Advanced Algorithms (Fall 2025)

# Greedy Algorithms

Lecturers: 尹一通，刘景铖，栗师

Nanjing University

# Outline

1. Greedy Algorithms and Matroids
   - Recap: Maximum-Weight Spanning Tree Problem
   - Maximum-Weight Independent Set in Matroids
   - Examples of Matroids

2. Greedy Approximation Algorithms
   - $(\ln n + 1)$-Approximation for Set-Cover
   - $\left(1 - \frac{1}{e}\right)$-Approximation for Maximum Coverage
   - Submodular Functions
   - $\left(1 - \frac{1}{e}\right)$-Approximation for Cardinality-Constrained Submodular Maximization

# Outline

1. Greedy Algorithms and Matroids
   - Recap: Maximum-Weight Spanning Tree Problem
   - Maximum-Weight Independent Set in Matroids
   - Examples of Matroids

2. Greedy Approximation Algorithms
   - $(\ln n + 1)$-Approximation for Set-Cover
   - $(1 - \frac{1}{e})$-Approximation for Maximum Coverage
   - Submodular Functions
   - $(1 - \frac{1}{e})$-Approximation for Cardinality-Constrained Submodular Maximization

**Maximum-Weight Spanning Tree Problem**

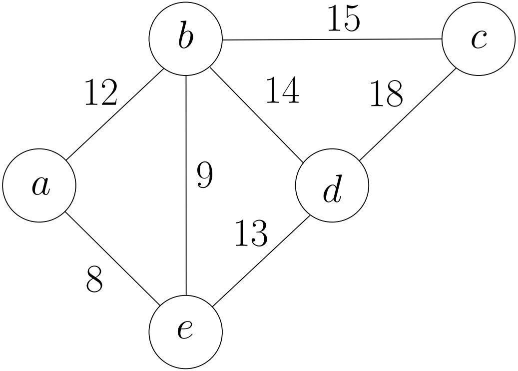
**Input:** Graph $G = (V, E)$ and edge weights $w \in \mathbb{Z}_{>0}^{E}$

**Output:** the spanning tree $T$ of $G$ with the maximum total weight

## Maximum-Weight Spanning Tree Problem

**Input:** Graph $G = (V, E)$ and edge weights $w \in \mathbb{Z}_{>0}^{E}$

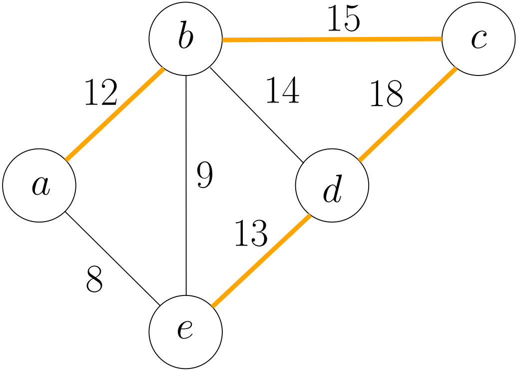
**Output:** the spanning tree $T$ of $G$ with the maximum total weight



## Maximum-Weight Spanning Tree Problem

**Input:** Graph $G = (V, E)$ and edge weights $w \in \mathbb{Z}^E_{>0}$

**Output:** the spanning tree $T$ of $G$ with the maximum total weight

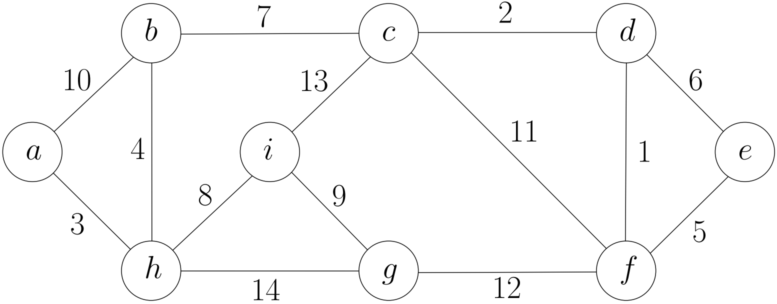


**Kruskal's Algorithm for Maximum-Weight Spanning Tree**

$F \leftarrow \emptyset$
sort edges in $E$ in non-increasing order of weights $w$
**for** each edge $(u, v)$ in the order **do**
  **if** $u$ and $v$ are not connected by a path of edges in $F$ **then**
    $F \leftarrow F \cup \{(u, v)\}$
**return** $(V, F)$

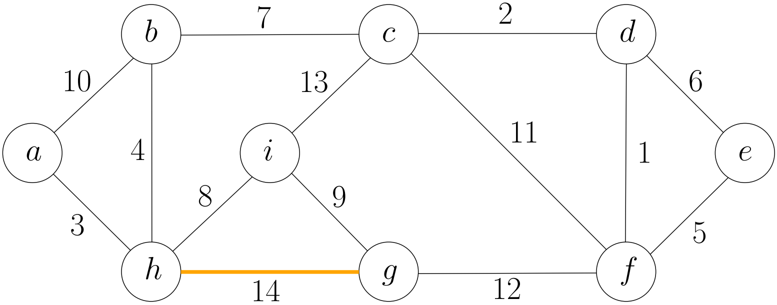
## Kruskal's Algorithm for Maximum-Weight Spanning Tree

$F \leftarrow \emptyset$
sort edges in $E$ in non-increasing order of weights $w$
**for** each edge $(u, v)$ in the order **do**
    **if** $u$ and $v$ are not connected by a path of edges in $F$ **then**
        $F \leftarrow F \cup \{(u, v)\}$
**return** $(V, F)$



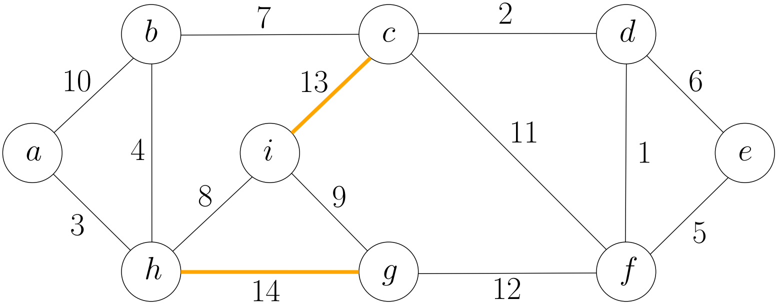
## Kruskal's Algorithm for Maximum-Weight Spanning Tree

$F \leftarrow \emptyset$
sort edges in $E$ in non-increasing order of weights $w$
**for** each edge $(u, v)$ in the order **do**
    **if** $u$ and $v$ are not connected by a path of edges in $F$ **then**
        $F \leftarrow F \cup \{(u, v)\}$
**return** $(V, F)$



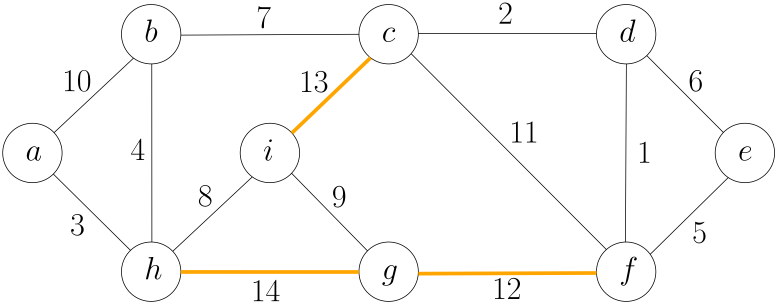
## Kruskal's Algorithm for Maximum-Weight Spanning Tree

$F \leftarrow \emptyset$
sort edges in $E$ in non-increasing order of weights $w$
**for** each edge $(u, v)$ in the order **do**
    **if** $u$ and $v$ are not connected by a path of edges in $F$ **then**
        $F \leftarrow F \cup \{(u, v)\}$
**return** $(V, F)$



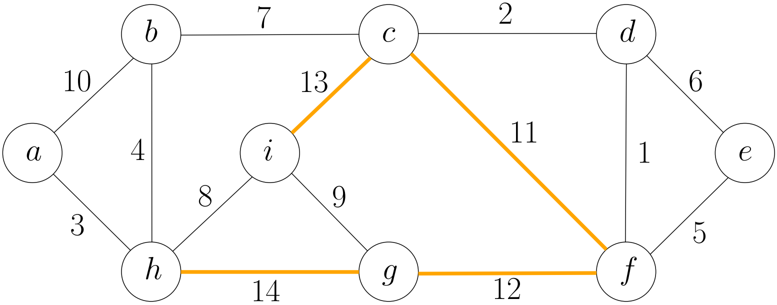
## Kruskal's Algorithm for Maximum-Weight Spanning Tree

$F \leftarrow \emptyset$
sort edges in $E$ in non-increasing order of weights $w$
**for** each edge $(u, v)$ in the order **do**
    **if** $u$ and $v$ are not connected by a path of edges in $F$ **then**
        $F \leftarrow F \cup \{(u, v)\}$
**return** $(V, F)$



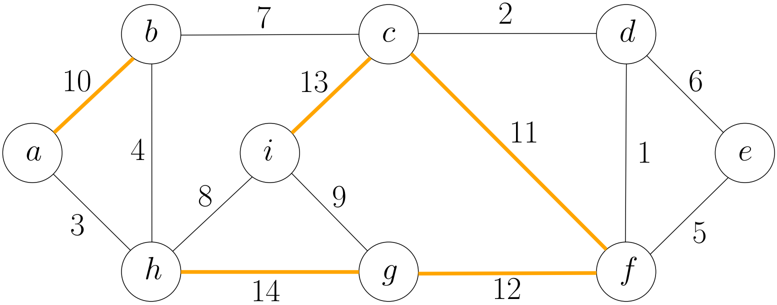
## Kruskal's Algorithm for Maximum-Weight Spanning Tree

1: $F \leftarrow \emptyset$
2: sort edges in $E$ in non-increasing order of weights $w$
3: **for** each edge $(u, v)$ in the order **do**
4:     **if** $u$ and $v$ are not connected by a path of edges in $F$ **then**
5:         $F \leftarrow F \cup \{(u, v)\}$
6: **return** $(V, F)$



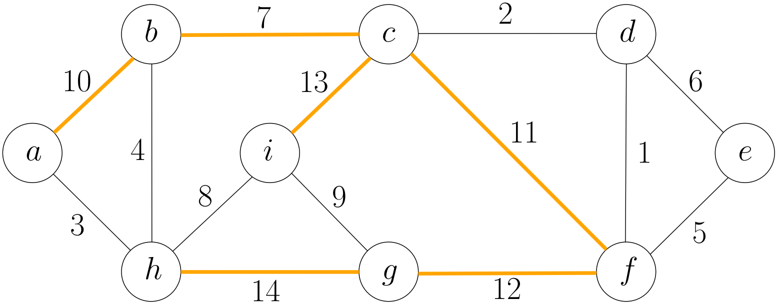
## Kruskal's Algorithm for Maximum-Weight Spanning Tree

$F \leftarrow \emptyset$
sort edges in $E$ in non-increasing order of weights $w$
**for** each edge $(u, v)$ in the order **do**
    **if** $u$ and $v$ are not connected by a path of edges in $F$ **then**
        $F \leftarrow F \cup \{(u, v)\}$
**return** $(V, F)$



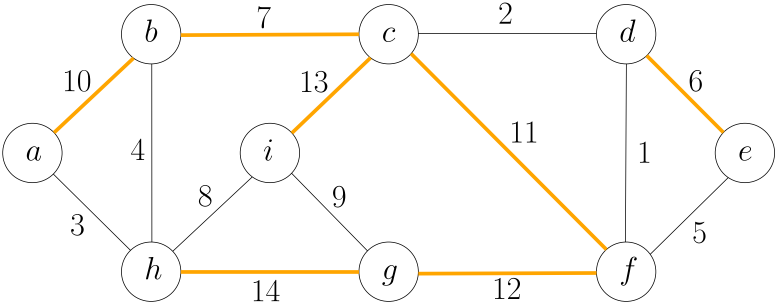
## Kruskal's Algorithm for Maximum-Weight Spanning Tree

$F \leftarrow \emptyset$
sort edges in $E$ in non-increasing order of weights $w$
**for** each edge $(u, v)$ in the order **do**
    **if** $u$ and $v$ are not connected by a path of edges in $F$ **then**
        $F \leftarrow F \cup \{(u, v)\}$
**return** $(V, F)$



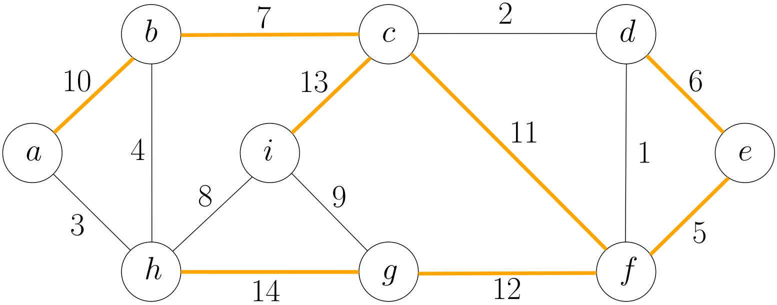
## Kruskal's Algorithm for Maximum-Weight Spanning Tree

$F \leftarrow \emptyset$
sort edges in $E$ in non-increasing order of weights $w$
**for** each edge $(u, v)$ in the order **do**
    **if** $u$ and $v$ are not connected by a path of edges in $F$ **then**
        $F \leftarrow F \cup \{(u, v)\}$
**return** $(V, F)$



## Kruskal's Algorithm for Maximum-Weight Spanning Tree

$F \leftarrow \emptyset$
sort edges in $E$ in non-increasing order of weights $w$
**for** each edge $(u, v)$ in the order **do**
    **if** $u$ and $v$ are not connected by a path of edges in $F$ **then**
        $F \leftarrow F \cup \{(u, v)\}$
**return** $(V, F)$



# Proof of Correctness of Kruskal's Algorithm

**Maximum-Weight Spanning Tree (MST) with Pre-Selected Edges**

**Input:** Graph $G = (V, E)$ and edge weights $w \in \mathbb{Z}_{>0}^{E}$
a set $F_0 \subseteq E$ of edges, that does not contain a cycle

**Output:** the maximum-weight spanning tree $T = (V, E_T)$ of $G$ satisfying $F_0 \subseteq E_T$

# Proof of Correctness of Kruskal's Algorithm

**Maximum-Weight Spanning Tree (MST) with Pre-Selected Edges**

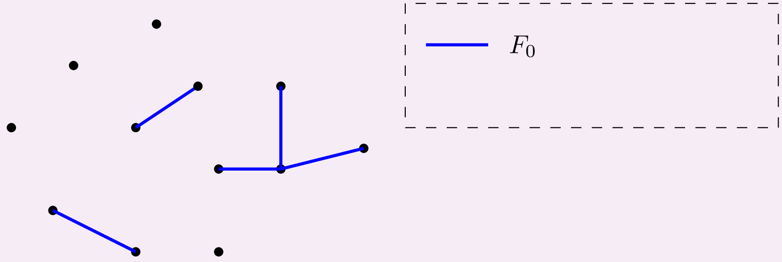
**Input:** Graph $G = (V, E)$ and edge weights $w \in \mathbb{Z}_{>0}^E$
a set $F_0 \subseteq E$ of edges, that does not contain a cycle

**Output:** the maximum-weight spanning tree $T = (V, E_T)$ of $G$ satisfying $F_0 \subseteq E_T$

**Lemma** (Key Lemma) Given an instance $(G = (V, E), w, F_0)$ of the MST with pre-selected edges problem, let $e^*$ be the maximum weight edge in $E \setminus F_0$ such that $F_0 \cup \{e^*\}$ does not contain a cycle. Then there is an optimum solution $T = (V, E_T)$ to the instance with $e^* \in E_T$.

# Proof of Correctness of Kruskal's Algorithm

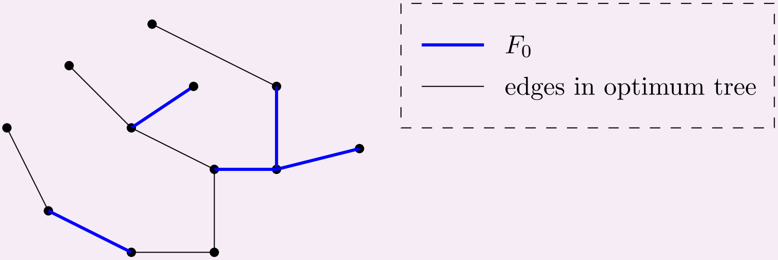
**Proof of Key Lemma.**




$\square$

# Proof of Correctness of Kruskal's Algorithm

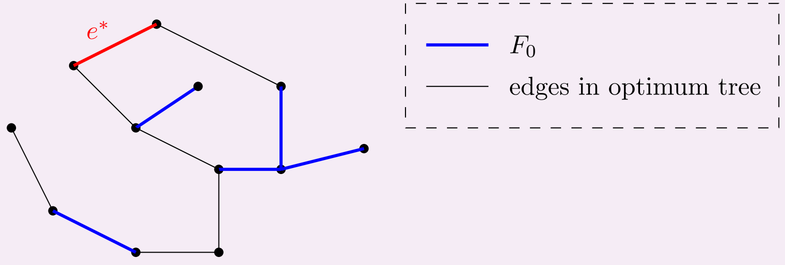
**Proof of Key Lemma.**




□

# Proof of Correctness of Kruskal's Algorithm

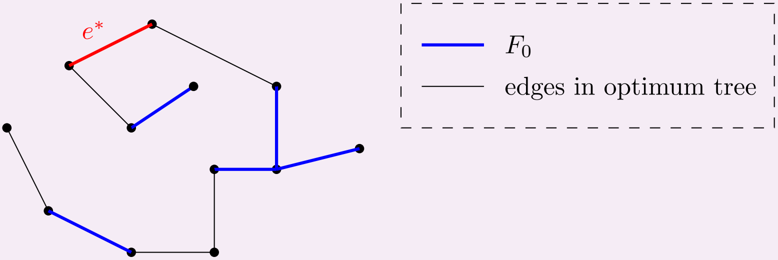
**Proof of Key Lemma.**




□

# Proof of Correctness of Kruskal's Algorithm

**Proof of Key Lemma.**




□

# Outline

1. Greedy Algorithms and Matroids
   - Recap: Maximum-Weight Spanning Tree Problem
   - Maximum-Weight Independent Set in Matroids
   - Examples of Matroids

2. Greedy Approximation Algorithms
   - $(\ln n + 1)$-Approximation for Set-Cover
   - $\left(1 - \frac{1}{e}\right)$-Approximation for Maximum Coverage
   - Submodular Functions
   - $\left(1 - \frac{1}{e}\right)$-Approximation for Cardinality-Constrained Submodular Maximization

**Q:** Does the greedy algorithm work for more general problems?

**Q:** Does the greedy algorithm work for more general problems?

### A General Maximization Problem

**Input:** $E$: the ground set of elements

$w \in \mathbb{Z}_{>0}^E$: weight vector on elements

$\mathcal{S}$: an (implicitly given) family of subsets of $E$

- $\emptyset \in \mathcal{S}$
- $\mathcal{S}$ is downward closed: if $A \in \mathcal{S}, B \subsetneq A$, then $B \in \mathcal{S}$.

**Output:** $A \in \mathcal{S}$ that maximizes $\sum_{e \in A} w_e$

**Q:** Does the greedy algorithm work for more general problems?

## A General Maximization Problem

**Input:** $E$: the ground set of elements

$w \in \mathbb{Z}_{>0}^E$: weight vector on elements

$\mathcal{S}$: an (implicitly given) family of subsets of $E$

- $\emptyset \in \mathcal{S}$
- $\mathcal{S}$ is downward closed: if $A \in \mathcal{S}, B \subsetneq A$, then $B \in \mathcal{S}$.

**Output:** $A \in \mathcal{S}$ that maximizes $\sum_{e \in A} w_e$

- maximum-weight spanning tree: $\mathcal{S} =$ family of forests

## Greedy Algorithm

$A \leftarrow \emptyset$

sort elements in $E$ in non-decreasing order of weights $w$

**for** each element $e$ in the order **do**

  **if** $A \cup \{e\} \in \mathcal{S}$ **then** $A \leftarrow A \cup \{e\}$

**return** $A$

### Greedy Algorithm

1: $A \leftarrow \emptyset$

2: sort elements in $E$ in non-decreasing order of weights $w$

3: **for** each element $e$ in the order **do**

4: **if** $A \cup \{e\} \in \mathcal{S}$ **then** $A \leftarrow A \cup \{e\}$

5: **return** $A$

### Examples where Greedy Algorithm is Not Optimum

- Knapsack Packing: given elements $E$, where every element has a value and a cost, and a cost budget $C$, the goal is to find a maximum value subset of items with cost at most $C$

## Greedy Algorithm

```
A ← ∅
sort elements in E in non-decreasing order of weights w
for each element e in the order do
    if A ∪ {e} ∈ S then A ← A ∪ {e}
return A
```

## Examples where Greedy Algorithm is Not Optimum

- Knapsack Packing: given elements $E$, where every element has a value and a cost, and a cost budget $C$, the goal is to find a maximum value subset of items with cost at most $C$
- Maximum Weight Bipartite Graph Matching

**Greedy Algorithm**

$A \leftarrow \emptyset$
sort elements in $E$ in non-decreasing order of weights $w$
**for** each element $e$ in the order **do**
    **if** $A \cup \{e\} \in \mathcal{S}$ **then** $A \leftarrow A \cup \{e\}$
**return** $A$

**Examples where Greedy Algorithm is Not Optimum**

- Knapsack Packing: given elements $E$, where every element has a value and a cost, and a cost budget $C$, the goal is to find a maximum value subset of items with cost at most $C$
- Maximum Weight Bipartite Graph Matching

- Matroids: cases where greedy algorithm is optimum

**Def.** A (finite) matroid $\mathcal{M}$ is a pair $(E, \mathcal{I})$, where $E$ is a finite set (called the ground set) and $\mathcal{I}$ is a family of subsets of $E$ (called independent sets) with the following properties:

1. $\emptyset \in \mathcal{I}$.
2. (downward-closed property) If $B \subsetneq A \in \mathcal{I}$, then $B \in \mathcal{I}$.
3. (augmentation/exchange property) If $A, B \in \mathcal{I}$ and $|B| < |A|$, then there exists $e \in A \setminus B$ such that $B \cup \{e\} \in \mathcal{I}$.

**Def.** A (finite) matroid $\mathcal{M}$ is a pair $(E, \mathcal{I})$, where $E$ is a finite set (called the ground set) and $\mathcal{I}$ is a family of subsets of $E$ (called independent sets) with the following properties:

1. $\emptyset \in \mathcal{I}$.
2. (downward-closed property) If $B \subsetneq A \in \mathcal{I}$, then $B \in \mathcal{I}$.
3. (augmentation/exchange property) If $A, B \in \mathcal{I}$ and $|B| < |A|$, then there exists $e \in A \setminus B$ such that $B \cup \{e\} \in \mathcal{I}$.

**Lemma** Let $G = (V, E)$. $F \subseteq E$ is in $\mathcal{I}$ iff $(V, F)$ is a forest. Then $(E, \mathcal{I})$ is a matroid, and it is called a graphic matroid.

**Def.** A (finite) matroid $\mathcal{M}$ is a pair $(E, \mathcal{I})$, where $E$ is a finite set (called the ground set) and $\mathcal{I}$ is a family of subsets of $E$ (called independent sets) with the following properties:

1. $\emptyset \in \mathcal{I}$.
2. (downward-closed property) If $B \subsetneq A \in \mathcal{I}$, then $B \in \mathcal{I}$.
3. (augmentation/exchange property) If $A, B \in \mathcal{I}$ and $|B| < |A|$, then there exists $e \in A \setminus B$ such that $B \cup \{e\} \in \mathcal{I}$.

**Lemma** Let $G = (V, E)$. $F \subseteq E$ is in $\mathcal{I}$ iff $(V, F)$ is a forest. Then $(E, \mathcal{I})$ is a matroid, and it is called a graphic matroid.

Proof of Exchange Property.

- $|B| < |A| \Rightarrow (V, B)$ has more CC than $(V, A)$.
- Some edge in $A$ connects two different CC of $(V, B)$. $\square$

## Feasible Family for Knapsack Packing Does Not Satisfy Augmentation Property

- $c_1 = c_2 = 10, c_3 = 20, C = 20$.
- $\{1, 2\}, \{3\} \in \mathcal{I}$, but $\{1, 3\}, \{2, 3\} \notin \mathcal{I}$.

### Feasible Family for Knapsack Packing Does Not Satisfy Augmentation Property

- $c_1 = c_2 = 10, c_3 = 20, C = 20.$
- $\{1, 2\}, \{3\} \in \mathcal{I}$, but $\{1, 3\}, \{2, 3\} \notin \mathcal{I}$.

### Feasible Family for Bipartite Matching Does Not Satisfy Augmentation Property

- Complete bipartite graph between $\{a_1, a_2\}$ and $\{b_1, b_2\}$.
- $\{(a_1, b_1), (a_2, b_2)\}, \{(a_1, b_2)\} \in \mathcal{I}$.

### Feasible Family for Knapsack Packing Does Not Satisfy Augmentation Property

- $c_1 = c_2 = 10, c_3 = 20, C = 20$.
- $\{1, 2\}, \{3\} \in \mathcal{I}$, but $\{1, 3\}, \{2, 3\} \notin \mathcal{I}$.

### Feasible Family for Bipartite Matching Does Not Satisfy Augmentation Property

- Complete bipartite graph between $\{a_1, a_2\}$ and $\{b_1, b_2\}$.
- $\{(a_1, b_1), (a_2, b_2)\}, \{(a_1, b_2)\} \in \mathcal{I}$.

**Theorem** The greedy algorithm gives optimum solution for the maximum-weight independent set problem in a matroid.

**Lemma** (Key Lemma)

- given: matroid $\mathcal{M} = (E, \mathcal{I})$, weights $w \in \mathbb{Z}_{>0}^{E}$, $A \in \mathcal{I}$,
- goal: find a maximum weight independent set containing $A$
- $e^* = \arg\max_{e \in E \setminus A : A \cup \{e\} \in \mathcal{I}} w_e$, assuming $e^*$ exists

**Lemma** (Key Lemma)

- given: matroid $\mathcal{M} = (E, \mathcal{I})$, weights $w \in \mathbb{Z}_{>0}^{E}$, $A \in \mathcal{I}$,
- goal: find a maximum weight independent set containing $A$
- $e^* = \arg\max_{e \in E \setminus A : A \cup \{e\} \in \mathcal{I}} w_e$, assuming $e^*$ exists
- Then, some optimum solution contains $e^*$

**Lemma** (Key Lemma)

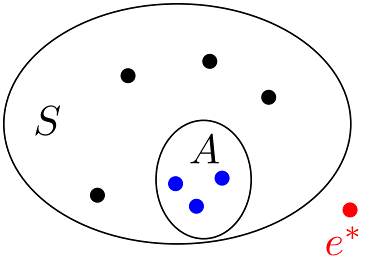
- given: matroid $\mathcal{M} = (E, \mathcal{I})$, weights $w \in \mathbb{Z}_{>0}^{E}$, $A \in \mathcal{I}$,
- goal: find a maximum weight independent set containing $A$
- $e^* = \arg\max_{e \in E \setminus A: A \cup \{e\} \in \mathcal{I}} w_e$, assuming $e^*$ exists
- Then, some optimum solution contains $e^*$

- let $S \supseteq A, S \in \mathcal{I}$ be an optimum solution, $e^* \notin S$

**Lemma** (Key Lemma)

- given: matroid $\mathcal{M} = (E, \mathcal{I})$, weights $w \in \mathbb{Z}_{>0}^E$, $A \in \mathcal{I}$,
- goal: find a maximum weight independent set containing $A$
- $e^* = \arg\max_{e \in E \setminus A: A \cup \{e\} \in \mathcal{I}} w_e$, assuming $e^*$ exists
- Then, some optimum solution contains $e^*$

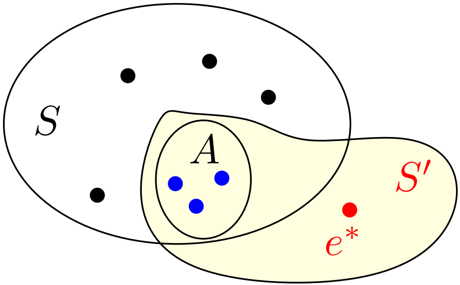
- let $S \supseteq A, S \in \mathcal{I}$ be an optimum solution, $e^* \notin S$



**Lemma** (Key Lemma)

- given: matroid $\mathcal{M} = (E, \mathcal{I})$, weights $w \in \mathbb{Z}_{>0}^E$, $A \in \mathcal{I}$,
- goal: find a maximum weight independent set containing $A$
- $e^* = \arg\max_{e \in E \setminus A: A \cup \{e\} \in \mathcal{I}} w_e$, assuming $e^*$ exists
- Then, some optimum solution contains $e^*$

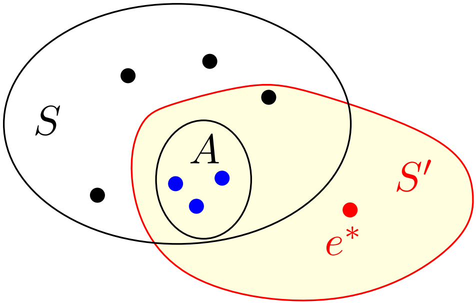
- let $S \supseteq A, S \in \mathcal{I}$ be an optimum solution, $e^* \notin S$



**Lemma** (Key Lemma)

- given: matroid $\mathcal{M} = (E, \mathcal{I})$, weights $w \in \mathbb{Z}_{>0}^E$, $A \in \mathcal{I}$,
- goal: find a maximum weight independent set containing $A$
- $e^* = \arg\max_{e \in E \setminus A: A \cup \{e\} \in \mathcal{I}} w_e$, assuming $e^*$ exists
- Then, some optimum solution contains $e^*$

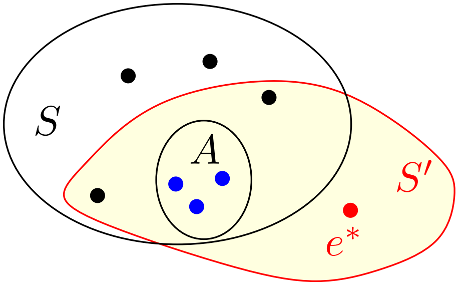
- let $S \supseteq A, S \in \mathcal{I}$ be an optimum solution, $e^* \notin S$



**Lemma** (Key Lemma)

- given: matroid $\mathcal{M} = (E, \mathcal{I})$, weights $w \in \mathbb{Z}_{>0}^E$, $A \in \mathcal{I}$,
- goal: find a maximum weight independent set containing $A$
- $e^* = \arg\max_{e \in E \setminus A: A \cup \{e\} \in \mathcal{I}} w_e$, assuming $e^*$ exists
- Then, some optimum solution contains $e^*$

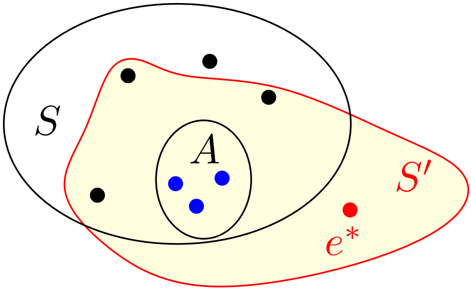
- let $S \supseteq A, S \in \mathcal{I}$ be an optimum solution, $e^* \notin S$



**Lemma** (Key Lemma)

- given: matroid $\mathcal{M} = (E, \mathcal{I})$, weights $w \in \mathbb{Z}_{>0}^E$, $A \in \mathcal{I}$,
- goal: find a maximum weight independent set containing $A$
- $e^* = \arg\max_{e \in E \setminus A : A \cup \{e\} \in \mathcal{I}} w_e$, assuming $e^*$ exists
- Then, some optimum solution contains $e^*$

- let $S \supseteq A, S \in \mathcal{I}$ be an optimum solution, $e^* \notin S$



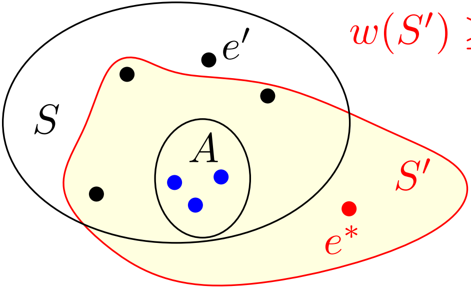
**Lemma** (Key Lemma)

- given: matroid $\mathcal{M} = (E, \mathcal{I})$, weights $w \in \mathbb{Z}_{>0}^E$, $A \in \mathcal{I}$,
- goal: find a maximum weight independent set containing $A$
- $e^* = \arg\max_{e \in E \setminus A: A \cup \{e\} \in \mathcal{I}} w_e$, assuming $e^*$ exists
- Then, some optimum solution contains $e^*$

- let $S \supseteq A, S \in \mathcal{I}$ be an optimum solution, $e^* \notin S$

$w_{e^*} \geq w_{e'}$

$w(S') \geq w(S)$



**Lemma** (Key Lemma)

- given: matroid $\mathcal{M} = (E, \mathcal{I})$, weights $w \in \mathbb{Z}_{>0}^E$, $A \in \mathcal{I}$,
- goal: find a maximum weight independent set containing $A$
- $e^* = \arg\max_{e \in E \setminus A : A \cup \{e\} \in \mathcal{I}} w_e$, assuming $e^*$ exists
- Then, some optimum solution contains $e^*$

Proof.

- let $S \supseteq A, S \in \mathcal{I}$ be an optimum solution, $e^* \notin S$

**Lemma** (Key Lemma)

- given: matroid $\mathcal{M} = (E, \mathcal{I})$, weights $w \in \mathbb{Z}^E_{>0}$, $A \in \mathcal{I}$,
- goal: find a maximum weight independent set containing $A$
- $e^* = \arg\max_{e \in E \setminus A: A \cup \{e\} \in \mathcal{I}} w_e$, assuming $e^*$ exists
- Then, some optimum solution contains $e^*$

Proof.

- let $S \supseteq A, S \in \mathcal{I}$ be an optimum solution, $e^* \notin S$

$S' \leftarrow A \cup \{e^*\}$
**while** $|S'| < |S|$ **do**
    let $e$ be any element in $S \setminus S'$ with $S' \cup \{e\} \in \mathcal{I}$ $\triangleright$ $e$ exists due to exchange property
    $S' \leftarrow S' \cup \{e\}$

**Lemma** (Key Lemma)

- given: matroid $\mathcal{M} = (E, \mathcal{I})$, weights $w \in \mathbb{Z}_{>0}^E$, $A \in \mathcal{I}$,
- goal: find a maximum weight independent set containing $A$
- $e^* = \arg\max_{e \in E \setminus A : A \cup \{e\} \in \mathcal{I}} w_e$, assuming $e^*$ exists
- Then, some optimum solution contains $e^*$

Proof.

- let $S \supseteq A, S \in \mathcal{I}$ be an optimum solution, $e^* \notin S$

```
S' ← A ∪ {e*}
while |S'| < |S| do
    let e be any element in S \ S' with S' ∪ {e} ∈ I
                                    ▷ e exists due to exchange property
    S' ← S' ∪ {e}
```

- $S'$ and $S$ differ by exactly one element

**Lemma** (Key Lemma)

- given: matroid $\mathcal{M} = (E, \mathcal{I})$, weights $w \in \mathbb{Z}_{>0}^E$, $A \in \mathcal{I}$,
- goal: find a maximum weight independent set containing $A$
- $e^* = \arg\max_{e \in E \setminus A: A \cup \{e\} \in \mathcal{I}} w_e$, assuming $e^*$ exists
- Then, some optimum solution contains $e^*$

Proof.

- let $S \supseteq A, S \in \mathcal{I}$ be an optimum solution, $e^* \notin S$

  $S' \leftarrow A \cup \{e^*\}$<br>
  **while** $|S'| < |S|$ **do**<br>
      let $e$ be any element in $S \setminus S'$ with $S' \cup \{e\} \in \mathcal{I}$<br>
      $\triangleright$ $e$ exists due to exchange property<br>
      $S' \leftarrow S' \cup \{e\}$

- $S'$ and $S$ differ by exactly one element
- $w(S') := \sum_{e \in S'} w_e \geq w(S) \implies S'$ is also optimum $\square$

# Outline

# Examples of Matroids

- $E$: the ground set $\qquad$ $\mathcal{I}$: the family of independent sets

# Examples of Matroids

- $E$: the ground set $\qquad$ $\mathcal{I}$: the family of independent sets
- Uniform Matroid: $k \in \mathbb{Z}_{>0}$.

$$\mathcal{I} = \{A \subseteq E : |A| \leq k\}.$$

# Examples of Matroids

- $E$: the ground set $\mathcal{I}$: the family of independent sets
- Uniform Matroid: $k \in \mathbb{Z}_{>0}$.

$$\mathcal{I} = \{A \subseteq E : |A| \leq k\}.$$

- Partition Matroid: partition $(E_1, E_2, \cdots, E_t)$ of $E$, positive integers $k_1, k_2, \cdots, k_t$

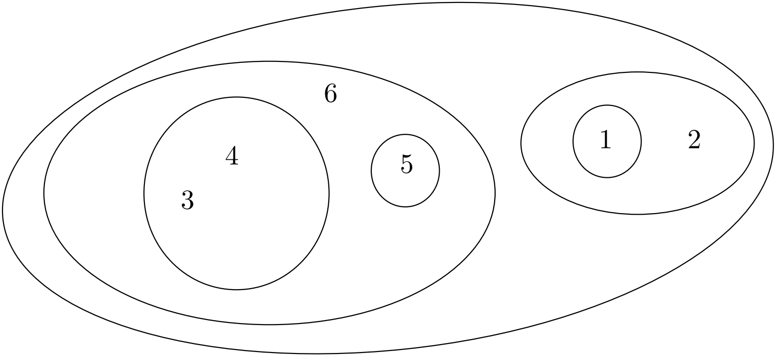
$$\mathcal{I} = \{A \subseteq E : |A \cap E_i| \leq k_i, \forall i \in [t]\}.$$

# Examples of Matroids

- $E$: the ground set $\qquad\qquad \mathcal{I}$: the family of independent sets
- Uniform Matroid: $k \in \mathbb{Z}_{>0}$.
  $$\mathcal{I} = \{A \subseteq E : |A| \leq k\}.$$
- Partition Matroid: partition $(E_1, E_2, \cdots, E_t)$ of $E$, positive integers $k_1, k_2, \cdots, k_t$
  $$\mathcal{I} = \{A \subseteq E : |A \cap E_i| \leq k_i, \forall i \in [t]\}.$$
- Laminar Matroid: laminar family of subsets of $E$ $\{E_1, E_2, \cdots, E_t\}$, positive integers $k_1, k_2, \cdots, k_t$
  $$\mathcal{I} = \{A \subseteq E : |A \cap E_i| \leq k_i, \forall i \in [t]\}.$$

**Def.** A family $\{E_1, E_2, \cdots, E_t\}$ of subsets of $E$ is said to be laminar if for every two distinct subsets $E_i, E_j$ in the family, we have $E_i \cap E_j = \emptyset$ or $E_i \subsetneq E_j$ or $E_j \subsetneq E_i$.

- $\Big\{\{1\}, \{1,2\}, \{3,4\}, \{5\}, \{3,4,5,6\}, \{1,2,3,4,5,6\}\Big\}$ is a laminar family.



# Examples of Matroids

- $E$: the ground set $\qquad$ $\mathcal{I}$: the family of independent sets
- Graphic Matroid: graph $G = (V, E)$

  $$\mathcal{I} = \{A \subseteq E : (V, A) \text{ is a forest}\}$$

# Examples of Matroids

- $E$: the ground set $\qquad$ $\mathcal{I}$: the family of independent sets
- Graphic Matroid: graph $G = (V, E)$

  $\mathcal{I} = \{A \subseteq E : (V, A) \text{ is a forest}\}$
- Transversal Matroid: a bipartite graph $G = (E \uplus B, \mathcal{E})$

  $\mathcal{I} = \{A \subseteq E : \text{there is a matching in } G \text{ covering } A\}$

# Examples of Matroids

- $E$: the ground set $\qquad\qquad$ $\mathcal{I}$: the family of independent sets
- Graphic Matroid: graph $G = (V, E)$

$$\mathcal{I} = \{A \subseteq E : (V, A) \text{ is a forest}\}$$

- Transversal Matroid: a bipartite graph $G = (E \uplus B, \mathcal{E})$

$$\mathcal{I} = \{A \subseteq E : \text{there is a matching in } G \text{ covering } A\}$$

- Linear Matroid: a vector $\vec{v}_e \in \mathbb{R}^d$ for every $e \in E$

$$\mathcal{I} = \{A \subseteq E : \text{vectors } \{\vec{v}_e\}_{e \in A} \text{ are linearly independent}\}$$

# Examples of Matroids

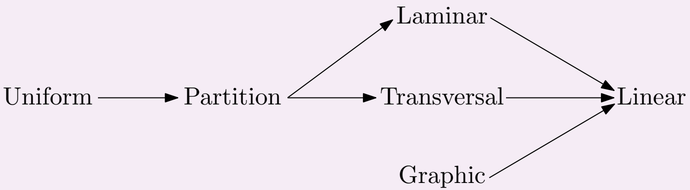
- $E$: the ground set $\qquad$ $\mathcal{I}$: the family of independent sets
- Graphic Matroid: graph $G = (V, E)$

  $\mathcal{I} = \{A \subseteq E : (V, A) \text{ is a forest}\}$
- Transversal Matroid: a bipartite graph $G = (E \uplus B, \mathcal{E})$

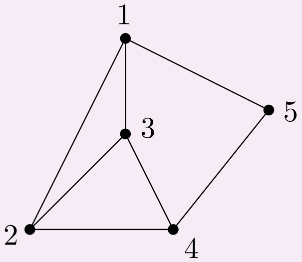
  $\mathcal{I} = \{A \subseteq E : \text{there is a matching in } G \text{ covering } A\}$
- Linear Matroid: a vector $\vec{v}_e \in \mathbb{R}^d$ for every $e \in E$

  $\mathcal{I} = \{A \subseteq E : \text{vectors } \{\vec{v}_e\}_{e \in A} \text{ are linearly independent}\}$

**Relationship between matroids**



## A Graphic Matroid is A Linear Matroid



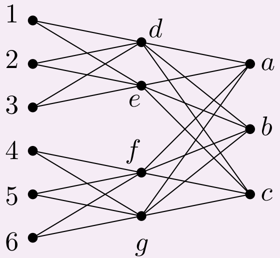

| edges | vectors |
|---|---|
| $(1,2)$ | $(1,-1,0,0,0)$ |
| $(1,3)$ | $(1,0,-1,0,0)$ |
| $(1,5)$ | $(1,0,0,0,-1)$ |
| $(2,3)$ | $(0,1,-1,0,0)$ |
| $(2,4)$ | $(0,1,0,-1,0)$ |
| $(3,4)$ | $(0,0,1,-1,0)$ |
| $(4,5)$ | $(0,0,0,1,-1)$ |

# A Laminar Matroid is A Linear Matroid

## Example

| sets | upper bounds |
|---:|:---:|
| $\{1, 2, 3\}$ | 2 |
| $\{4, 5, 6\}$ | 2 |
| $\{1, 2, 3, 4, 5, 6\}$ | 3 |

## A DAG (left to right)

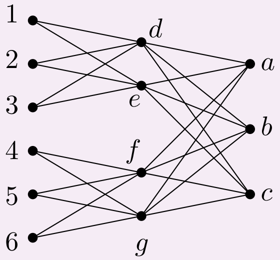


# A Laminar Matroid is A Linear Matroid

**Example**

| sets | upper bounds |
|---:|:---:|
| $\{1,2,3\}$ | 2 |
| $\{4,5,6\}$ | 2 |
| $\{1,2,3,4,5,6\}$ | 3 |

**A DAG (left to right)**




- $x^a, x^b, x^c \in \mathbb{R}^3$ are linearly independent rational vectors
- $x^d, x^e, x^f, x^g$: $\text{rand}(0,1) \cdot x^a + \text{rand}(0,1) \cdot x^b + \text{rand}(0,1) \cdot x^c$
- $x^1, x^2, x^3$: $\text{rand}(0,1) \cdot x^d + \text{rand}(0,1) \cdot x^e$
- $x^4, x^5, x^6$: $\text{rand}(0,1) \cdot x^f + \text{rand}(0,1) \cdot x^g$
- each $\text{rand}(0,1)$ gives an independent random real in $[0,1]$
- almost surely, all the random numbers are algebraically independent

# Outline

# Recap: Approximation Algorithms

- For minimization problems:

$$\text{approximation ratio} := \frac{\text{cost of our solution}}{\text{cost of optimum solution}} \geq 1$$

- For maximization problems:

$$\text{approximation ratio} := \frac{\text{value of our solution}}{\text{value of optimum solution}} \leq 1$$

or

$$\text{approximation ratio} := \frac{\text{value of optimum solution}}{\text{value of our solution}} \geq 1$$

# Outline

1. Greedy Algorithms and Matroids
   - Recap: Maximum-Weight Spanning Tree Problem
   - Maximum-Weight Independent Set in Matroids
   - Examples of Matroids
2. Greedy Approximation Algorithms
   - $(\ln n + 1)$-Approximation for Set-Cover
   - $(1 - \frac{1}{e})$-Approximation for Maximum Coverage
   - Submodular Functions
   - $(1 - \frac{1}{e})$-Approximation for Cardinality-Constrained Submodular Maximization

**Set Cover**

**Input:** $U, |U| = n$: ground set

$S_1, S_2, \cdots, S_m \subseteq U$

**Output:** minimum size set $C \subseteq [m]$ such that $\bigcup_{i \in C} S_i = U$

## Set Cover

**Input:** $U, |U| = n$: ground set

$S_1, S_2, \cdots, S_m \subseteq U$

**Output:** minimum size set $C \subseteq [m]$ such that $\bigcup_{i \in C} S_i = U$

## Greedy Algorithm for Set Cover

1: $C \leftarrow \emptyset, U' \leftarrow U$
2: **while** $U' \neq \emptyset$ **do**
3: choose the $i$ that maximizes $|U' \cap S_i|$
4: $C \leftarrow C \cup \{i\}, U' \leftarrow U' \setminus S_i$
5: **return** $C$

- $g$: minimum number of sets needed to cover $U$

**Lemma** Let $u_t, t \in \mathbb{Z}_{\geq 0}$ be the number of uncovered elements after $t$ steps. Then for every $t \geq 1$, we have

$$u_t \leq \left(1 - \frac{1}{g}\right) \cdot u_{t-1}.$$

- $g$: minimum number of sets needed to cover $U$

**Lemma** Let $u_t, t \in \mathbb{Z}_{\geq 0}$ be the number of uncovered elements after $t$ steps. Then for every $t \geq 1$, we have

$$u_t \leq \left(1 - \frac{1}{g}\right) \cdot u_{t-1}.$$

Proof.

- Consider the $g$ sets $S_1^*, S_2^*, \cdots, S_g^*$ in optimum solution
- $S_1^* \cup S_2^* \cup \cdots \cup S_g^* = U$

- $g$: minimum number of sets needed to cover $U$

**Lemma** Let $u_t, t \in \mathbb{Z}_{\geq 0}$ be the number of uncovered elements after $t$ steps. Then for every $t \geq 1$, we have

$$u_t \leq \left(1 - \frac{1}{g}\right) \cdot u_{t-1}.$$

Proof.

- Consider the $g$ sets $S_1^*, S_2^*, \cdots, S_g^*$ in optimum solution
- $S_1^* \cup S_2^* \cup \cdots \cup S_g^* = U$
- at beginning of step $t$, some set in $S_1^*, S_2^*, \cdots, S_g^*$ must contain $\geq \frac{u_{t-1}}{g}$ uncovered elements
- $u_t \leq u_{t-1} - \frac{u_{t-1}}{g} = \left(1 - \frac{1}{g}\right) u_{t-1}$. $\square$

## Proof of $(\ln n + 1)$-approximation.

- Let $t = \lceil g \cdot \ln n \rceil$. $u_0 = n$. Then

$$u_t \leq \left(1 - \frac{1}{g}\right)^{g \cdot \ln n} \cdot n < e^{-\ln n} \cdot n = n \cdot \frac{1}{n} = 1.$$

- So $u_t = 0$, approximation ratio $\leq \frac{\lceil g \cdot \ln n \rceil}{g} \leq \ln n + 1$. $\square$

**Proof of $(\ln n + 1)$-approximation.**

- Let $t = \lceil g \cdot \ln n \rceil$. $u_0 = n$. Then
$$u_t \leq \left(1 - \frac{1}{g}\right)^{g \cdot \ln n} \cdot n < e^{-\ln n} \cdot n = n \cdot \frac{1}{n} = 1.$$
- So $u_t = 0$, approximation ratio $\leq \frac{\lceil g \cdot \ln n \rceil}{g} \leq \ln n + 1$. $\square$

- A more careful analysis gives a $H_n$-approximation, where $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$ is the $n$-th harmonic number.
- $\ln(n+1) < H_n < \ln n + 1$.

### Proof of $(\ln n + 1)$-approximation.

- Let $t = \lceil g \cdot \ln n \rceil$. $u_0 = n$. Then
$$u_t \le \left(1 - \frac{1}{g}\right)^{g \cdot \ln n} \cdot n < e^{-\ln n} \cdot n = n \cdot \frac{1}{n} = 1.$$
- So $u_t = 0$, approximation ratio $\le \frac{\lceil g \cdot \ln n \rceil}{g} \le \ln n + 1$. □

- A more careful analysis gives a $H_n$-approximation, where $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$ is the $n$-th harmonic number.
- $\ln(n+1) < H_n < \ln n + 1$.

### $(1 - c)\ln n$-hardness for any $c = \Omega(1)$

Let $c > 0$ be any constant. There is no polynomial-time $(1 - c)\ln n$-approximation algorithm for set-cover, unless

- NP $\subseteq$ quasi-poly-time, [Lund, Yannakakis 1994; Feige 1998]
- P $=$ NP. [Dinur, Steuer 2014]

# Outline

1. Greedy Algorithms and Matroids
   - Recap: Maximum-Weight Spanning Tree Problem
   - Maximum-Weight Independent Set in Matroids
   - Examples of Matroids
2. Greedy Approximation Algorithms
   - $(\ln n + 1)$-Approximation for Set-Cover
   - $(1 - \frac{1}{e})$-Approximation for Maximum Coverage
   - Submodular Functions
   - $(1 - \frac{1}{e})$-Approximation for Cardinality-Constrained Submodular Maximization

- set cover: use smallest number of sets to cover all elements.
- maximum coverage: use $k$ sets to cover maximum number of elements

- set cover: use smallest number of sets to cover all elements.
- maximum coverage: use $k$ sets to cover maximum number of elements

**Maximum Coverage**

**Input:** $U, |U| = n$: ground set,

$S_1, S_2, \cdots, S_m \subseteq U$, $\quad k \in [m]$

**Output:** $C \subseteq [m], |C| = k$ with the maximum $\bigcup_{i \in C} S_i$

- set cover: use smallest number of sets to cover all elements.
- maximum coverage: use $k$ sets to cover maximum number of elements

**Maximum Coverage**

**Input:** $U, |U| = n$: ground set,

$S_1, S_2, \cdots, S_m \subseteq U$, $\quad k \in [m]$

**Output:** $C \subseteq [m], |C| = k$ with the maximum $\bigcup_{i \in C} S_i$

**Greedy Algorithm for Maximum Coverage**

1: $C \leftarrow \emptyset, U' \leftarrow U$
2: **for** $t \leftarrow 1$ to $k$ **do**
3: choose the $i$ that maximizes $|U' \cap S_i|$
4: $C \leftarrow C \cup \{i\}, U' \leftarrow U' \setminus S_i$
5: **return** $C$

**Theorem** Greedy algorithm gives $(1 - \frac{1}{e})$-approximation for maximum coverage.

**Theorem** Greedy algorithm gives $(1 - \frac{1}{e})$-approximation for maximum coverage.

Proof.

- $o$: max. number of elements that can be covered by $k$ sets.
- $p_t$: $\#$(covered elements) by greedy algorithm after step $t$

**Theorem** Greedy algorithm gives $(1 - \frac{1}{e})$-approximation for maximum coverage.

Proof.

- $o$: max. number of elements that can be covered by $k$ sets.
- $p_t$: #(covered elements) by greedy algorithm after step $t$
- $p_t \geq p_{t-1} + \dfrac{o - p_{t-1}}{k}$

**Theorem** Greedy algorithm gives $(1 - \frac{1}{e})$-approximation for maximum coverage.

Proof.

- $o$: max. number of elements that can be covered by $k$ sets.
- $p_t$: #(covered elements) by greedy algorithm after step $t$
- $p_t \geq p_{t-1} + \dfrac{o - p_{t-1}}{k}$
- $o - p_t \leq o - p_{t-1} - \frac{o-p_{t-1}}{k} = \left(1 - \frac{1}{k}\right)(o - p_{t-1})$

**Theorem** Greedy algorithm gives $(1 - \frac{1}{e})$-approximation for maximum coverage.

Proof.

- $o$: max. number of elements that can be covered by $k$ sets.
- $p_t$: #(covered elements) by greedy algorithm after step $t$
- $p_t \geq p_{t-1} + \dfrac{o - p_{t-1}}{k}$
- $o - p_t \leq o - p_{t-1} - \frac{o - p_{t-1}}{k} = \left(1 - \frac{1}{k}\right)(o - p_{t-1})$
- $o - p_k \leq \left(1 - \frac{1}{k}\right)^k (o - p_0) \leq \frac{1}{e} \cdot o$

**Theorem** Greedy algorithm gives $(1 - \frac{1}{e})$-approximation for maximum coverage.

Proof.

- $o$: max. number of elements that can be covered by $k$ sets.
- $p_t$: $\#$(covered elements) by greedy algorithm after step $t$
- $p_t \geq p_{t-1} + \dfrac{o - p_{t-1}}{k}$
- $o - p_t \leq o - p_{t-1} - \frac{o - p_{t-1}}{k} = \left(1 - \frac{1}{k}\right)(o - p_{t-1})$
- $o - p_k \leq \left(1 - \frac{1}{k}\right)^k (o - p_0) \leq \frac{1}{e} \cdot o$
- $p_k \geq \left(1 - \frac{1}{e}\right) \cdot o$ $\square$

**Theorem** Greedy algorithm gives $(1 - \frac{1}{e})$-approximation for maximum coverage.

Proof.

- $o$: max. number of elements that can be covered by $k$ sets.
- $p_t$: $\#$(covered elements) by greedy algorithm after step $t$
- $p_t \geq p_{t-1} + \dfrac{o - p_{t-1}}{k}$
- $o - p_t \leq o - p_{t-1} - \frac{o-p_{t-1}}{k} = \left(1 - \frac{1}{k}\right)(o - p_{t-1})$
- $o - p_k \leq \left(1 - \frac{1}{k}\right)^k (o - p_0) \leq \frac{1}{e} \cdot o$
- $p_k \geq \left(1 - \frac{1}{e}\right) \cdot o$ $\square$

- The $(1 - \frac{1}{e})$-approximation extends to a more general problem.

# Outline

**Def.** Let $n \in \mathbb{Z}_{>0}$. A set function $f : 2^{[n]} \to \mathbb{R}$ is called submodular if it satisfies one of the following three equivalent conditions:

(1) $\forall A, B \subseteq [n]$:

$$f(A \cup B) + f(A \cap B) \leq f(A) + f(B).$$

(2) $\forall A \subseteq B \subsetneq [n], i \in [n] \setminus B$:

$$f(B \cup \{i\}) - f(B) \leq f(A \cup \{i\}) - f(A).$$

(3) $\forall A \subseteq [n], i, j \in [n] \setminus A, i \neq j$:

$$f(A \cup \{i, j\}) + f(A) \leq f(A \cup \{i\}) + f(A \cup \{j\}).$$

**Def.** Let $n \in \mathbb{Z}_{>0}$. A set function $f : 2^{[n]} \to \mathbb{R}$ is called submodular if it satisfies one of the following three equivalent conditions:

(1) $\forall A, B \subseteq [n]$:

$$f(A \cup B) + f(A \cap B) \le f(A) + f(B).$$

(2) $\forall A \subseteq B \subsetneq [n], i \in [n] \setminus B$:

$$f(B \cup \{i\}) - f(B) \le f(A \cup \{i\}) - f(A).$$

(3) $\forall A \subseteq [n], i, j \in [n] \setminus A, i \neq j$:

$$f(A \cup \{i, j\}) + f(A) \le f(A \cup \{i\}) + f(A \cup \{j\}).$$

- (2): diminishing marginal values: the marginal value by getting $i$ when I have $B$ is at most that when I have $A \subseteq B$.

**Def.** Let $n \in \mathbb{Z}_{>0}$. A set function $f : 2^{[n]} \to \mathbb{R}$ is called submodular if it satisfies one of the following three equivalent conditions:

(1) $\forall A, B \subseteq [n]$:

$$f(A \cup B) + f(A \cap B) \leq f(A) + f(B).$$

(2) $\forall A \subseteq B \subsetneq [n], i \in [n] \setminus B$:

$$f(B \cup \{i\}) - f(B) \leq f(A \cup \{i\}) - f(A).$$

(3) $\forall A \subseteq [n], i, j \in [n] \setminus A, i \neq j$:

$$f(A \cup \{i, j\}) + f(A) \leq f(A \cup \{i\}) + f(A \cup \{j\}).$$

- (2): diminishing marginal values: the marginal value by getting $i$ when I have $B$ is at most that when I have $A \subseteq B$.
- $(1) \Rightarrow (2) \Rightarrow (3)$, $\qquad (3) \Rightarrow (2) \Rightarrow (1)$

# Examples of Sumodular Functions

- linear function: $f(S) = \sum_{i \in S} w_i, \forall S \subseteq [n]$

# Examples of Sumodular Functions

- linear function: $f(S) = \sum_{i \in S} w_i, \forall S \subseteq [n]$
- budget-additive function: $f(S) = \min \left\{ \sum_{i \in S} w_i, B \right\}, \forall S \subseteq [n]$

# Examples of Sumodular Functions

- linear function: $f(S) = \sum_{i \in S} w_i, \forall S \subseteq [n]$
- budget-additive function: $f(S) = \min \left\{ \sum_{i \in S} w_i, B \right\}, \forall S \subseteq [n]$
- coverage function: given sets $S_1, S_2, \cdots, S_n \subseteq \Omega$,

$$f(C) := \left| \bigcup_{i \in C} S_i \right|, \forall C \subseteq [n]$$

# Examples of Sumodular Functions

- linear function: $f(S) = \sum_{i \in S} w_i, \forall S \subseteq [n]$
- budget-additive function: $f(S) = \min \left\{ \sum_{i \in S} w_i, B \right\}, \forall S \subseteq [n]$
- coverage function: given sets $S_1, S_2, \cdots, S_n \subseteq \Omega$,

$$f(C) := \left| \bigcup_{i \in C} S_i \right|, \forall C \subseteq [n]$$

- matroid rank function:

**Def.** Given a matroid $\mathcal{M} = (E, \mathcal{I})$, the rank of any $A \subseteq E$ is defined as

$$r_{\mathcal{M}}(A) = \max \left\{ |A'| : A' \subseteq A, A' \in \mathcal{I} \right\}.$$

The function $r_{\mathcal{M}} : 2^E \to \mathbb{Z}_{\geq 0}$ is called the rank function of $\mathcal{M}$.

# Examples of Sumodular Functions

- linear function: $f(S) = \sum_{i \in S} w_i, \forall S \subseteq [n]$
- budget-additive function: $f(S) = \min \left\{ \sum_{i \in S} w_i, B \right\}, \forall S \subseteq [n]$
- coverage function: given sets $S_1, S_2, \cdots, S_n \subseteq \Omega$,

$$f(C) := \left| \bigcup_{i \in C} S_i \right|, \forall C \subseteq [n]$$

- matroid rank function:

**Def.** Given a matroid $\mathcal{M} = (E, \mathcal{I})$, the rank of any $A \subseteq E$ is defined as

$$r_{\mathcal{M}}(A) = \max \left\{ |A'| : A' \subseteq A, A' \in \mathcal{I} \right\}.$$

The function $r_{\mathcal{M}} : 2^E \to \mathbb{Z}_{\geq 0}$ is called the rank function of $\mathcal{M}$.

- cut function: given graph $G = ([n], E)$

$$f(A) = \left| E(A, [n] \setminus A) \right|, \forall A \subseteq [n]$$

# Examples of Sumodular Functions

- linear function, budget-additive function, coverage function,

# Examples of Sumodular Functions

- linear function, budget-additive function, coverage function,
- matroid rank function, cut function

# Examples of Sumodular Functions

- linear function, budget-additive function, coverage function,
- matroid rank function, cut function
- entropy function: given random variables $X_1, X_2, \cdots, X_n$

$$f(S) := H(X_i : i \in S), \forall S \subseteq [n]$$

# Examples of Sumodular Functions

- linear function, budget-additive function, coverage function,
- matroid rank function, cut function
- entropy function: given random variables $X_1, X_2, \cdots, X_n$

$$f(S) := H(X_i : i \in S), \forall S \subseteq [n]$$

**Def.** A submodular function $f : 2^{[n]} \to \mathbb{R}$ is said to be monotone if $f(A) \leq f(B)$ for every $A \subseteq B \subseteq [n]$.

# Examples of Sumodular Functions

- linear function, budget-additive function, coverage function,
- matroid rank function, cut function
- entropy function: given random variables $X_1, X_2, \cdots, X_n$

$$f(S) := H(X_i : i \in S), \forall S \subseteq [n]$$

**Def.** A submodular function $f : 2^{[n]} \to \mathbb{R}$ is said to be monotone if $f(A) \leq f(B)$ for every $A \subseteq B \subseteq [n]$.

**Def.** A submodular function $f : 2^{[n]} \to \mathbb{R}$ is said to be symmetric if $f(A) = f([n] \setminus A)$ for every $A \subseteq [n]$.

# Examples of Sumodular Functions

- linear function, budget-additive function, coverage function,
- matroid rank function, cut function
- entropy function: given random variables $X_1, X_2, \cdots, X_n$

$$f(S) := H(X_i : i \in S), \forall S \subseteq [n]$$

**Def.** A submodular function $f : 2^{[n]} \to \mathbb{R}$ is said to be monotone if $f(A) \leq f(B)$ for every $A \subseteq B \subseteq [n]$.

**Def.** A submodular function $f : 2^{[n]} \to \mathbb{R}$ is said to be symmetric if $f(A) = f([n] \setminus A)$ for every $A \subseteq [n]$.

- coverage, matroid rank and entropy functions are monotone

# Examples of Sumodular Functions

- linear function, budget-additive function, coverage function,
- matroid rank function, cut function
- entropy function: given random variables $X_1, X_2, \cdots, X_n$

$$f(S) := H(X_i : i \in S), \forall S \subseteq [n]$$

**Def.** A submodular function $f : 2^{[n]} \to \mathbb{R}$ is said to be monotone if $f(A) \leq f(B)$ for every $A \subseteq B \subseteq [n]$.

**Def.** A submodular function $f : 2^{[n]} \to \mathbb{R}$ is said to be symmetric if $f(A) = f([n] \setminus A)$ for every $A \subseteq [n]$.

- coverage, matroid rank and entropy functions are monotone
- cut function is symmetric

# Outline

# $(1-\frac{1}{e})$-Approximation for Submodular Maximization with Cardinality Constraint

**Submodular Maximization under a Cardinality Constraint**

**Input:** An oracle to a non-negative monotone submodular function $f : 2^{[n]} \to \mathbb{R}_{\geq 0}$, $\qquad k \in [n]$

**Output:** A subset $S \subseteq [n]$ with $|S| = k$, so as to maximize $f(S)$

# $(1-\frac{1}{e})$-Approximation for Submodular Maximization with Cardinality Constraint

**Submodular Maximization under a Cardinality Constraint**

**Input:** An oracle to a non-negative monotone submodular function $f: 2^{[n]} \to \mathbb{R}_{\geq 0}$, $k \in [n]$

**Output:** A subset $S \subseteq [n]$ with $|S| = k$, so as to maximize $f(S)$

- We can assume $f(\emptyset) = 0$

# $(1 - \frac{1}{e})$-Approximation for Submodular Maximization with Cardinality Constraint

**Submodular Maximization under a Cardinality Constraint**

**Input:** An oracle to a non-negative monotone submodular function $f : 2^{[n]} \to \mathbb{R}_{\geq 0}$, $k \in [n]$

**Output:** A subset $S \subseteq [n]$ with $|S| = k$, so as to maximize $f(S)$

- We can assume $f(\emptyset) = 0$

**Greedy Algorithm for the Problem**

```
S ← ∅
for t ← 1 to k do
    choose the i that maximizes f(S ∪ {i})
    S ← S ∪ {i}
return S
```

**Theorem** Greedy algorithm gives $(1 - \frac{1}{e})$-approximation for submodular-maximization under a cardinality constraint.

**Theorem** Greedy algorithm gives $(1-\frac{1}{e})$-approximation for submodular-maximization under a cardinality constraint.

Proof.

- $o$: optimum value
- $p_t$: value obtained by greedy algorithm after step $t$
- need to prove: $p_t \geq p_{t-1} + \dfrac{o - p_{t-1}}{k}$
- $o - p_t \leq o - p_{t-1} - \frac{o-p_{t-1}}{k} = \left(1 - \frac{1}{k}\right)(o - p_{t-1})$
- $o - p_k \leq \left(1 - \frac{1}{k}\right)^k (o - p_0) \leq \frac{1}{e} \cdot o$
- $p_k \geq \left(1 - \frac{1}{e}\right) \cdot o$ $\square$

**Def.** A set function $f : 2^{[n]} \to \mathbb{R}_{\geq 0}$ is sub-additive if for every two sets $A, B \subseteq [n]$, we have $f(A \cup B) \leq f(A) + f(B)$.

**Def.** A set function $f : 2^{[n]} \to \mathbb{R}_{\geq 0}$ is sub-additive if for every two sets $A, B \subseteq [n]$, we have $f(A \cup B) \leq f(A) + f(B)$.

**Lemma** A non-negative submodular set function $f : 2^{[n]} \to \mathbb{R}_{\geq 0}$ is sub-additive.

**Def.** A set function $f: 2^{[n]} \to \mathbb{R}_{\geq 0}$ is sub-additive if for every two sets $A, B \subseteq [n]$, we have $f(A \cup B) \leq f(A) + f(B)$.

**Lemma** A non-negative submodular set function $f: 2^{[n]} \to \mathbb{R}_{\geq 0}$ is sub-additive.

*Proof.*

For $A, B \subseteq [n]$, we have $f(A \cup B) + f(A \cap B) \leq f(A) + f(B)$.
So, $f(A \cup B) \leq f(A) + f(B)$ as $f(A \cap B) \geq 0$. $\square$

**Lemma** Let $f : 2^{[n]} \to \mathbb{R}$ be submodular. Let $S \subseteq [n]$, and $f_S(A) = f(S \cup A) - f(S)$ for every $A \subseteq [n]$. ($f_S$ is the marginal value function for set $S$.) Then $f_S$ is also submodular.

**Lemma** Let $f : 2^{[n]} \to \mathbb{R}$ be submodular. Let $S \subseteq [n]$, and $f_S(A) = f(S \cup A) - f(S)$ for every $A \subseteq [n]$. ($f_S$ is the marginal value function for set $S$.) Then $f_S$ is also submodular.

Proof.

- Let $A, B \subseteq [n] \setminus S$; it suffices to consider ground set $[n] \setminus S$.

$$
\begin{aligned}
& f_S(A \cup B) + f_S(A \cap B) - (f_S(A) + f_S(B)) \\
&= f(S \cup A \cup B) - f(S) + f(S \cup (A \cap B)) - f(S) \\
&\quad - \Big( f(S \cup A) - f(S) + f(S \cup B) - f(S) \Big) \\
&= f(S \cup A \cup B) + f(S \cup (A \cap B)) - f(S \cup A) - f(S \cup B) \\
&\leq 0
\end{aligned}
$$

- The last inequality is by $S \cup A \cup B = (S \cup A) \cup (S \cup B)$, $S \cup (A \cap B) = (S \cup A) \cap (S \cup B)$ and submodularity of $f$. $\square$

## Proof of $p_t \geq p_{t-1} + \frac{o - p_{t-1}}{k}$.

- $S^* \subseteq [n]$: optimum set, $|S^*| = k$, $o = f(S^*)$
- $S$: set chosen by the algorithm at beginning of time step $t$
  $|S| = t - 1$, $p_{t-1} = f(S)$

## Proof of $p_t \geq p_{t-1} + \frac{o-p_{t-1}}{k}$.

- $S^* \subseteq [n]$: optimum set, $|S^*| = k$, $o = f(S^*)$
- $S$: set chosen by the algorithm at beginning of time step $t$
  $|S| = t - 1$, $p_{t-1} = f(S)$
- $f_S$ is submodular and thus sub-additive

$$f_S(S^*) \leq \sum_{i \in S^*} f_S(i) \quad \Rightarrow \quad \exists i \in S^*, f_S(i) \geq \frac{1}{k} f_S(S^*)$$

## Proof of $p_t \geq p_{t-1} + \frac{o - p_{t-1}}{k}$.

- $S^* \subseteq [n]$: optimum set, $|S^*| = k$, $o = f(S^*)$
- $S$: set chosen by the algorithm at beginning of time step $t$
  $|S| = t - 1$, $p_{t-1} = f(S)$
- $f_S$ is submodular and thus sub-additive

$$f_S(S^*) \leq \sum_{i \in S^*} f_S(i) \quad \Rightarrow \quad \exists i \in S^*, f_S(i) \geq \frac{1}{k} f_S(S^*)$$

- for the $i$, we have

$$f(S \cup \{i\}) - f(S) \geq \frac{1}{k}(f(S^*) - f(S))$$

$$p_t \geq f(S \cup \{i\}) \geq p_{t-1} + \frac{1}{k}(o - p_{t-1}) \qquad \square$$

Submodular Maximization for Monotone Functions:

| Constraint | Approx. | Hardness | Technique |
|---|---|---|---|
| $\|S\| \leq k$ | $1 - 1/e$ | $1 - 1/e$ | greedy |
| matroid | $1 - 1/e$ | $1 - 1/e$ | multilinear ext. |
| $O(1)$ knapsacks | $1 - 1/e$ | $1 - 1/e$ | multilinear ext. |
| $k$ matroids | $k + \epsilon$ | $\Omega(k/\log k)$ | local search |
| $k$ matroids<br>$O(1)$ knapsacks | $O(k)$ | $\Omega(k/\log k)$ | multilinear ext. |

Submodular Maximization for Non-Monotone Functions:

| Constraint | Approx. | Hardness | Technique |
|---|---|---|---|
| Unconstrained | $1/2$ | $1/2$ | combinatorial |
| matroid | $1/e$ | $0.48$ | multilinear ext. |
| $O(1)$ knapsacks | $1/e$ | $0.49$ | multilinear ext. |
| $k$ matroids | $k + O(1)$ | $\Omega(k/\log k)$ | local search |
| $k$ matroids<br>$O(1)$ knapsacks | $O(k)$ | $\Omega(k/\log k)$ | multilinear ext. |

From Prof. Jan Vodrak's slides "Optimization of Submodular Functions"

# Submodular Minimization

| Constraint | Approx. | Hardness | Technique |
| --- | --- | --- | --- |
| Unconstrained | $1$ | $1$ | combinatorial |
| $\|S\| \geq k$, Monotone | $\tilde{O}(\sqrt{n})$ * | $\Omega(\sqrt{n})$ * | combinatorial |

- * bounds are for query complexity under oracle model.