min & \sum_e c(e) x_e \\ \text{s.t.} & \forall S \in \mathcal{S} \quad \sum_{e: \delta(S)} x_e \geq 1 \\ & \forall e \in E \quad x_e \in \{0, 1\} \end{array}$$

Here  $\delta(S)$  denotes the set of edges with exactly one end-point in  $S$ , and  $\mathcal{S} = \{S \subseteq V : s \in S, t \notin S\}$ .

# Primal Dual for Shortest Path

**The Dual:**

$$\begin{array}{ll} \max & \sum_S y_S \\ \text{s.t.} & \forall e \in E \quad \sum_{S:e \in \delta(S)} y_S \leq c(e) \\ & \forall S \in \mathcal{S} \quad y_S \geq 0 \end{array}$$

Here  $\delta(S)$  denotes the set of edges with exactly one end-point in  $S$ , and  $\mathcal{S} = \{S \subseteq V : s \in S, t \notin S\}$ .

# Primal Dual for Shortest Path

The Dual:

$$\begin{array}{ll} \max & \sum_S y_S \\ \text{s.t.} & \forall e \in E \quad \sum_{S:e \in \delta(S)} y_S \leq c(e) \\ & \forall S \in \mathcal{S} \quad y_S \geq 0 \end{array}$$

Here  $\delta(S)$  denotes the set of edges with exactly one end-point in  $S$ , and  $\mathcal{S} = \{S \subseteq V : s \in S, t \notin S\}$ .

# Primal Dual for Shortest Path

We can interpret the value  $y_S$  as the width of a moat surrounding the set  $S$ .

Each set can have its own moat but all moats must be disjoint.

An edge cannot be shorter than all the moats that it has to cross.

# Primal Dual for Shortest Path

We can interpret the value  $y_S$  as the width of a moat surrounding the set  $S$ .

Each set can have its own moat but all moats must be disjoint.

An edge cannot be shorter than all the moats that it has to cross.



# Primal Dual for Shortest Path

We can interpret the value  $y_S$  as the width of a moat surrounding the set  $S$ .

Each set can have its own moat but all moats must be disjoint.

An edge cannot be shorter than all the moats that it has to cross.

# Primal Dual for Shortest Path

We can interpret the value  $y_S$  as the width of a moat surrounding the set  $S$ .

Each set can have its own moat but all moats must be disjoint.

An edge cannot be shorter than all the moats that it has to cross.

### Algorithm 1 PrimalDualShortestPath

- 1:  $\gamma \leftarrow 0$
- 2:  $F \leftarrow \emptyset$
- 3: **while** there is no  $s$ - $t$  path in  $(V, F)$  **do**
- 4:     Let  $C$  be the connected component of  $(V, F)$  containing  $s$
- 5:     Increase  $\gamma_C$  until there is an edge  $e' \in \delta(C)$  such that  $\sum_{S:e' \in \delta(S)} \gamma_S = c(e')$ .
- 6:      $F \leftarrow F \cup \{e'\}$
- 7: **Let  $P$  be an  $s$ - $t$  path in  $(V, F)$**
- 8: **return  $P$**

## Lemma 93

*At each point in time the set  $F$  forms a tree.*

Proof:

Assume that  $F$  contains a cycle. Then the cycle is contained in  $F$  and  $F$  is not a tree. This contradicts the fact that  $F$  is a forest. Therefore,  $F$  is a tree.

## Lemma 93

*At each point in time the set  $F$  forms a tree.*

### Proof:

- ▶ In each iteration we take the current connected component from  $(V, F)$  that contains  $s$  (call this component  $C$ ) and add some edge from  $\delta(C)$  to  $F$ .
- ▶ Since, at most one end-point of the new edge is in  $C$  the edge cannot close a cycle.

## Lemma 93

*At each point in time the set  $F$  forms a tree.*

### Proof:

- ▶ In each iteration we take the current connected component from  $(V, F)$  that contains  $s$  (call this component  $C$ ) and add some edge from  $\delta(C)$  to  $F$ .
- ▶ Since, at most one end-point of the new edge is in  $C$  the edge cannot close a cycle.

$$\sum_{e \in P} c(e)$$

$$\sum_{e \in P} c(e) = \sum_{e \in P} \sum_{S: e \in \delta(S)} y_S$$



$$\begin{aligned}\sum_{e \in P} c(e) &= \sum_{e \in P} \sum_{S: e \in \delta(S)} \gamma_S \\ &= \sum_{S: s \in S, t \notin S} |P \cap \delta(S)| \cdot \gamma_S .\end{aligned}$$

$$\begin{aligned}\sum_{e \in P} c(e) &= \sum_{e \in P} \sum_{S: e \in \delta(S)} y_S \\ &= \sum_{S: s \in S, t \notin S} |P \cap \delta(S)| \cdot y_S .\end{aligned}$$

If we can show that  $y_S > 0$  implies  $|P \cap \delta(S)| = 1$  gives

$$\sum_{e \in P} c(e) = \sum_S y_S \leq \text{OPT}$$

by weak duality.

$$\begin{aligned}\sum_{e \in P} c(e) &= \sum_{e \in P} \sum_{S: e \in \delta(S)} y_S \\ &= \sum_{S: s \in S, t \notin S} |P \cap \delta(S)| \cdot y_S .\end{aligned}$$

If we can show that  $y_S > 0$  implies  $|P \cap \delta(S)| = 1$  gives

$$\sum_{e \in P} c(e) = \sum_S y_S \leq \text{OPT}$$

by weak duality.

Hence, we find a shortest path.

If  $\delta(S)$  contains two edges from  $P$  then there must exist a subpath  $P'$  of  $P$  that starts and ends with a vertex from  $S$  (and all interior vertices are not in  $S$ ).

When we increased  $y_S$ ,  $S$  was a connected component of the set of edges  $F'$  that we had chosen till this point.

$F' \cup P'$  contains a cycle. Hence, also the final set of edges contains a cycle.

This is a contradiction.

If  $\delta(S)$  contains two edges from  $P$  then there must exist a subpath  $P'$  of  $P$  that starts and ends with a vertex from  $S$  (and all interior vertices are not in  $S$ ).

When we increased  $y_S$ ,  $S$  was a connected component of the set of edges  $F'$  that we had chosen till this point.

$F' \cup P'$  contains a cycle. Hence, also the final set of edges contains a cycle.

This is a contradiction.

If  $\delta(S)$  contains two edges from  $P$  then there must exist a subpath  $P'$  of  $P$  that starts and ends with a vertex from  $S$  (and all interior vertices are not in  $S$ ).

When we increased  $y_S$ ,  $S$  was a connected component of the set of edges  $F'$  that we had chosen till this point.

$F' \cup P'$  contains a cycle. Hence, also the final set of edges contains a cycle.

This is a contradiction.

If  $\delta(S)$  contains two edges from  $P$  then there must exist a subpath  $P'$  of  $P$  that starts and ends with a vertex from  $S$  (and all interior vertices are not in  $S$ ).

When we increased  $y_S$ ,  $S$  was a connected component of the set of edges  $F'$  that we had chosen till this point.

$F' \cup P'$  contains a cycle. Hence, also the final set of edges contains a cycle.

This is a contradiction.

If  $\delta(S)$  contains two edges from  $P$  then there must exist a subpath  $P'$  of  $P$  that starts and ends with a vertex from  $S$  (and all interior vertices are not in  $S$ ).

When we increased  $y_S$ ,  $S$  was a connected component of the set of edges  $F'$  that we had chosen till this point.

$F' \cup P'$  contains a cycle. Hence, also the final set of edges contains a cycle.

This is a contradiction.



## Steiner Forest Problem:

Given a graph  $G = (V, E)$ , together with source-target pairs  $s_i, t_i$ ,  $i = 1, \dots, k$ , and a cost function  $c : E \rightarrow \mathbb{R}^+$  on the edges. Find a subset  $F \subseteq E$  of the edges such that for every  $i \in \{1, \dots, k\}$  there is a path between  $s_i$  and  $t_i$  only using edges in  $F$ .

$$\begin{array}{ll} \min & \sum_e c(e)x_e \\ \text{s.t.} & \forall S \subseteq V : S \in S_i \text{ for some } i \quad \sum_{e \in \delta(S)} x_e \geq 1 \\ & \forall e \in E \quad x_e \in \{0, 1\} \end{array}$$

Here  $S_i$  contains all sets  $S$  such that  $s_i \in S$  and  $t_i \notin S$ .

## Steiner Forest Problem:

Given a graph  $G = (V, E)$ , together with source-target pairs  $s_i, t_i$ ,  $i = 1, \dots, k$ , and a cost function  $c : E \rightarrow \mathbb{R}^+$  on the edges. Find a subset  $F \subseteq E$  of the edges such that for every  $i \in \{1, \dots, k\}$  there is a path between  $s_i$  and  $t_i$  only using edges in  $F$ .

$$\begin{array}{ll} \min & \sum_e c(e)x_e \\ \text{s.t.} & \forall S \subseteq V : S \in S_i \text{ for some } i \quad \sum_{e \in \delta(S)} x_e \geq 1 \\ & \forall e \in E \quad x_e \in \{0, 1\} \end{array}$$

Here  $S_i$  contains all sets  $S$  such that  $s_i \in S$  and  $t_i \notin S$ .

## Steiner Forest Problem:

Given a graph  $G = (V, E)$ , together with source-target pairs  $s_i, t_i$ ,  $i = 1, \dots, k$ , and a cost function  $c : E \rightarrow \mathbb{R}^+$  on the edges. Find a subset  $F \subseteq E$  of the edges such that for every  $i \in \{1, \dots, k\}$  there is a path between  $s_i$  and  $t_i$  only using edges in  $F$ .

$$\begin{array}{ll} \min & \sum_e c(e)x_e \\ \text{s.t.} & \forall S \subseteq V : S \in S_i \text{ for some } i \quad \sum_{e \in \delta(S)} x_e \geq 1 \\ & \forall e \in E \quad x_e \in \{0, 1\} \end{array}$$

Here  $S_i$  contains all sets  $S$  such that  $s_i \in S$  and  $t_i \notin S$ .

$$\begin{array}{ll}
 \max & \sum_{S: \exists i \text{ s.t. } S \in S_i} \mathcal{Y}_S \\
 \text{s.t.} & \forall e \in E \quad \sum_{S: e \in \delta(S)} \mathcal{Y}_S \leq c(e) \\
 & \mathcal{Y}_S \geq 0
 \end{array}$$

The difference to the dual of the shortest path problem is that we have many more variables (sets for which we can generate a moat of non-zero width).

### Algorithm 1 FirstTry

- 1:  $\gamma \leftarrow 0$
- 2:  $F \leftarrow \emptyset$
- 3: **while** not all  $s_i$ - $t_i$  pairs connected in  $F$  **do**
- 4:     Let  $C$  be some connected component of  $(V, F)$  such that  $|C \cap \{s_i, t_i\}| = 1$  for some  $i$ .
- 5:     Increase  $\gamma_C$  until there is an edge  $e' \in \delta(C)$  s.t.  
        $\sum_{S \in \mathcal{S}_i: e' \in \delta(S)} \gamma_S = c_{e'}$
- 6:      $F \leftarrow F \cup \{e'\}$
- 7: **return**  $\bigcup_i P_i$

$$\sum_{e \in F} c(e)$$

$$\sum_{e \in F} c(e) = \sum_{e \in F} \sum_{S: e \in \delta(S)} y_S$$

$$\sum_{e \in F} c(e) = \sum_{e \in F} \sum_{S: e \in \delta(S)} \gamma_S = \sum_S |\delta(S) \cap F| \cdot \gamma_S .$$



$$\sum_{e \in F} c(e) = \sum_{e \in F} \sum_{S: e \in \delta(S)} y_S = \sum_S |\delta(S) \cap F| \cdot y_S .$$

$$\sum_{e \in F} c(e) = \sum_{e \in F} \sum_{S: e \in \delta(S)} \gamma_S = \sum_S |\delta(S) \cap F| \cdot \gamma_S .$$

If we show that  $\gamma_S > 0$  implies that  $|\delta(S) \cap F| \leq \alpha$  we are in good shape.

However, this is not true:

- ▶ Take a complete graph on  $k + 1$  vertices  $v_0, v_1, \dots, v_k$ .

$$\sum_{e \in F} c(e) = \sum_{e \in F} \sum_{S: e \in \delta(S)} \gamma_S = \sum_S |\delta(S) \cap F| \cdot \gamma_S .$$

If we show that  $\gamma_S > 0$  implies that  $|\delta(S) \cap F| \leq \alpha$  we are in good shape.

However, this is not true:

- ▶ Take a complete graph on  $k + 1$  vertices  $v_0, v_1, \dots, v_k$ .
- ▶ The  $i$ -th pair is  $v_0 - v_i$ .

$$\sum_{e \in F} c(e) = \sum_{e \in F} \sum_{S: e \in \delta(S)} y_S = \sum_S |\delta(S) \cap F| \cdot y_S .$$

If we show that  $y_S > 0$  implies that  $|\delta(S) \cap F| \leq \alpha$  we are in good shape.

However, this is not true:

- ▶ Take a complete graph on  $k + 1$  vertices  $v_0, v_1, \dots, v_k$ .
- ▶ The  $i$ -th pair is  $v_0 - v_i$ .
- ▶ The first component  $C$  could be  $\{v_0\}$ .

$$\sum_{e \in F} c(e) = \sum_{e \in F} \sum_{S: e \in \delta(S)} y_S = \sum_S |\delta(S) \cap F| \cdot y_S .$$

If we show that  $y_S > 0$  implies that  $|\delta(S) \cap F| \leq \alpha$  we are in good shape.

However, this is not true:

- ▶ Take a complete graph on  $k + 1$  vertices  $v_0, v_1, \dots, v_k$ .
- ▶ The  $i$ -th pair is  $v_0 - v_i$ .
- ▶ The first component  $C$  could be  $\{v_0\}$ .
- ▶ We only set  $y_{\{v_0\}} = 1$ . All other dual variables stay 0.

$$\sum_{e \in F} c(e) = \sum_{e \in F} \sum_{S: e \in \delta(S)} y_S = \sum_S |\delta(S) \cap F| \cdot y_S .$$

If we show that  $y_S > 0$  implies that  $|\delta(S) \cap F| \leq \alpha$  we are in good shape.

However, this is not true:

- ▶ Take a complete graph on  $k + 1$  vertices  $v_0, v_1, \dots, v_k$ .
- ▶ The  $i$ -th pair is  $v_0 - v_i$ .
- ▶ The first component  $C$  could be  $\{v_0\}$ .
- ▶ We only set  $y_{\{v_0\}} = 1$ . All other dual variables stay 0.
- ▶ The final set  $F$  contains all edges  $\{v_0, v_i\}$ ,  $i = 1, \dots, k$ .

$$\sum_{e \in F} c(e) = \sum_{e \in F} \sum_{S: e \in \delta(S)} y_S = \sum_S |\delta(S) \cap F| \cdot y_S .$$

If we show that  $y_S > 0$  implies that  $|\delta(S) \cap F| \leq \alpha$  we are in good shape.

However, this is not true:

- ▶ Take a complete graph on  $k + 1$  vertices  $v_0, v_1, \dots, v_k$ .
- ▶ The  $i$ -th pair is  $v_0 - v_i$ .
- ▶ The first component  $C$  could be  $\{v_0\}$ .
- ▶ We only set  $y_{\{v_0\}} = 1$ . All other dual variables stay 0.
- ▶ The final set  $F$  contains all edges  $\{v_0, v_i\}$ ,  $i = 1, \dots, k$ .
- ▶  $y_{\{v_0\}} > 0$  but  $|\delta(\{v_0\}) \cap F| = k$ .

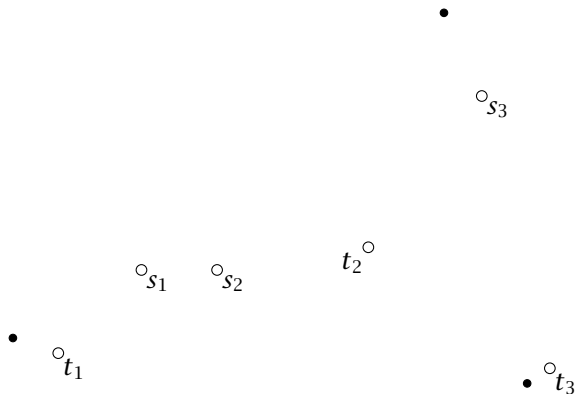
## Algorithm 1 SecondTry

```
1:  $y \leftarrow 0; F \leftarrow \emptyset; \ell \leftarrow 0$ 
2: while not all  $s_i-t_i$  pairs connected in  $F$  do
3:    $\ell \leftarrow \ell + 1$ 
4:   Let  $\mathfrak{C}$  be set of all connected components  $C$  of  $(V, F)$ 
     such that  $|C \cap \{s_i, t_i\}| = 1$  for some  $i$ .
5:   Increase  $y_C$  for all  $C \in \mathfrak{C}$  uniformly until for some edge
      $e_\ell \in \delta(C')$ ,  $C' \in \mathfrak{C}$  s.t.  $\sum_{S: e_\ell \in \delta(S)} y_S = c_{e_\ell}$ 
6:    $F \leftarrow F \cup \{e_\ell\}$ 
7:  $F' \leftarrow F$ 
8: for  $k \leftarrow \ell$  downto 1 do // reverse deletion
9:   if  $F' - e_k$  is feasible solution then
10:    remove  $e_k$  from  $F'$ 
11: return  $F'$ 
```

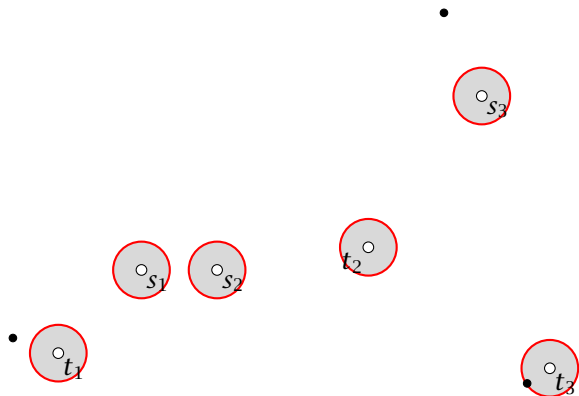


The reverse deletion step is not strictly necessary this way. It would also be sufficient to simply delete all unnecessary edges in any order.

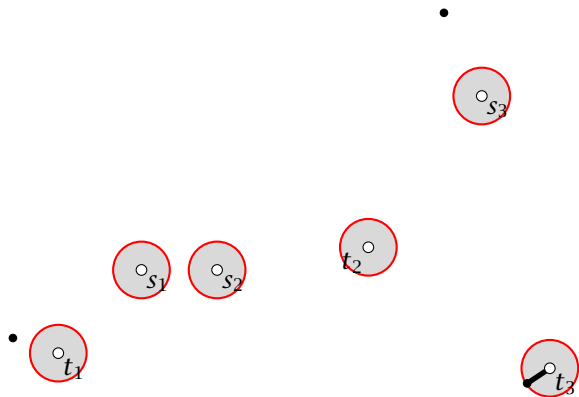
# Example



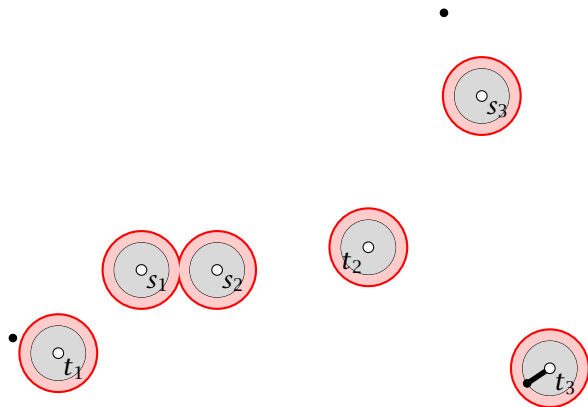
# Example



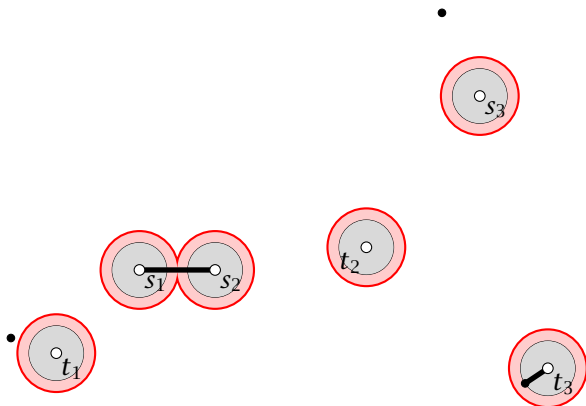
# Example



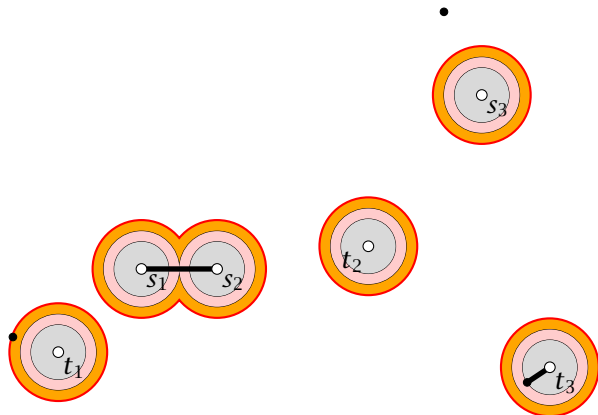
# Example



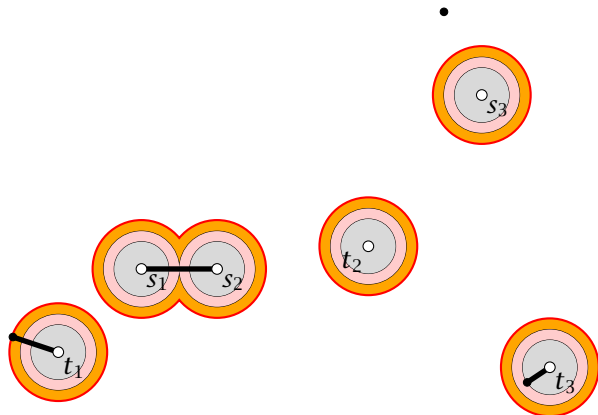
# Example



# Example

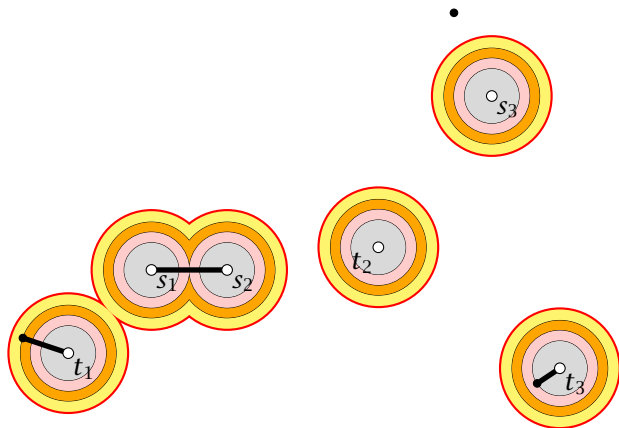


# Example

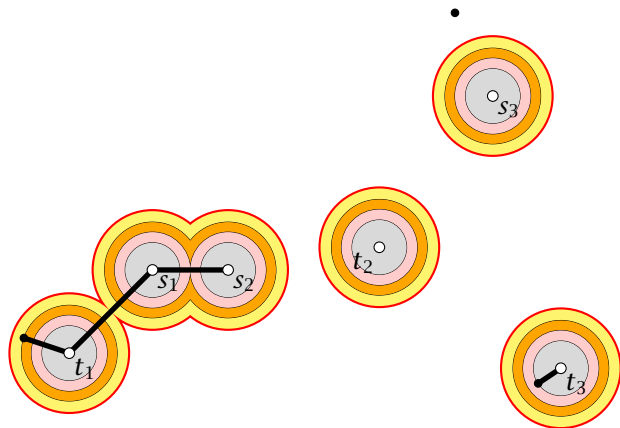




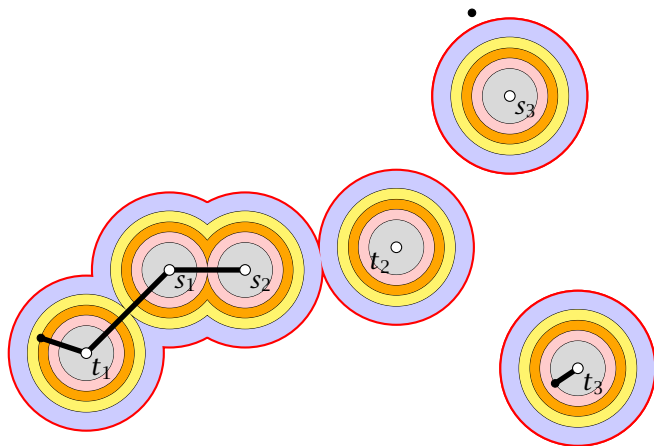
# Example



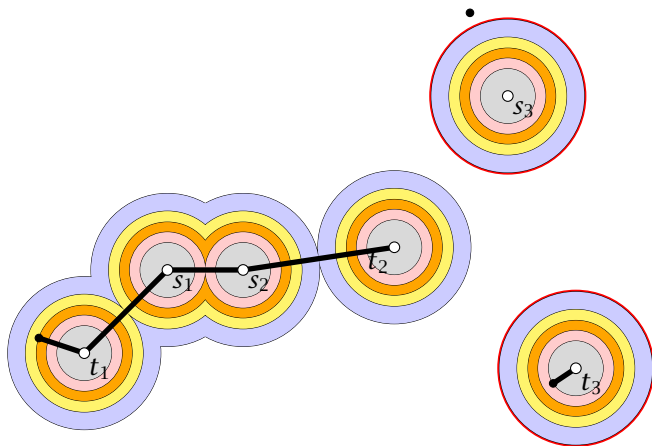
# Example



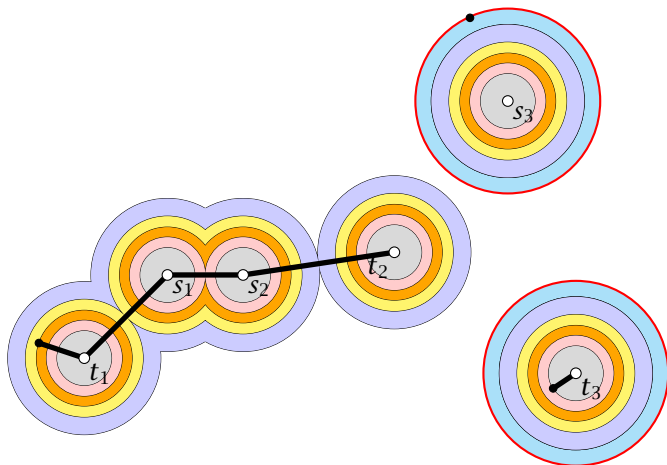
# Example



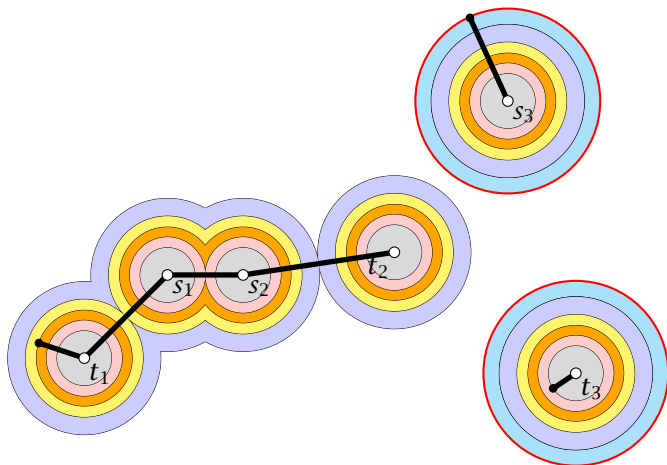
# Example



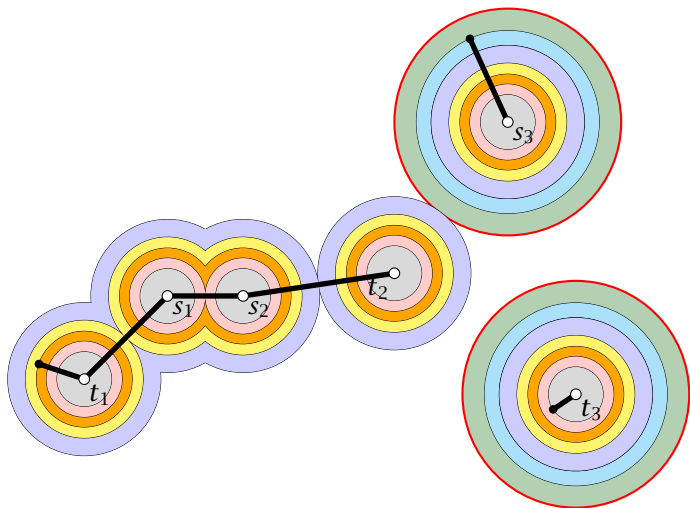
# Example



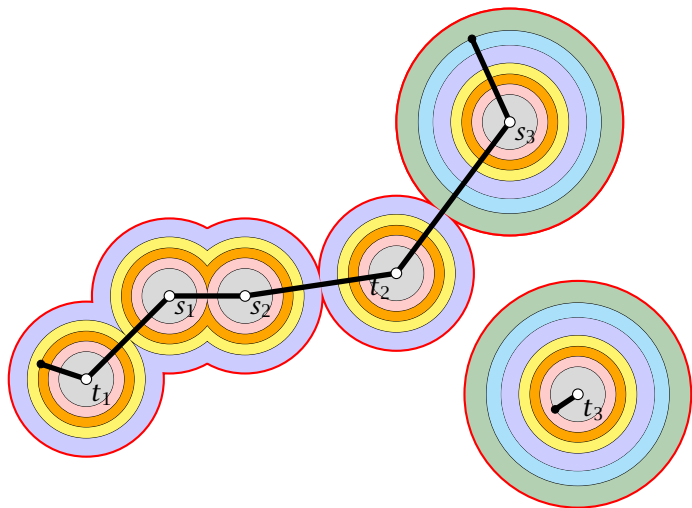
# Example



# Example

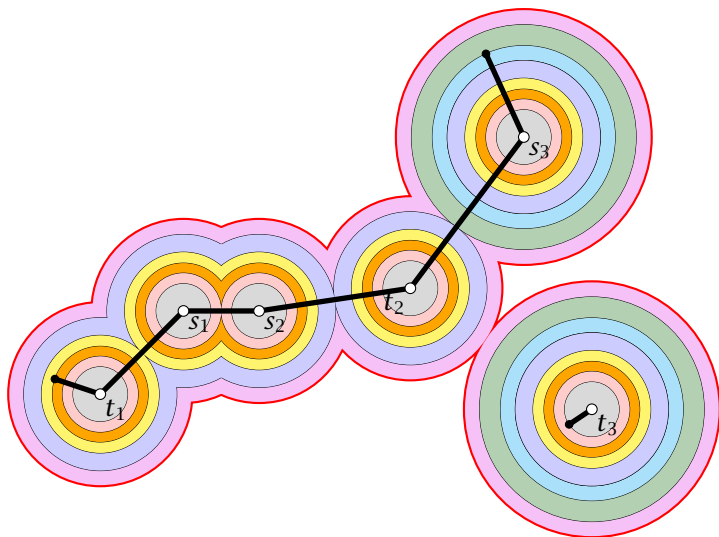


# Example

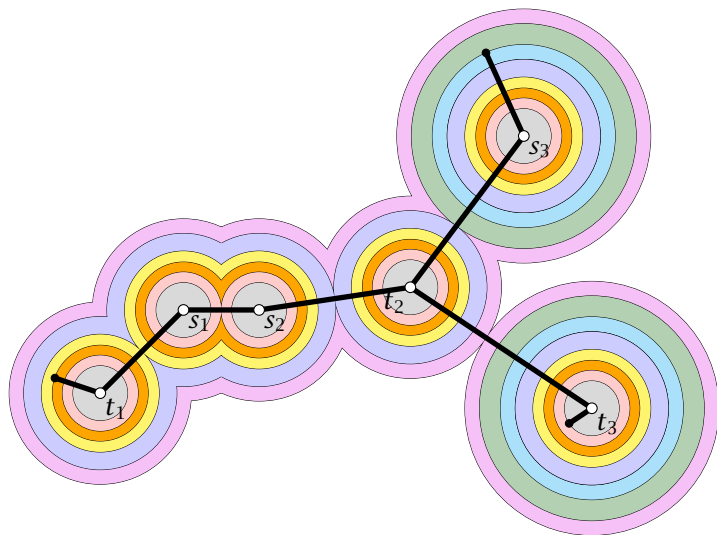




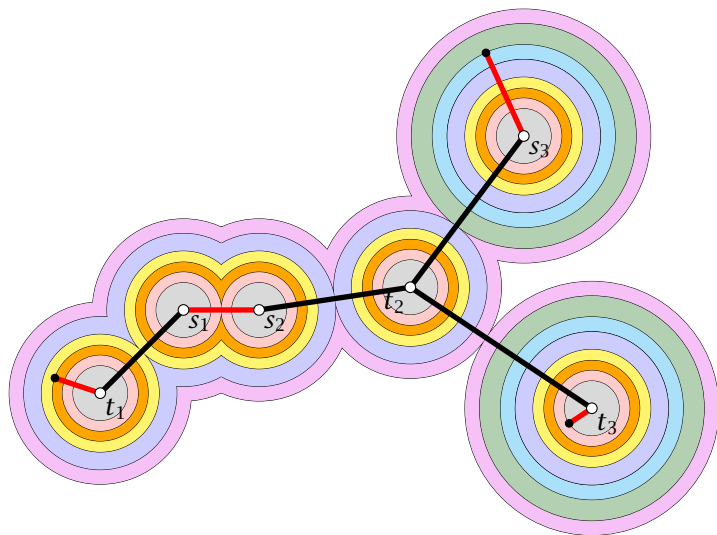
# Example



# Example



# Example



## Lemma 94

For any  $\mathcal{C}$  in any iteration of the algorithm

$$\sum_{C \in \mathcal{C}} |\delta(C) \cap F'| \leq 2|\mathcal{C}|$$

This means that the number of times a moat from  $\mathcal{C}$  is crossed in the final solution is at most twice the number of moats.

**Proof:** later...

$$\sum_{e \in F'} c_e = \sum_{e \in F'} \sum_{S: e \in \delta(S)} \gamma_S = \sum_S |F' \cap \delta(S)| \cdot \gamma_S.$$

We want to show that

$$\sum_S |F' \cap \delta(S)| \cdot \gamma_S \leq 2 \sum_S \gamma_S$$

At each iteration, the increase of the left-hand side is

at most  $\gamma_S$  (because  $|F' \cap \delta(S)|$  increases by at most 1).

At the same time, the increase of the right-hand side is  $2\gamma_S$ .

Hence, by the previous lemma, the inequality holds after  $k$  iterations if it holds in the beginning of the iteration.

$$\sum_{e \in F'} c_e = \sum_{e \in F'} \sum_{S: e \in \delta(S)} \gamma_S = \sum_S |F' \cap \delta(S)| \cdot \gamma_S.$$

We want to show that

$$\sum_S |F' \cap \delta(S)| \cdot \gamma_S \leq 2 \sum_S \gamma_S$$

At each iteration, the increase of the left-hand side is

at most the increase of the right-hand side times 2.

Since the increase of the right-hand side is  $\leq \gamma_S$ ,

we conclude, by the previous lemma, the inequality holds after  $k$  iterations if it holds in the beginning of the iteration.

$$\sum_{e \in F'} c_e = \sum_{e \in F'} \sum_{S: e \in \delta(S)} \gamma_S = \sum_S |F' \cap \delta(S)| \cdot \gamma_S .$$

We want to show that

$$\sum_S |F' \cap \delta(S)| \cdot \gamma_S \leq 2 \sum_S \gamma_S$$

Let us consider the increase of the left-hand side if we

add an edge  $e \in F'$  to  $F'$  (i.e.  $e \in F' \setminus F$ ).

Let  $S_1, \dots, S_k$  be the components of  $\delta(e)$ .

The increase of the right-hand side is  $\sum_{i=1}^k \gamma_{S_i}$ .

Since, by the previous lemma, the inequality holds after the

addition of  $e$ , it holds in the beginning of the operation.

$$\sum_{e \in F'} c_e = \sum_{e \in F'} \sum_{S: e \in \delta(S)} y_S = \sum_S |F' \cap \delta(S)| \cdot y_S .$$

We want to show that

$$\sum_S |F' \cap \delta(S)| \cdot y_S \leq 2 \sum_S y_S$$



$$\sum_{e \in F'} c_e = \sum_{e \in F'} \sum_{S: e \in \delta(S)} y_S = \sum_S |F' \cap \delta(S)| \cdot y_S .$$

We want to show that

$$\sum_S |F' \cap \delta(S)| \cdot y_S \leq 2 \sum_S y_S$$

- ▶ In the  $i$ -th iteration the increase of the left-hand side is

$$\epsilon \sum_{C \in \mathcal{C}} |F' \cap \delta(C)|$$

and the increase of the right hand side is  $2\epsilon|\mathcal{C}|$ .

- ▶ Hence, by the previous lemma the inequality holds after the iteration if it holds in the beginning of the iteration.

$$\sum_{e \in F'} c_e = \sum_{e \in F'} \sum_{S: e \in \delta(S)} y_S = \sum_S |F' \cap \delta(S)| \cdot y_S .$$

We want to show that

$$\sum_S |F' \cap \delta(S)| \cdot y_S \leq 2 \sum_S y_S$$

- ▶ In the  $i$ -th iteration the increase of the left-hand side is

$$\epsilon \sum_{C \in \mathcal{C}} |F' \cap \delta(C)|$$

and the increase of the right hand side is  $2\epsilon|\mathcal{C}|$ .

- ▶ Hence, by the previous lemma the inequality holds after the iteration if it holds in the beginning of the iteration.

## Lemma 95

For any set of connected components  $\mathcal{C}$  in any iteration of the algorithm

$$\sum_{C \in \mathcal{C}} |\delta(C) \cap F'| \leq 2|\mathcal{C}|$$

Proof:

## Lemma 95

For any set of connected components  $\mathcal{C}$  in any iteration of the algorithm

$$\sum_{C \in \mathcal{C}} |\delta(C) \cap F'| \leq 2|\mathcal{C}|$$

### Proof:

- ▶ At any point during the algorithm the set of edges forms a forest (why?).
- ▶ Fix iteration  $i$ . Let  $F_i$  be the set of edges in  $F$  at the beginning of the iteration.
- ▶ Let  $H = F' - F_i$ .
- ▶ All edges in  $H$  are necessary for the solution.

## Lemma 95

For any set of connected components  $\mathcal{C}$  in any iteration of the algorithm

$$\sum_{C \in \mathcal{C}} |\delta(C) \cap F'| \leq 2|\mathcal{C}|$$

### Proof:

- ▶ At any point during the algorithm the set of edges forms a forest (why?).
- ▶ Fix iteration  $i$ . Let  $F_i$  be the set of edges in  $F$  at the beginning of the iteration.
- ▶ Let  $H = F' - F_i$ .
- ▶ All edges in  $H$  are necessary for the solution.

## Lemma 95

For any set of connected components  $\mathcal{C}$  in any iteration of the algorithm

$$\sum_{C \in \mathcal{C}} |\delta(C) \cap F'| \leq 2|\mathcal{C}|$$

### Proof:

- ▶ At any point during the algorithm the set of edges forms a forest (why?).
- ▶ Fix iteration  $i$ . Let  $F_i$  be the set of edges in  $F$  at the beginning of the iteration.
- ▶ Let  $H = F' - F_i$ .
- ▶ All edges in  $H$  are necessary for the solution.

## Lemma 95

For any set of connected components  $\mathcal{C}$  in any iteration of the algorithm

$$\sum_{C \in \mathcal{C}} |\delta(C) \cap F'| \leq 2|\mathcal{C}|$$

### Proof:

- ▶ At any point during the algorithm the set of edges forms a forest (why?).
- ▶ Fix iteration  $i$ . Let  $F_i$  be the set of edges in  $F$  at the beginning of the iteration.
- ▶ Let  $H = F' - F_i$ .
- ▶ All edges in  $H$  are necessary for the solution.

- ▶ Contract all edges in  $F_i$  into single vertices  $V'$ .
- ▶ We can consider the forest  $H$  on the set of vertices  $V'$ .
- ▶ Let  $\deg(v)$  be the degree of a vertex  $v \in V'$  within this forest.
- ▶ Color a vertex  $v \in V'$  red if it corresponds to a component from  $\mathbb{C}$  (an active component). Otw. color it blue. (Let  $B$  the set of blue vertices (with non-zero degree) and  $R$  the set of red vertices)
- ▶ We have

$$\sum_{v \in R} \deg(v) \geq \sum_{C \in \mathbb{C}} |\delta(C) \cap F'| \stackrel{?}{\leq} 2|\mathbb{C}| = 2|R|$$



- ▶ Contract all edges in  $F_i$  into single vertices  $V'$ .
- ▶ We can consider the forest  $H$  on the set of vertices  $V'$ .
- ▶ Let  $\deg(v)$  be the degree of a vertex  $v \in V'$  within this forest.
- ▶ Color a vertex  $v \in V'$  red if it corresponds to a component from  $\mathbb{C}$  (an active component). Otw. color it blue. (Let  $B$  the set of blue vertices (with non-zero degree) and  $R$  the set of red vertices)
- ▶ We have

$$\sum_{v \in R} \deg(v) \geq \sum_{C \in \mathbb{C}} |\delta(C) \cap V'| \stackrel{?}{\leq} 2|\mathbb{C}| = 2|R|$$

- ▶ Contract all edges in  $F_i$  into single vertices  $V'$ .
- ▶ We can consider the forest  $H$  on the set of vertices  $V'$ .
- ▶ Let  $\deg(v)$  be the degree of a vertex  $v \in V'$  within this forest.
- ▶ Color a vertex  $v \in V'$  red if it corresponds to a component from  $\mathbb{C}$  (an active component). Otw. color it blue. (Let  $B$  the set of blue vertices (with non-zero degree) and  $R$  the set of red vertices)
- ▶ We have

$$\sum_{v \in R} \deg(v) \geq \sum_{C \in \mathbb{C}} |\delta(C) \cap F'| \stackrel{?}{\leq} 2|\mathbb{C}| = 2|R|$$

- ▶ Contract all edges in  $F_i$  into single vertices  $V'$ .
- ▶ We can consider the forest  $H$  on the set of vertices  $V'$ .
- ▶ Let  $\deg(v)$  be the degree of a vertex  $v \in V'$  within this forest.
- ▶ Color a vertex  $v \in V'$  red if it corresponds to a component from  $\mathbb{C}$  (an active component). Otw. color it blue. (Let  $B$  the set of blue vertices (with non-zero degree) and  $R$  the set of red vertices)
- ▶ We have

$$\sum_{v \in R} \deg(v) \geq \sum_{C \in \mathbb{C}} |\delta(C) \cap V'| \stackrel{?}{\leq} 2|\mathbb{C}| = 2|R|$$

- ▶ Contract all edges in  $F_i$  into single vertices  $V'$ .
- ▶ We can consider the forest  $H$  on the set of vertices  $V'$ .
- ▶ Let  $\deg(v)$  be the degree of a vertex  $v \in V'$  within this forest.
- ▶ Color a vertex  $v \in V'$  **red** if it corresponds to a component from  $\mathcal{C}$  (an active component). Otw. color it blue. (Let  $B$  the set of blue vertices (with non-zero degree) and  $R$  the set of red vertices)
- ▶ We have

$$\sum_{v \in R} \deg(v) \geq \sum_{C \in \mathcal{C}} |\delta(C) \cap V'| \stackrel{?}{\leq} 2|\mathcal{C}| = 2|R|$$

- ▶ Suppose that no node in  $B$  has degree one.

- ▶ Suppose that no node in  $B$  has degree one.
- ▶ Then

- ▶ Suppose that no node in  $B$  has degree one.
- ▶ Then

$$\sum_{v \in R} \deg(v)$$

- ▶ Suppose that no node in  $B$  has degree one.
- ▶ Then

$$\sum_{v \in R} \deg(v) = \sum_{v \in R \cup B} \deg(v) - \sum_{v \in B} \deg(v)$$



- ▶ Suppose that no node in  $B$  has degree one.
- ▶ Then

$$\begin{aligned}\sum_{v \in R} \deg(v) &= \sum_{v \in R \cup B} \deg(v) - \sum_{v \in B} \deg(v) \\ &\leq 2(|R| + |B|) - 2|B|\end{aligned}$$

- ▶ Suppose that no node in  $B$  has degree one.
- ▶ Then

$$\begin{aligned}\sum_{v \in R} \deg(v) &= \sum_{v \in R \cup B} \deg(v) - \sum_{v \in B} \deg(v) \\ &\leq 2(|R| + |B|) - 2|B| = 2|R|\end{aligned}$$

- ▶ Suppose that no node in  $B$  has degree one.
- ▶ Then

$$\begin{aligned}\sum_{v \in R} \deg(v) &= \sum_{v \in R \cup B} \deg(v) - \sum_{v \in B} \deg(v) \\ &\leq 2(|R| + |B|) - 2|B| = 2|R|\end{aligned}$$

- ▶ Every blue vertex with non-zero degree must have degree at least two.

- ▶ Suppose that no node in  $B$  has degree one.
- ▶ Then

$$\begin{aligned}\sum_{v \in R} \deg(v) &= \sum_{v \in R \cup B} \deg(v) - \sum_{v \in B} \deg(v) \\ &\leq 2(|R| + |B|) - 2|B| = 2|R|\end{aligned}$$

- ▶ Every blue vertex with non-zero degree must have degree at least two.
  - ▶ Suppose not. The single edge connecting  $b \in B$  comes from  $H$ , and, hence, is necessary.

- ▶ Suppose that no node in  $B$  has degree one.
- ▶ Then

$$\begin{aligned}\sum_{v \in R} \deg(v) &= \sum_{v \in R \cup B} \deg(v) - \sum_{v \in B} \deg(v) \\ &\leq 2(|R| + |B|) - 2|B| = 2|R|\end{aligned}$$

- ▶ Every blue vertex with non-zero degree must have degree at least two.
  - ▶ Suppose not. The single edge connecting  $b \in B$  comes from  $H$ , and, hence, is necessary.
  - ▶ But this means that the cluster corresponding to  $b$  must separate a source-target pair.

- ▶ Suppose that no node in  $B$  has degree one.
- ▶ Then

$$\begin{aligned}\sum_{v \in R} \deg(v) &= \sum_{v \in R \cup B} \deg(v) - \sum_{v \in B} \deg(v) \\ &\leq 2(|R| + |B|) - 2|B| = 2|R|\end{aligned}$$

- ▶ Every blue vertex with non-zero degree must have degree at least two.
  - ▶ Suppose not. The single edge connecting  $b \in B$  comes from  $H$ , and, hence, is necessary.
  - ▶ But this means that the cluster corresponding to  $b$  must separate a source-target pair.
  - ▶ But then it must be a red node.