CMPSC 465
Data Structures and Algorithms
Spring 2022

Instructor: Chunhao Wang

Greedy algorithms

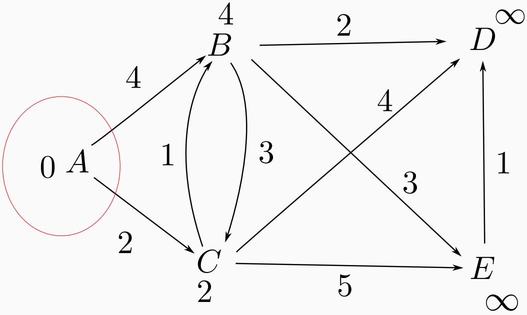
Greedy algorithms

Warm-up

Greedy algorithms

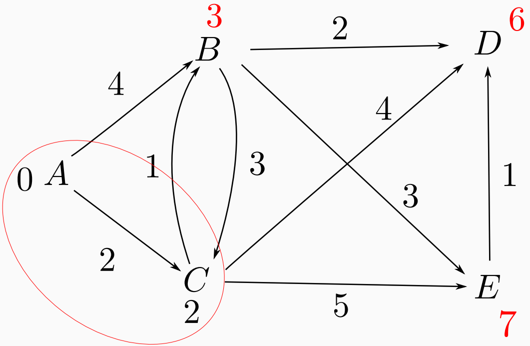
> "Greedy ... is good. Greedy is right. Greedy works."
>
> — Wall Street

Warm-up example: Dijkstra's algorithm (I)



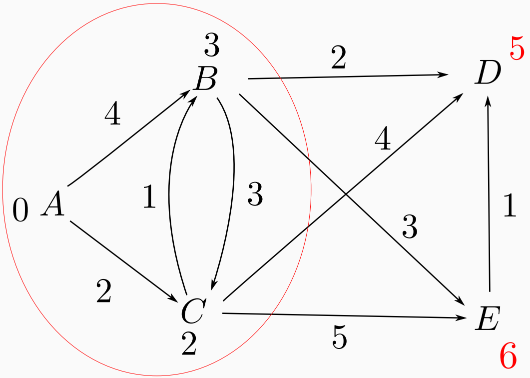
A	
B	4
C	2
D	
E	

Warm-up example: Dijkstra's algorithm (II)



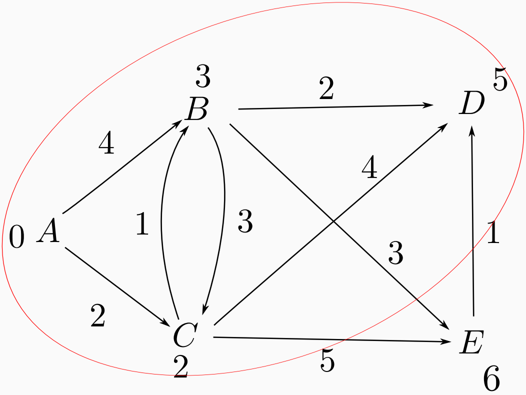
A C	
B	3
D	6
E	7

Warm-up example: Dijkstra's algorithm (III)



A C B	
D	5
E	6

Warm-up example: Dijkstra's algorithm (IV)



A C B D	
E	6

How to design a greedy algorithm?

Key step (the *greedy heuristic*)

View the problem as one where a sequence of choices are made,

How to design a greedy algorithm?

Key step (the *greedy heuristic*)

View the problem as one where a sequence of choices are made, and each choice leaves a single subproblem to solve

How to design a greedy algorithm?

Key step (the *greedy heuristic*)

View the problem as one where a sequence of choices are made, and each choice leaves a single subproblem to solve

Question: do greedy algorithms always work?

i.e. do they always produce the **optimal solution**?

A failure example (I)

0-1 Knapsack Problem

A Thief has a backpack with certain capacity. There is a set of items with certain weight and value.

A failure example (I)

0-1 Knapsack Problem

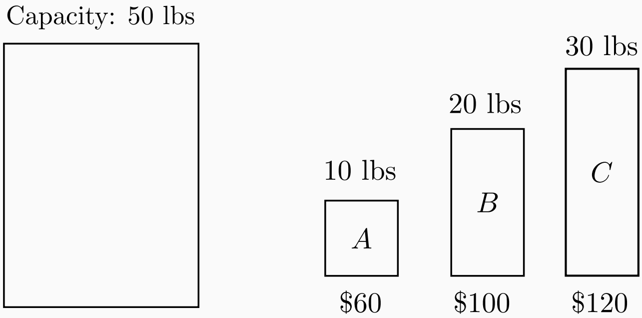
A Thief has a backpack with certain capacity. There is a set of items with certain weight and value. **Goal:** pack the backpack with the largest value

A failure example (I)

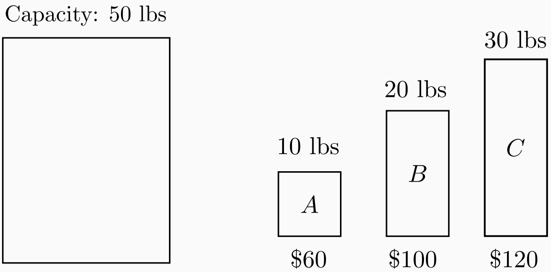
0-1 Knapsack Problem

A Thief has a backpack with certain capacity. There is a set of items with certain weight and value. **Goal:** pack the backpack with the largest value

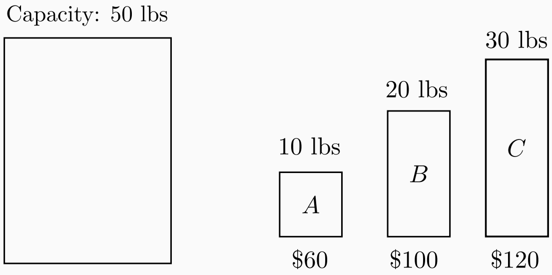
Example:



A failure example (II)

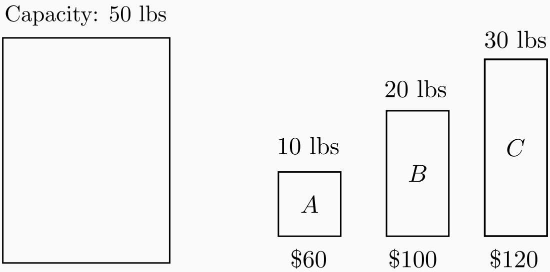


A failure example (II)



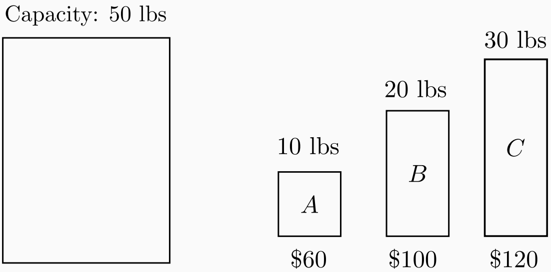
Greedy heuristic:

A failure example (II)



Greedy heuristic: always pick the item with the highest value/weight

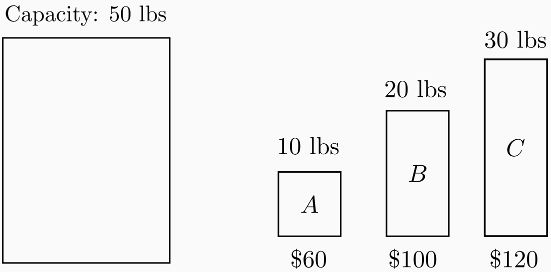
A failure example (II)



Greedy heuristic: always pick the item with the highest value/weight

Greedy solution:

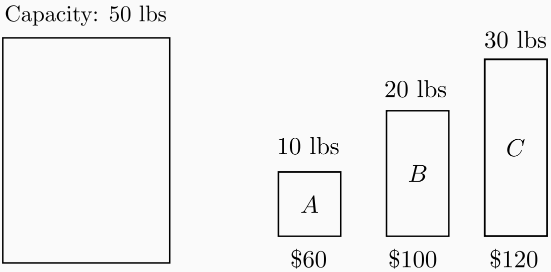
A failure example (II)



Greedy heuristic: always pick the item with the highest value/weight

Greedy solution: A, B total value: \$160

A failure example (II)

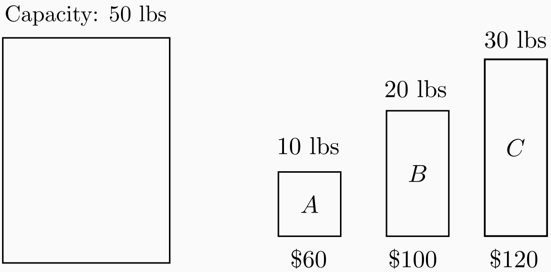


Greedy heuristic: always pick the item with the highest value/weight

Greedy solution: A, B total value: \$160

Better solutions?

A failure example (II)



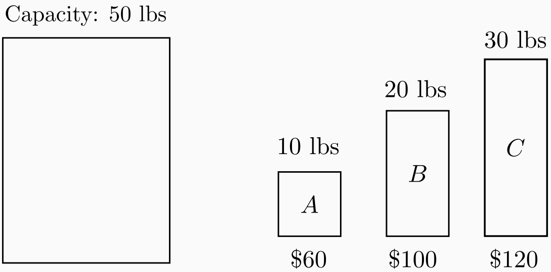
Greedy heuristic: always pick the item with the highest value/weight

Greedy solution: A, B total value: \$160

Better solutions?

- B, C total value: \$220

A failure example (II)



Greedy heuristic: always pick the item with the highest value/weight

Greedy solution: A, B total value: \$160

Better solutions?

- B, C total value: \$220
- A, C total value: \$180

Greedy choice

In order for the greedy heuristic to work, the problem should have the following property

Greedy choice property

There exists an optimal solution that makes the greedy choice

Greedy choice

In order for the greedy heuristic to work, the problem should have the following property

Greedy choice property

There exists an optimal solution that makes the greedy choice

Or: the current greedy choice is contained in **some** optimal solution

Greedy choice

In order for the greedy heuristic to work, the problem should have the following property

Greedy choice property

There exists an optimal solution that makes the greedy choice

Or: the current greedy choice is contained in **some** optimal solution

The greedy choice property shows that "the greedy choice is safe"

Greedy choice

In order for the greedy heuristic to work, the problem should have the following property

Greedy choice property

There exists an optimal solution that makes the greedy choice

Or: the current greedy choice is contained in **some** optimal solution

The greedy choice property shows that "the greedy choice is safe"

However, it won't guarantee that make a sequence of greedy choices leads to an optimal solution

Greedy choice

In order for the greedy heuristic to work, the problem should have the following property

Greedy choice property

There exists an optimal solution that makes the greedy choice

Or: the current greedy choice is contained in **some** optimal solution

The greedy choice property shows that "the greedy choice is safe"

However, it won't guarantee that make a sequence of greedy choices leads to an optimal solution

The 0-1 Knapsack problem fails in this property

Another failure example

Maximum weighted path

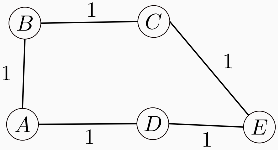
Given a graph with edge weights, and given two vertices. Find a path between them with the maximum total weight

Another failure example

Maximum weighted path

Given a graph with edge weights, and given two vertices. Find a path between them with the maximum total weight

Example:

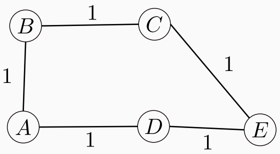


Another failure example

Maximum weighted path

Given a graph with edge weights, and given two vertices. Find a path between them with the maximum total weight

Example:



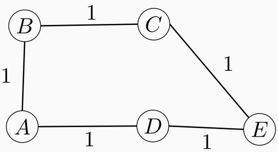
$\textsc{MaxWeightedPath}(A, C) = 3 \quad (A - D - E - C)$

Another failure example

Maximum weighted path

Given a graph with edge weights, and given two vertices. Find a path between them with the maximum total weight

Example:



$\textsc{MaxWeightedPath}(A, C) = 3 \quad (A - D - E - C)$

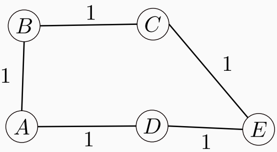
However, $\textsc{MaxWeightedPath}(A, C) \neq \textsc{MaxWeightedPath}(A, D) + \textsc{MaxWeightedPath}(D, C)$

Another failure example

Maximum weighted path

Given a graph with edge weights, and given two vertices. Find a path between them with the maximum total weight

Example:



$\textsc{MaxWeightedPath}(A, C) = 3 \quad (A - D - E - C)$

However, $\textsc{MaxWeightedPath}(A, C) \neq \textsc{MaxWeightedPath}(A, D) + \textsc{MaxWeightedPath}(D, C)$

Global optimal cannot be obtained by combining local optima

Optimal substructure

In order for the greedy heuristic to work, the problem should **also** have the following property

Optimal substructure property

An optimal solution to the problem contains within it the optimal solutions to the subproblems

Optimal substructure

In order for the greedy heuristic to work, the problem should **also** have the following property

Optimal substructure property

An optimal solution to the problem contains within it the optimal solutions to the subproblems

Or: solutions to subproblems can be combined to the global optimal solution

Optimal substructure

In order for the greedy heuristic to work, the problem should **also** have the following property

Optimal substructure property

An optimal solution to the problem contains within it the optimal solutions to the subproblems

Or: solutions to subproblems can be combined to the global optimal solution

The maximum weighted path problem fails in this property

Summary of greedy algorithms

- Design a greedy algorithm: find a greedy heuristic

Summary of greedy algorithms

- Design a greedy algorithm: find a greedy heuristic
  - View the problem as one where a sequence of choices are made, and each choice leaves a single subproblem to solve

Summary of greedy algorithms

- Design a greedy algorithm: find a greedy heuristic
  - View the problem as one where a sequence of choices are made, and each choice leaves a single subproblem to solve
- To prove this greedy algorithm yields an optimal solution:

Summary of greedy algorithms

- Design a greedy algorithm: find a greedy heuristic
  - View the problem as one where a sequence of choices are made, and each choice leaves a single subproblem to solve
- To prove this greedy algorithm yields an optimal solution:
  - Prove this problem has the greedy choice property w.r.t. the greedy heuristic

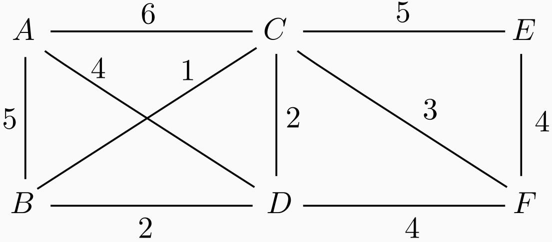
Summary of greedy algorithms

- Design a greedy algorithm: find a greedy heuristic
  - View the problem as one where a sequence of choices are made, and each choice leaves a single subproblem to solve
- To prove this greedy algorithm yields an optimal solution:
  - Prove this problem has the greedy choice property w.r.t. the greedy heuristic
  - Prove this problem has the optimal substructure property

Greedy algorithms

Minimum Spanning Tree

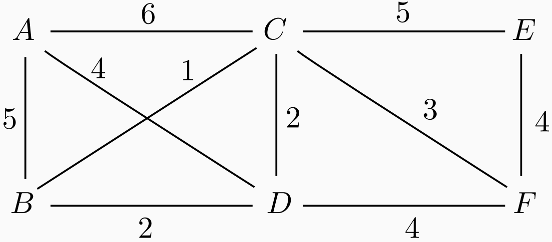
Minimum Spanning Tree (MST)



The Minimum Spanning Tree Problem (MST)

Input: undirected graph $G = (V, E)$ with edge weights $w_e \in \mathbb{R}, \forall e \in E$.

Minimum Spanning Tree (MST)

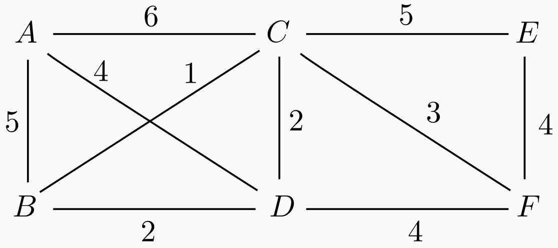


The Minimum Spanning Tree Problem (MST)

Input: undirected graph $G = (V, E)$ with edge weights $w_e \in \mathbb{R}, \forall e \in E$.

Output: A **tree** $T = (V, E')$ s.t.

Minimum Spanning Tree (MST)



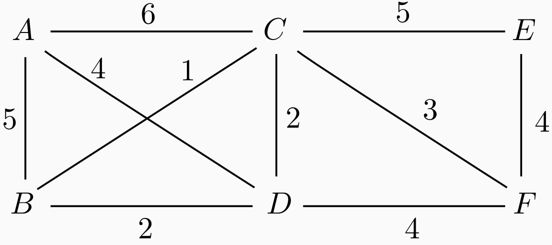
The Minimum Spanning Tree Problem (MST)

Input: undirected graph $G = (V, E)$ with edge weights $w_e \in \mathbb{R}, \forall e \in E$.

Output: A **tree** $T = (V, E')$ s.t.

1. $E' \subseteq E$

Minimum Spanning Tree (MST)



The Minimum Spanning Tree Problem (MST)

Input: undirected graph $G = (V, E)$ with edge weights $w_e \in \mathbb{R}, \forall e \in E$.
Output: A **tree** $T = (V, E')$ s.t.

1. $E' \subseteq E$
2. E' minimizes $\mathrm{weight}(T) = \sum_{e \in E'} w_e$

Greedy approach for MST

View the problem as one where a sequence of choices are made, and each choice leaves a single subproblem to solve

Greedy approach for MST

View the problem as one where a sequence of choices are made, and each choice leaves a single subproblem to solve

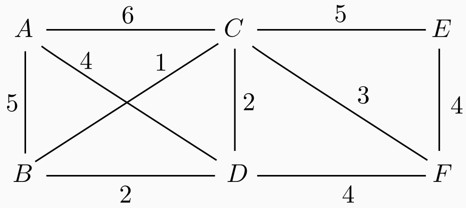
Greedy heuristic: add the lightest edge that doesn't induce a cycle

Greedy approach for MST

View the problem as one where a sequence of choices are made, and each choice leaves a single subproblem to solve

Greedy heuristic: add the lightest edge that doesn't induce a cycle

Example:

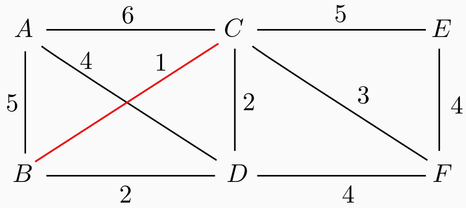


Greedy approach for MST

View the problem as one where a sequence of choices are made, and each choice leaves a single subproblem to solve

Greedy heuristic: add the lightest edge that doesn't induce a cycle

Example:

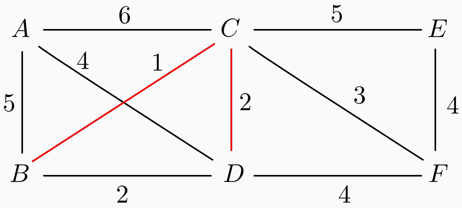


Greedy approach for MST

View the problem as one where a sequence of choices are made, and each choice leaves a single subproblem to solve

Greedy heuristic: add the lightest edge that doesn't induce a cycle

Example:

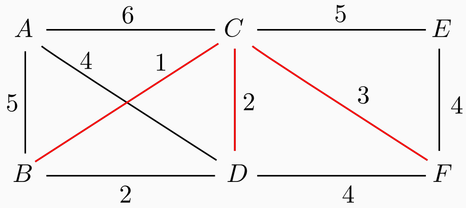


Greedy approach for MST

View the problem as one where a sequence of choices are made, and each choice leaves a single subproblem to solve

Greedy heuristic: add the lightest edge that doesn't induce a cycle

Example:

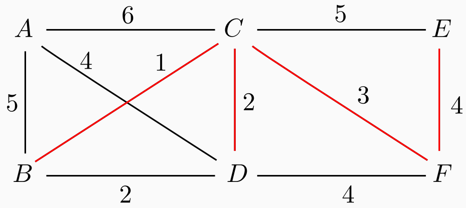


Greedy approach for MST

View the problem as one where a sequence of choices are made, and each choice leaves a single subproblem to solve

Greedy heuristic: add the lightest edge that doesn't induce a cycle

Example:

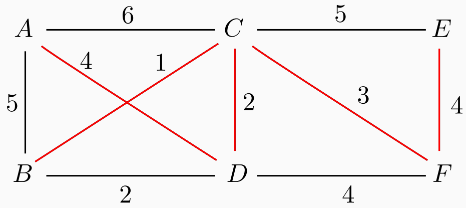


Greedy approach for MST

View the problem as one where a sequence of choices are made, and each choice leaves a single subproblem to solve

Greedy heuristic: add the lightest edge that doesn't induce a cycle

Example:

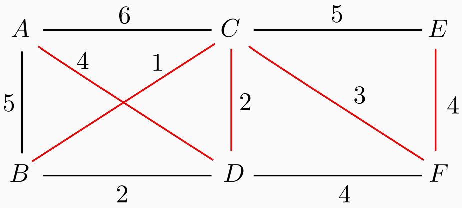


Greedy approach for MST

View the problem as one where a sequence of choices are made, and each choice leaves a single subproblem to solve

Greedy heuristic: add the lightest edge that doesn't induce a cycle

Example:



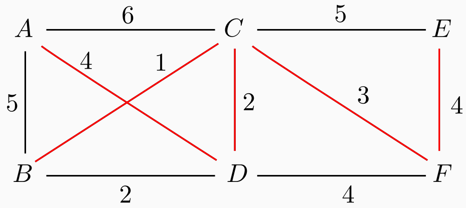
$\text{weight}(T) = 14$

Greedy approach for MST

View the problem as one where a sequence of choices are made, and each choice leaves a single subproblem to solve

Greedy heuristic: add the lightest edge that doesn't induce a cycle

Example:



$\text{weight}(T) = 14$

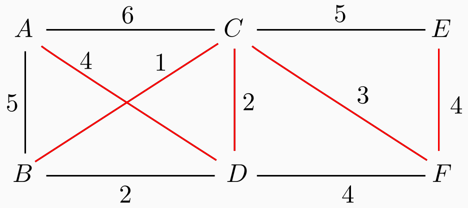
Optimality?

Greedy approach for MST

View the problem as one where a sequence of choices are made, and each choice leaves a single subproblem to solve

Greedy heuristic: add the lightest edge that doesn't induce a cycle

Example:



$\text{weight}(T) = 14$

Optimality?

We need to show greedy choice and optimal substructure

Optimality of greedy (I)

Some technical preliminaries

Optimality of greedy (I)

Some technical preliminaries

Definition

A **partition** of a set V is in the form (A, B) where $A \cup B = V$ and $A \cap B = \emptyset$

Optimality of greedy (I)

Some technical preliminaries

Definition

A **partition** of a set V is in the form (A, B) where $A \cup B = V$ and $A \cap B = \emptyset$

Definition

For an undirected graph $G = (V, E)$, a **cut** is a partition of V

Optimality of greedy (I)

Some technical preliminaries

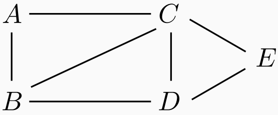
Definition

A **partition** of a set V is in the form (A, B) where $A \cup B = V$ and $A \cap B = \emptyset$

Definition

For an undirected graph $G = (V, E)$, a **cut** is a partition of V

Example:



$(\{A, B, C\}, \{D, E\})$
$(\{A, B\}, \{D, E\})$

Optimality of greedy (I)

Some technical preliminaries

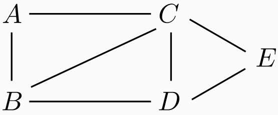
Definition

A **partition** of a set V is in the form (A, B) where $A \cup B = V$ and $A \cap B = \emptyset$

Definition

For an undirected graph $G = (V, E)$, a **cut** is a partition of V

Example:



$(\{A, B, C\}, \{D, E\})$ is a cut

$(\{A, B\}, \{D, E\})$

Optimality of greedy (I)

Some technical preliminaries

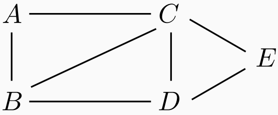
Definition

A **partition** of a set V is in the form (A, B) where $A \cup B = V$ and $A \cap B = \emptyset$

Definition

For an undirected graph $G = (V, E)$, a **cut** is a partition of V

Example:



$(\{A, B, C\}, \{D, E\})$ is a cut

$(\{A, B\}, \{D, E\})$ is a not a cut

Optimality of greedy (II)

Definition

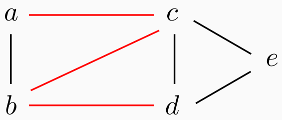
A cut $(S, V - S)$ **respects** $A \subseteq E$ if $\forall (u, v) \in A$, either $u, v \in S$ or $u, v \in V - S$

Optimality of greedy (II)

Definition

A cut $(S, V - S)$ **respects** $A \subseteq E$ if $\forall (u, v) \in A$, either $u, v \in S$ or $u, v \in V - S$

Example:



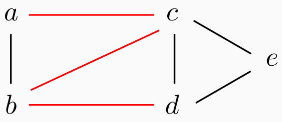
A: red edges

Optimality of greedy (II)

Definition

A cut $(S, V - S)$ **respects** $A \subseteq E$ if $\forall (u, v) \in A$, either $u, v \in S$ or $u, v \in V - S$

Example:



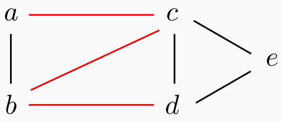
A: red edges
cut $(\{b, d\}, \{a, c, e\})$
cut $(\{e\}, \{a, b, c, d\})$

Optimality of greedy (II)

Definition

A cut $(S, V - S)$ **respects** $A \subseteq E$ if $\forall (u, v) \in A$, either $u, v \in S$ or $u, v \in V - S$

Example:



A: red edges

cut $(\{b, d\}, \{a, c, e\})$ doesn't respect A

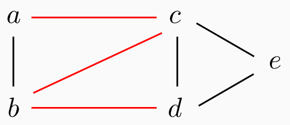
cut $(\{e\}, \{a, b, c, d\})$

Optimality of greedy (II)

Definition

A cut $(S, V - S)$ **respects** $A \subseteq E$ if $\forall (u, v) \in A$, either $u, v \in S$ or $u, v \in V - S$

Example:



A: red edges

cut $(\{b, d\}, \{a, c, e\})$ doesn't respect A

cut $(\{e\}, \{a, b, c, d\})$ respects A

Optimality of greedy (III)

Theorem (The cut property)

Let A be a subset of edges of some MST of $G = (V, E)$.

Optimality of greedy (III)

Theorem (The cut property)

Let A be a subset of edges of some MST of $G = (V, E)$. Let $(S, V - S)$ be a cut that respects A.

Optimality of greedy (III)

Theorem (The cut property)

Let A be a subset of edges of some MST of $G = (V, E)$. Let $(S, V - S)$ be a cut that respects A. Let e be the lightest edge across the cut.

Optimality of greedy (III)

Theorem (The cut property)

Let A be a subset of edges of some MST of $G = (V, E)$. Let $(S, V - S)$ be a cut that respects A. Let e be the lightest edge across the cut. Then $A \cup \{e\}$ is part of some MST

Optimality of greedy (III)

Theorem (The cut property)

Let A be a subset of edges of some MST of $G = (V, E)$. Let $(S, V - S)$ be a cut that respects A. Let e be the lightest edge across the cut. Then $A \cup \{e\}$ is part of some MST

How does this help?

Optimality of greedy (III)

Theorem (The cut property)

Let A be a subset of edges of some MST of $G = (V, E)$. Let $(S, V - S)$ be a cut that respects A. Let e be the lightest edge across the cut. Then $A \cup \{e\}$ is part of some MST

How does this help?

- It shows the greedy choice property:

Optimality of greedy (III)

Theorem (The cut property)

Let A be a subset of edges of some MST of $G = (V, E)$. Let $(S, V - S)$ be a cut that respects A. Let e be the lightest edge across the cut. Then $A \cup \{e\}$ is part of some MST

How does this help?

- It shows the greedy choice property:
  there exists an optimal solution that makes the greedy choice

Optimality of greedy (III)

Theorem (The cut property)

Let A be a subset of edges of some MST of $G = (V, E)$. Let $(S, V - S)$ be a cut that respects A. Let e be the lightest edge across the cut. Then $A \cup \{e\}$ is part of some MST

How does this help?

- It shows the greedy choice property:
  there exists an optimal solution that makes the greedy choice
- It also demonstrates the optimal substructure property

Optimality of greedy (III)

Theorem (The cut property)

Let A be a subset of edges of some MST of $G = (V, E)$. Let $(S, V - S)$ be a cut that respects A. Let e be the lightest edge across the cut. Then $A \cup \{e\}$ is part of some MST

How does this help?

- It shows the greedy choice property:
  there exists an optimal solution that makes the greedy choice
- It also demonstrates the optimal substructure property
  by applying this theorem multiple times until you can't