

# Matroid\*

Xiaofeng Gao

Department of Computer Science and Engineering  
Shanghai Jiao Tong University, P.R.China

Algorithm Course @ Shanghai Jiao Tong University

---

\*Special Thanks is given to Prof. Ding-Zhu Du for sharing his teaching materials.

# Outline

- 1 Matroid
  - Independent System
  - Matroid
- 2 Greedy Algorithm on Matroid
  - $u(F)$  and  $v(F)$
  - Greedy-MAX Algorithm
- 3 Task Scheduling Problem
  - Unit-Time Task Scheduling
  - Greedy Approach

# Independent System

Consider a finite set  $S$  and a collection  $\mathbf{C}$  of subsets of  $S$ .

# Independent System

Consider a finite set  $S$  and a collection  $\mathbf{C}$  of subsets of  $S$ .

$(S, \mathbf{C})$  is called an **independent system** if

$$A \subset B, B \in \mathbf{C} \Rightarrow A \in \mathbf{C}.$$

# Independent System

Consider a finite set  $S$  and a collection  $\mathbf{C}$  of subsets of  $S$ .

$(S, \mathbf{C})$  is called an **independent system** if

$$A \subset B, B \in \mathbf{C} \Rightarrow A \in \mathbf{C}.$$

We say that  $\mathbf{C}$  is **hereditary** if it satisfies this property.

Each subset in  $\mathbf{C}$  is called an **independent subset**.

Note that the empty set  $\emptyset$  is necessarily a member of  $\mathbf{C}$ .

# An Example

**Example:** Given an undirected graph  $G = (V, E)$ , Define  $\mathbf{H}$  as:

$\mathbf{H} = \{F \subseteq E \mid F \text{ is a Hamiltonian circuit or a union of disjoint paths}\}.$

Then  $(E, \mathbf{H})$  is an independent system.

# An Example

**Example:** Given an undirected graph  $G = (V, E)$ , Define  $\mathbf{H}$  as:

$\mathbf{H} = \{F \subseteq E \mid F \text{ is a Hamiltonian circuit or a union of disjoint paths}\}.$

Then  $(E, \mathbf{H})$  is an independent system.

**Proof:** (Hereditary)

Given any  $F \in \mathbf{H}$  and  $P \subset F$ .

Since  $F$  is either a Hamiltonian circuit or a union of disjoint path,  $P$  must be a union of disjoint paths, which obviously belongs to  $\mathbf{H}$ .  $\square$

# Outline

- 1 Matroid
  - Independent System
  - Matroid
- 2 Greedy Algorithm on Matroid
  - $u(F)$  and  $v(F)$
  - Greedy-MAX Algorithm
- 3 Task Scheduling Problem
  - Unit-Time Task Scheduling
  - Greedy Approach



# Matroid

An independent system  $(S, \mathbf{C})$  is a **matroid** if it satisfies the **exchange property**:

$$A, B \in \mathbf{C} \text{ and } |A| < |B| \Rightarrow \exists x \in B \setminus A \text{ such that } A \cup \{x\} \in \mathbf{C}.$$

# Matroid

An independent system  $(S, \mathbf{C})$  is a **matroid** if it satisfies the **exchange property**:

$$A, B \in \mathbf{C} \text{ and } |A| < |B| \Rightarrow \exists x \in B \setminus A \text{ such that } A \cup \{x\} \in \mathbf{C}.$$

Thus a matroid should satisfy two requirements: **hereditary** and **exchange property**.

# Matric Matroid

**Matric Matroid:** Consider a matrix  $M$ . Let  $S$  be the set of row vectors of  $M$  and  $\mathbf{C}$  the collection of all linearly independent subsets of  $S$ . Then  $(S, \mathbf{C})$  is a matroid.

# Matric Matroid

**Matric Matroid:** Consider a matrix  $M$ . Let  $S$  be the set of row vectors of  $M$  and  $\mathbf{C}$  the collection of all linearly independent subsets of  $S$ . Then  $(S, \mathbf{C})$  is a matroid.

**Proof:**

- **Hereditary:** If  $A \subset B$  and  $B \in \mathbf{C}$ , meaning  $B$  is a linearly independent subset of row vectors of  $M$ , then  $A$  must be linearly independent.

# Matric Matroid

**Matric Matroid:** Consider a matrix  $M$ . Let  $S$  be the set of row vectors of  $M$  and  $\mathbf{C}$  the collection of all linearly independent subsets of  $S$ . Then  $(S, \mathbf{C})$  is a matroid.

## Proof:

- **Hereditary:** If  $A \subset B$  and  $B \in \mathbf{C}$ , meaning  $B$  is a linearly independent subset of row vectors of  $M$ , then  $A$  must be linearly independent.
- **Exchange Property:** The exchange property is a well known fact for **linearly independence**. Say, If  $A, B$  are sets of linearly independent rows of  $M$ , and  $|A| < |B|$ , then  $\dim \text{span}(A) < \dim \text{span}(B)$ . Choose a row  $x$  in  $B$  that is not contained in  $\text{span}(A)$ . Then  $A \cup \{x\}$  is a linearly independent subset of rows of  $M$ .  $\square$

# Graphic Matroid

**Graphic Matroid  $M_G$ :** Consider a (undirected) graph  $G = (V, E)$ . Let  $S = E$  and  $\mathbf{C}$  the collection of all edge sets each of which induces an acyclic subgraph of  $G$ . Then  $M_G = (S, \mathbf{C})$  is a matroid.

# Graphic Matroid

**Graphic Matroid  $M_G$ :** Consider a (undirected) graph  $G = (V, E)$ . Let  $S = E$  and  $\mathbf{C}$  the collection of all edge sets each of which induces an acyclic subgraph of  $G$ . Then  $M_G = (S, \mathbf{C})$  is a matroid.

**Proof:**

- **Hereditary:** If  $B$  is an edge set which induces an acyclic subgraph of  $G$ , obviously any  $A \subset B$  induces an acyclic subgraph.

# Graphic Matroid

**Graphic Matroid  $M_G$ :** Consider a (undirected) graph  $G = (V, E)$ . Let  $S = E$  and  $\mathbf{C}$  the collection of all edge sets each of which induces an acyclic subgraph of  $G$ . Then  $M_G = (S, \mathbf{C})$  is a matroid.

## Proof:

- **Hereditary:** If  $B$  is an edge set which induces an acyclic subgraph of  $G$ , obviously any  $A \subset B$  induces an acyclic subgraph.
- **Exchange Property:** consider  $A, B \in \mathbf{C}$  with  $|A| < |B|$ .

Note that  $(V, A)$  has  $|V| - |A|$  connected components and  $(V, B)$  has  $|V| - |B|$  connected components.

Hence,  $B$  has an edge  $e$  connecting two connected components of  $(V, A)$ , which implies  $A \cup \{e\} \in \mathbf{C}$ .  $\square$



# More Examples

**Uniform matroid**  $U_{k,n}$ : A subset  $X \subseteq \{1, 2, \dots, n\}$  is independent if and only if  $|X| \leq k$ .

# More Examples

**Uniform matroid**  $U_{k,n}$ : A subset  $X \subseteq \{1, 2, \dots, n\}$  is independent if and only if  $|X| \leq k$ .

**Cographic matroid**  $M_G^*$ : Let  $G = (V, E)$  be an arbitrary undirected graph. A subset  $I \subseteq E$  is independent if the complementary subgraph  $(V, E \setminus I)$  of  $G$  is connected.

## More Examples

**Uniform matroid**  $U_{k,n}$ : A subset  $X \subseteq \{1, 2, \dots, n\}$  is independent if and only if  $|X| \leq k$ .

**Cographic matroid**  $M_G^*$ : Let  $G = (V, E)$  be an arbitrary undirected graph. A subset  $I \subseteq E$  is independent if the complementary subgraph  $(V, E \setminus I)$  of  $G$  is connected.

**Matching matroid**: Let  $G = (V, E)$  be an arbitrary undirected graph. A subset  $I \subseteq E$  is independent if there is a matching in  $G$  that covers  $I$ .

## More Examples

**Uniform matroid**  $U_{k,n}$ : A subset  $X \subseteq \{1, 2, \dots, n\}$  is independent if and only if  $|X| \leq k$ .

**Cographic matroid**  $M_G^*$ : Let  $G = (V, E)$  be an arbitrary undirected graph. A subset  $I \subseteq E$  is independent if the complementary subgraph  $(V, E \setminus I)$  of  $G$  is connected.

**Matching matroid**: Let  $G = (V, E)$  be an arbitrary undirected graph. A subset  $I \subseteq E$  is independent if there is a matching in  $G$  that covers  $I$ .

**Disjoint path matroid**: Let  $G = (V, E)$  be an arbitrary directed graph, and let  $s$  be a fixed vertex of  $G$ . A subset  $I \subseteq E$  is independent if and only if there are edge-disjoint paths from  $s$  to each vertex in  $I$ .

# Notation

The word “matroid” is due to **Hassler Whitney**<sup>[1]</sup>, who first studied matric matroid (1935).

Actually the greedy algorithm first appeared in the combinatorial optimization literature by **Jack Edmonds**<sup>[2]</sup> (1971).

An extension of matroid theory to **greedoid** theory was pioneered by **Korte and Lovász**, who greatly generalize the theory (1981-1984).



Hassler Whitney  
(1907-1989)  
**Wolf Prize** (1983)

[1] Hassler Whitney. On the abstract properties of linear dependence. *American Journal of Mathematics*, 57:509-533, 1935.

[2] Jack Edmonds. Matroids and the greedy algorithm. *Mathematical Programming*, 1:126-136, 1971.

# Outline

- 1 Matroid
  - Independent System
  - Matroid
- 2 Greedy Algorithm on Matroid
  - $u(F)$  and  $v(F)$
  - Greedy-MAX Algorithm
- 3 Task Scheduling Problem
  - Unit-Time Task Scheduling
  - Greedy Approach

# Extension

An element  $x$  is called an **extension** of an independent subset  $I$  if  $x \notin I$  and  $I \cup \{x\}$  is independent.

An independent subset is **maximal** if it has no extension.

For any subset  $F \subseteq S$ , an independent subset  $I \subseteq F$  is maximal in  $F$  if  $I$  has no extension in  $F$ .

# Maximal Independent Subset

Consider an independent system  $(S, \mathbf{C})$ . For  $F \subseteq S$ , define

$$u(F) = \min\{|I| \mid I \text{ is a maximal independent subset of } F\}$$

$$v(F) = \max\{|I| \mid I \text{ is an independent subset of } F\}$$



# An Example: Maximal Independent Vertex Set

**Independent Vertex Set:** Given a graph  $G = (V, E)$ , an independent vertex set is a subset  $I \subseteq V$  such that any two vertices in  $I$  are not directly connected.

# An Example: Maximal Independent Vertex Set

**Independent Vertex Set:** Given a graph  $G = (V, E)$ , an independent vertex set is a subset  $I \subseteq V$  such that any two vertices in  $I$  are not directly connected.

$(V, \mathbf{I})$  is an independent system, where  $\mathbf{I}$  is the collection of all independent vertex sets.

# An Example: Maximal Independent Vertex Set

**Independent Vertex Set:** Given a graph  $G = (V, E)$ , an independent vertex set is a subset  $I \subseteq V$  such that any two vertices in  $I$  are not directly connected.

$(V, \mathbf{I})$  is an independent system, where  $\mathbf{I}$  is the collection of all independent vertex sets.

$I$  is **maximal** if  $\forall v \in V \setminus I, I \cup \{v\}$  is not an independent vertex set any more. ( $u(V)$  is the cardinality value of such an  $I$  with minimum cardinality.)

# An Example: Maximal Independent Vertex Set

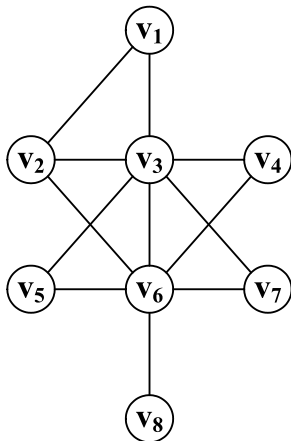
**Independent Vertex Set:** Given a graph  $G = (V, E)$ , an independent vertex set is a subset  $I \subseteq V$  such that any two vertices in  $I$  are not directly connected.

$(V, \mathbf{I})$  is an independent system, where  $\mathbf{I}$  is the collection of all independent vertex sets.

$I$  is **maximal** if  $\forall v \in V \setminus I, I \cup \{v\}$  is not an independent vertex set any more. ( $u(V)$  is the cardinality value of such an  $I$  with minimum cardinality.)

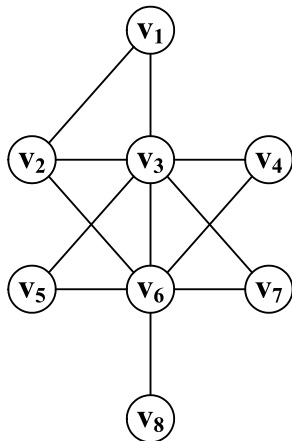
$I$  is **maximum** if it is with the largest cardinality among all maximal independent vertex set. ( $v(V)$  is the cardinality value of any maximum independent vertex set  $I$ .)

# An Independent Vertex Set Instance

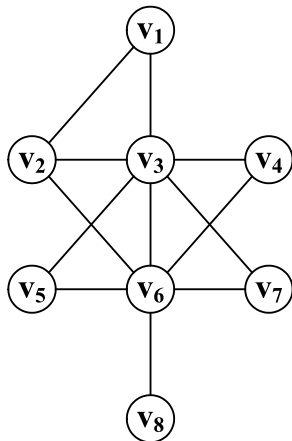


# An Independent Vertex Set Instance

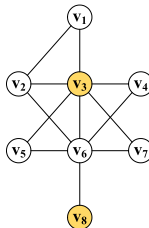
## Maximal Independent Vertex Set



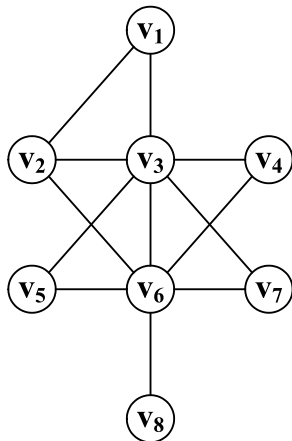
# An Independent Vertex Set Instance



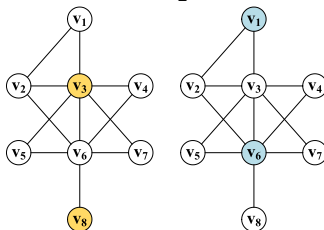
## Maximal Independent Vertex Set



# An Independent Vertex Set Instance

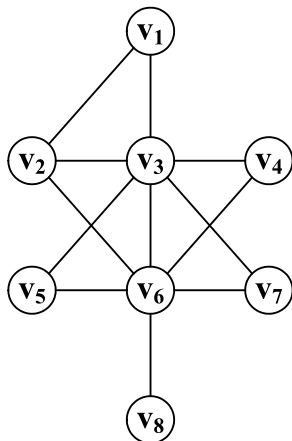


## Maximal Independent Vertex Set

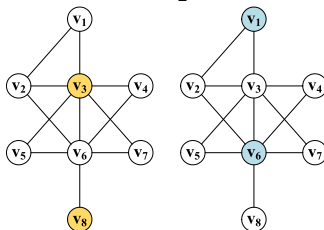




# An Independent Vertex Set Instance

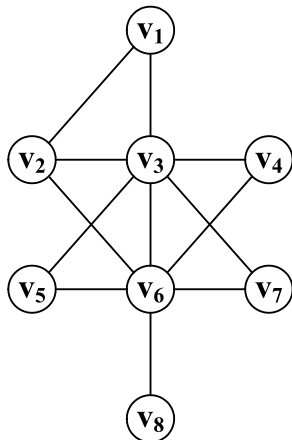


## Maximal Independent Vertex Set

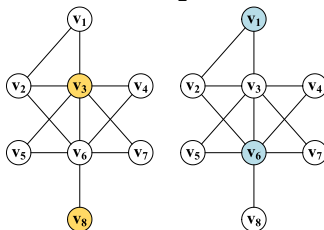


$$u(V) = 2$$

# An Independent Vertex Set Instance



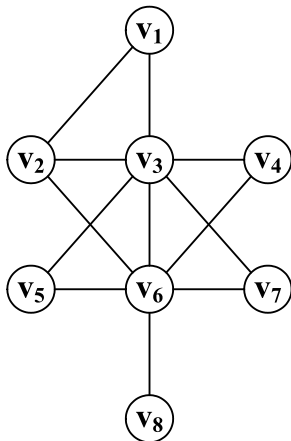
## Maximal Independent Vertex Set



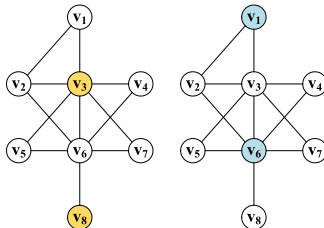
$$u(V) = 2$$

## Maximum Independent Vertex Set

# An Independent Vertex Set Instance

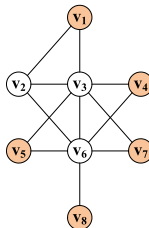


## Maximal Independent Vertex Set

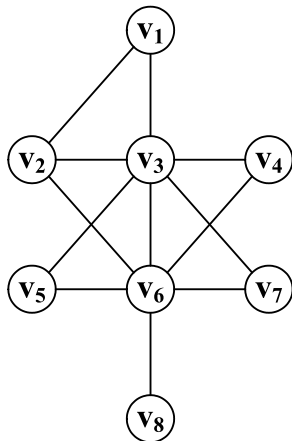


$$u(V) = 2$$

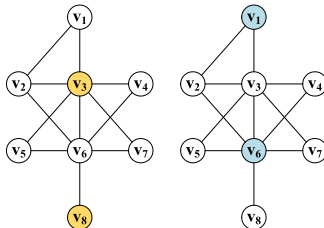
## Maximum Independent Vertex Set



# An Independent Vertex Set Instance

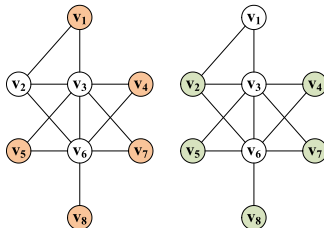


## Maximal Independent Vertex Set

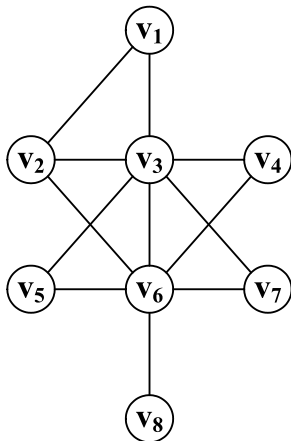


$$u(V) = 2$$

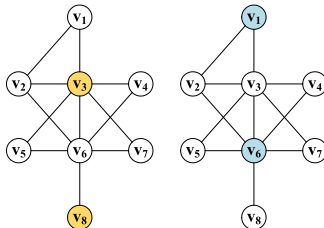
## Maximum Independent Vertex Set



# An Independent Vertex Set Instance

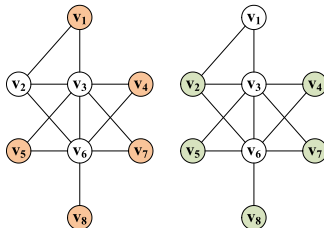


## Maximal Independent Vertex Set



$$u(V) = 2$$

## Maximum Independent Vertex Set



$$v(V) = 5$$

# Matroid Theorem

**Theorem:** An independent system  $(S, \mathbf{C})$  is a matroid if and only if for any  $F \subseteq S$ ,  $u(F) = v(F)$ .

# Matroid Theorem

**Theorem:** An independent system  $(S, \mathbf{C})$  is a matroid if and only if for any  $F \subseteq S$ ,  $u(F) = v(F)$ .

**Proof:** ( $\Rightarrow$ ) For two maximal independent subsets  $A$  and  $B$ , if  $|A| < |B|$ , then there must exist an  $x \in B$  such that  $A \cup \{x\} \in \mathbf{C}$ , contradicting the maximality of  $A$ .

# Matroid Theorem

**Theorem:** An independent system  $(S, \mathbf{C})$  is a matroid if and only if for any  $F \subseteq S$ ,  $u(F) = v(F)$ .

**Proof:** ( $\Rightarrow$ ) For two maximal independent subsets  $A$  and  $B$ , if  $|A| < |B|$ , then there must exist an  $x \in B$  such that  $A \cup \{x\} \in \mathbf{C}$ , contradicting the maximality of  $A$ .

( $\Leftarrow$ ) Consider two independent subsets  $A$  and  $B$  with  $|A| < |B|$ . Set  $F = A \cup B$ . Then every maximal independent subset  $I$  of  $F$  has size  $|I| \geq |B| > |A|$ . Hence,  $A$  cannot be a maximal independent subset of  $F$ , so  $A$  has an extension in  $F$ .  $\square$



# Matroid Theorem

**Theorem:** An independent system  $(S, \mathbf{C})$  is a matroid if and only if for any  $F \subseteq S$ ,  $u(F) = v(F)$ .

**Proof:** ( $\Rightarrow$ ) For two maximal independent subsets  $A$  and  $B$ , if  $|A| < |B|$ , then there must exist an  $x \in B$  such that  $A \cup \{x\} \in \mathbf{C}$ , contradicting the maximality of  $A$ .

( $\Leftarrow$ ) Consider two independent subsets  $A$  and  $B$  with  $|A| < |B|$ . Set  $F = A \cup B$ . Then every maximal independent subset  $I$  of  $F$  has size  $|I| \geq |B| > |A|$ . Hence,  $A$  cannot be a maximal independent subset of  $F$ , so  $A$  has an extension in  $F$ .  $\square$

Thus the definition of matroid could be either by exchange property or by  $u(F) = v(F)$ .

# Corollary

**Corollary:** All maximal independent subsets in a matroid have the same size.

# Corollary

**Corollary:** All maximal independent subsets in a matroid have the same size.

**Proof:** (Contradiction) Suppose  $A$  and  $B$  are two maximal independent subsets with  $|A| < |B|$ , then  $A$  must have an extension in  $A \cup B$ , which violates its maximality property.  $\square$

# Outline

- 1 Matroid
  - Independent System
  - Matroid
- 2 Greedy Algorithm on Matroid
  - $u(F)$  and  $v(F)$
  - Greedy-MAX Algorithm
- 3 Task Scheduling Problem
  - Unit-Time Task Scheduling
  - Greedy Approach

# Basis

In a matroid  $(S, \mathbf{C})$ , every maximal independent subset of  $S$  is called a **basis** (some reference call it **base**).

# Basis

In a matroid  $(S, \mathbf{C})$ , every maximal independent subset of  $S$  is called a **basis** (some reference call it **base**).

**Example:** In a graphic matroid  $M_G = (S, \mathbf{C})$ ,  $A \in \mathbf{C}$  is a basis if and only if  $A$  is a spanning tree.

# Weighted Independent System

An independent system  $(S, \mathbf{C})$  with a nonnegative function  $c : S \rightarrow \mathbb{R}^+$  is called a **weighted independent system**.

In a **weighted matroid**, there is a maximum weight independent subset which is a basis.

# Weighted Independent System

An independent system  $(S, \mathbf{C})$  with a nonnegative function  $c : S \rightarrow \mathbb{R}^+$  is called a **weighted independent system**.

In a **weighted matroid**, there is a maximum weight independent subset which is a basis.

Note: we can define the associated strictly positive weight function  $c(\cdot)$  to each element  $x \in S$ . Thus the weight function extends to subsets of  $S$  by summation:

$$c(A) = \sum_{x \in A} c(x).$$



## Greedy Algorithm for Independent System

We give a common greedy algorithm for any independent system  $(S, \mathbf{C})$  with cost function  $c$ , solving a maximization problem as:

# Greedy Algorithm for Independent System

We give a common greedy algorithm for any independent system  $(S, \mathbf{C})$  with cost function  $c$ , solving a maximization problem as:

$$\begin{array}{ll} \text{maximize} & c(I) \\ \text{subject to} & I \in \mathbf{C} \end{array}$$

# Greedy Algorithm for Independent System

We give a common greedy algorithm for any independent system  $(S, \mathbf{C})$  with cost function  $c$ , solving a maximization problem as:

$$\begin{array}{ll} \text{maximize} & c(I) \\ \text{subject to} & I \in \mathbf{C} \end{array}$$

The algorithm is written as:

---

**Algorithm 1:** Greedy-MAX

---

- 1 Sort all elements in  $S$  into ordering  $c(x_1) \geq c(x_2) \geq \dots \geq c(x_n)$ ;
- 2  $A \leftarrow \emptyset$ ;
- 3 **for**  $i = 1$  **to**  $n$  **do**
- 4     **if**  $A \cup \{x_i\} \in \mathbf{C}$  **then**
- 5          $A \leftarrow A \cup \{x_i\}$ ;
- 6 output  $A$ ;

# Time Complexity

Let  $n = |S|$  = number elements in  $S$ . Then sorting the elements of  $S$  requires  $O(n \log n)$ .

The **for-loop** iterates  $n$  times. In the body of the loop one needs to check whether  $A \cup \{x\}$  is in  $\mathbf{C}$ . If each check takes  $f(n)$  time, then the loop takes  $O(nf(n))$  time.

Thus, Greedy-MAX takes  $O(n \log n + nf(n))$  time.

# Greedy Theorem for Independent System

**Theorem:** Consider a weighted independent system. Let  $A_G$  be obtained by the Greedy Algorithm. Let  $A^*$  be an optimal solution. Then

$$1 \leq \frac{c(A^*)}{c(A_G)} \leq \max_{F \subseteq S} \frac{v(F)}{u(F)}$$

where  $v(F)$  is the maximum size of independent subset in  $F$  and  $u(F)$  is the minimum size of maximal independent subset in  $F$ .

# Proof

Denote  $S_i = \{x_1, \dots, x_i\}$ . (Sorted in nonincreasing order). Then we prove that  $S_i \cap A_G$  is a maximal independent subset of  $S_i$ .

# Proof

Denote  $S_i = \{x_1, \dots, x_i\}$ . (Sorted in nonincreasing order). Then we prove that  $S_i \cap A_G$  is a maximal independent subset of  $S_i$ .

(By Contradiction) If not, there exists an element  $x_j \in S_i \setminus A_G$  such that  $(S_i \cap A_G) \cup \{x_j\}$  is independent.

# Proof

Denote  $S_i = \{x_1, \dots, x_i\}$ . (Sorted in nonincreasing order). Then we prove that  $S_i \cap A_G$  is a maximal independent subset of  $S_i$ .

(By Contradiction) If not, there exists an element  $x_j \in S_i \setminus A_G$  such that  $(S_i \cap A_G) \cup \{x_j\}$  is independent.

However, at the beginning of the  $j$ th iteration of the loop in the Greedy-Max,  $x_j$  must be selected into  $A_G^{j-1}$ . (Since  $A_G^{j-1} \cup \{x_j\}$  must be a subset of  $(S_j \cap A_G) \cup \{x_j\}$ , and hence, is an independent set.)



# Proof

Denote  $S_i = \{x_1, \dots, x_i\}$ . (Sorted in nonincreasing order). Then we prove that  $S_i \cap A_G$  is a maximal independent subset of  $S_i$ .

(By Contradiction) If not, there exists an element  $x_j \in S_i \setminus A_G$  such that  $(S_i \cap A_G) \cup \{x_j\}$  is independent.

However, at the beginning of the  $j$ th iteration of the loop in the Greedy-Max,  $x_j$  must be selected into  $A_G^{j-1}$ . (Since  $A_G^{j-1} \cup \{x_j\}$  must be a subset of  $(S_j \cap A_G) \cup \{x_j\}$ , and hence, is an independent set.)

Therefore we have  $|S_i \cap A_G| \geq u(S_i)$ .

# Proof

Denote  $S_i = \{x_1, \dots, x_i\}$ . (Sorted in nonincreasing order). Then we prove that  $S_i \cap A_G$  is a maximal independent subset of  $S_i$ .

(By Contradiction) If not, there exists an element  $x_j \in S_i \setminus A_G$  such that  $(S_i \cap A_G) \cup \{x_j\}$  is independent.

However, at the beginning of the  $j$ th iteration of the loop in the Greedy-Max,  $x_j$  must be selected into  $A_G^{j-1}$ . (Since  $A_G^{j-1} \cup \{x_j\}$  must be a subset of  $(S_j \cap A_G) \cup \{x_j\}$ , and hence, is an independent set.)

Therefore we have  $|S_i \cap A_G| \geq u(S_i)$ .

Moreover, since  $S_i \cap A^*$  is independent, we have  $|S_i \cap A^*| \leq v(S_i)$ .

## Proof (2)

Now we express  $c(A_G)$  and  $c(A^*)$  in terms of  $|S_i \cap A_G|$  and  $|S_i \cap A^*|$ .

## Proof (2)

Now we express  $c(A_G)$  and  $c(A^*)$  in terms of  $|S_i \cap A_G|$  and  $|S_i \cap A^*|$ .

$$\text{Firstly, } |S_i \cap A_G| - |S_{i-1} \cap A_G| = \begin{cases} 1, & \text{if } x_i \in A_G, \\ 0, & \text{otherwise.} \end{cases}$$

## Proof (2)

Now we express  $c(A_G)$  and  $c(A^*)$  in terms of  $|S_i \cap A_G|$  and  $|S_i \cap A^*|$ .

$$\text{Firstly, } |S_i \cap A_G| - |S_{i-1} \cap A_G| = \begin{cases} 1, & \text{if } x_i \in A_G, \\ 0, & \text{otherwise.} \end{cases}$$

Therefore,

$$\begin{aligned} c(A_G) &= \sum_{x_i \in A_G} c(x_i) \\ &= c(x_1) \cdot |S_1 \cap A_G| + \sum_{i=2}^n c(x_i) \cdot (|S_i \cap A_G| - |S_{i-1} \cap A_G|) \\ &= \sum_{i=1}^{n-1} |S_i \cap A_G| \cdot (c(x_i) - c(x_{i+1})) + |S_n \cap A_G| \cdot c(x_n) \end{aligned}$$

## Proof (2)

Now we express  $c(A_G)$  and  $c(A^*)$  in terms of  $|S_i \cap A_G|$  and  $|S_i \cap A^*|$ .

$$\text{Firstly, } |S_i \cap A_G| - |S_{i-1} \cap A_G| = \begin{cases} 1, & \text{if } x_i \in A_G, \\ 0, & \text{otherwise.} \end{cases}$$

Therefore,

$$\begin{aligned} c(A_G) &= \sum_{x_i \in A_G} c(x_i) \\ &= c(x_1) \cdot |S_1 \cap A_G| + \sum_{i=2}^n c(x_i) \cdot (|S_i \cap A_G| - |S_{i-1} \cap A_G|) \\ &= \sum_{i=1}^{n-1} |S_i \cap A_G| \cdot (c(x_i) - c(x_{i+1})) + |S_n \cap A_G| \cdot c(x_n) \end{aligned}$$

Similarly,

$$c(A^*) = \sum_{i=1}^{n-1} |S_i \cap A^*| \cdot (c(x_i) - c(x_{i+1})) + |S_n \cap A^*| \cdot c(x_n)$$

## Proof (3)

Define  $\rho = \max_{F \subseteq S} \frac{v(F)}{u(F)}$ . Then we have

$$\begin{aligned} c(A^*) &= \sum_{i=1}^{n-1} |S_i \cap A^*| \cdot (c(x_i) - c(x_{i+1})) + |S_n \cap A^*| \cdot c(x_n) \\ &\leq \sum_{i=1}^{n-1} v(S_i) \cdot (c(x_i) - c(x_{i+1})) + v(S_n) \cdot c(x_n) \\ &\leq \sum_{i=1}^{n-1} \rho \cdot u(S_i) \cdot (c(x_i) - c(x_{i+1})) + \rho \cdot u(S_n) \cdot c(x_n) \\ &\leq \sum_{i=1}^{n-1} \rho \cdot |S_i \cap A_G| \cdot (c(x_i) - c(x_{i+1})) + \rho \cdot |S_n \cap A_G| \cdot c(x_n) \\ &= \rho \cdot c(A_G). \end{aligned}$$

## Proof (4)

Thus,

$$1 \leq \frac{c(A^*)}{c(A_G)} \leq \rho = \max_{F \subseteq S} \frac{v(F)}{u(F)}. \quad \square$$

**Note:** This theorem implies that if we use Greedy-MAX to find a subset  $I \in \mathbf{C}$  with the maximum weight, the result will not be that bad.

It is bounded by the size of the **maximum size independent subset** of  $S$  versus the **minimum size maximal independent subset** of  $S$ . Say,

$$\frac{1}{\rho} \cdot c(A^*) \leq c(A_G) \leq c(A^*).$$



# Corollary for Matroid

**Corollary:** If  $(S, \mathbf{C}, c)$  is a weighted matroid, then Greedy-MAX algorithm performs the optimal solution.

# Corollary for Matroid

**Corollary:** If  $(S, \mathbf{C}, c)$  is a weighted matroid, then Greedy-MAX algorithm performs the optimal solution.

**Proof:** Since in a matroid for any  $F \subseteq S$ ,  $u(F) = v(F)$ , the corollary can be directly derived from the previous theorem.  $\square$

# Minimizing or Maximizing?

Let  $M = (S, \mathbf{C})$  be a matroid.

The algorithm Greedy-MAX( $M, c$ ) returns a set  $I \in \mathbf{C}$  **maximizing** the weight  $c(I)$ .

If we would like to find a set  $I \in \mathbf{C}$  with minimal weight, then we can use Greedy-MAX with weight function

$$c^*(x_i) = m - c(x_i), \quad \forall x_i \in I,$$

where  $m$  is a real number such that  $m \geq \max_{x_i \in S} c(x_i)$ .

# An Example: Graphic Matroid

**Minimum Spanning Tree:** For a connected graph  $G = (V, E)$  with edge weight  $c : E \rightarrow \mathbb{R}^+$ , computing the minimum spanning tree.

# An Example: Graphic Matroid

**Minimum Spanning Tree:** For a connected graph  $G = (V, E)$  with edge weight  $c : E \rightarrow \mathbb{R}^+$ , computing the minimum spanning tree.

If we set  $c_{\max} = \max_{e \in E} c(e)$  and define  $c^*(e) = c_{\max} - c(e)$ , for every edge  $e \in E$ , then the MST problem is equivalent to find the maximum weight independent subset in the graphic matroid  $M_G$ .

# An Example: Graphic Matroid

**Minimum Spanning Tree:** For a connected graph  $G = (V, E)$  with edge weight  $c : E \rightarrow \mathbb{R}^+$ , computing the minimum spanning tree.

If we set  $c_{\max} = \max_{e \in E} c(e)$  and define  $c^*(e) = c_{\max} - c(e)$ , for every edge  $e \in E$ , then the MST problem is equivalent to find the maximum weight independent subset in the graphic matroid  $M_G$ .

This is because every maximum weight independent set is a base, i.e., a spanning tree which contains a fixed number of edges.

$$c^*(A) = (|V| - 1)c_{\max} - c(A).$$

An independent subset that maximizes the quantity  $c^*(A)$  must minimize  $c(A)$ .

## An Example (Cont.)

Thus if we implement Greedy-MAX to  $M_G$ , we will achieve a solution exactly the same as the **Kruskal** Algorithm.

## An Example (Cont.)

Thus if we implement Greedy-MAX to  $M_G$ , we will achieve a solution exactly the same as the **Kruskal** Algorithm.

We could also use the property of Greedy-MAX on Matroid to validate the correctness of the Kruskal algorithm.



## More Examples

**Matric matroid:** Given a matrix  $M$ , compute a subset of vectors of maximum total weight that span the column space of  $M$ .

## More Examples

**Matric matroid:** Given a matrix  $M$ , compute a subset of vectors of maximum total weight that span the column space of  $M$ .

**Uniform matroid:** Given a set of weighted objects, compute its  $k$  largest elements.

## More Examples

**Matric matroid:** Given a matrix  $M$ , compute a subset of vectors of maximum total weight that span the column space of  $M$ .

**Uniform matroid:** Given a set of weighted objects, compute its  $k$  largest elements.

**Cographic matroid:** Given a graph with weighted edges, compute its minimum spanning tree.

## More Examples

**Matric matroid:** Given a matrix  $M$ , compute a subset of vectors of maximum total weight that span the column space of  $M$ .

**Uniform matroid:** Given a set of weighted objects, compute its  $k$  largest elements.

**Cographic matroid:** Given a graph with weighted edges, compute its minimum spanning tree.

**Matching matroid:** Given a graph, determine whether it has a perfect matching.

## More Examples

**Matric matroid:** Given a matrix  $M$ , compute a subset of vectors of maximum total weight that span the column space of  $M$ .

**Uniform matroid:** Given a set of weighted objects, compute its  $k$  largest elements.

**Cographic matroid:** Given a graph with weighted edges, compute its minimum spanning tree.

**Matching matroid:** Given a graph, determine whether it has a perfect matching.

**Disjoint path matroid:** Given a directed graph with a special vertex  $s$ , find the largest set of edge-disjoint paths from  $s$  to other vertices.

# Matroid v.s. Greedy-MAX

**Theorem:** An independent system  $(S, \mathbf{C})$  is a matroid if and only if for any cost function  $c(\cdot)$ , the Greedy-MAX algorithm gives an optimal solution.

# Matroid v.s. Greedy-MAX

**Theorem:** An independent system  $(S, \mathbf{C})$  is a matroid if and only if for any cost function  $c(\cdot)$ , the Greedy-MAX algorithm gives an optimal solution.

**Proof.** ( $\Rightarrow$ ) When  $(S, \mathbf{C})$  is a matroid,  $u(F) = v(F)$  for any  $F \subseteq S$ . Therefore, Greedy-MAX gives optimal solution.

Next, we show ( $\Leftarrow$ ).

# Sufficiency

( $\Leftarrow$ ) For contradiction, suppose independent system  $(S, \mathbf{C})$  is not a matroid. Then there exists  $F \subseteq S$  such that  $F$  has two maximal independent sets  $I$  and  $J$  with  $|I| < |J|$ . Define

$$c(e) = \begin{cases} 1 + \varepsilon & \text{if } e \in I \\ 1 & \text{if } e \in J \setminus I \\ 0 & \text{if } e \in S \setminus (I \cup J) \end{cases}$$

where  $\varepsilon$  is a sufficiently small positive number to satisfy  $c(I) < c(J)$ . Then the Greedy-MAX algorithm will produce  $I$ , which is not optimal! □



# Outline

- 1 Matroid
  - Independent System
  - Matroid
- 2 Greedy Algorithm on Matroid
  - $u(F)$  and  $v(F)$
  - Greedy-MAX Algorithm
- 3 Task Scheduling Problem
  - Unit-Time Task Scheduling
  - Greedy Approach

# Unit-Time Task

A **unit-time task** is a job, such as a program to be run on a computer, that requires exactly one unit of time to complete.

# Unit-Time Task

A **unit-time task** is a job, such as a program to be run on a computer, that requires exactly one unit of time to complete.

Given a finite set  $S$  of unit-time tasks, a **schedule** for  $S$  is a permutation of  $S$  specifying the order in which to perform these tasks.

# Unit-Time Task

A **unit-time task** is a job, such as a program to be run on a computer, that requires exactly one unit of time to complete.

Given a finite set  $S$  of unit-time tasks, a **schedule** for  $S$  is a permutation of  $S$  specifying the order in which to perform these tasks.

For example, the first task in the schedule begins at time 0 and finishes at time 1, the second task begins at time 1 and finishes at time 2, and so on.

# Unit-time Task Scheduling Problem

The problem of **scheduling unit-time tasks with deadlines and penalties for a single processor** has the following inputs:

- a set  $S = \{1, 2, \dots, n\}$  of  $n$  unit-time tasks;
- a set of  $n$  integer deadlines  $d_1, d_2, \dots, d_n$ , such that each  $d_i$  satisfies  $1 \leq d_i \leq n$  and task  $i$  is supposed to finish by time  $d_i$ ;
- a set of  $n$  nonnegative weights or penalties  $w_1, w_2, \dots, w_n$ , such that a penalty  $w_i$  is incurred if task  $i$  is not finished by time  $d_i$ .

# Unit-time Task Scheduling Problem

The problem of **scheduling unit-time tasks with deadlines and penalties for a single processor** has the following inputs:

- a set  $S = \{1, 2, \dots, n\}$  of  $n$  unit-time tasks;
- a set of  $n$  integer deadlines  $d_1, d_2, \dots, d_n$ , such that each  $d_i$  satisfies  $1 \leq d_i \leq n$  and task  $i$  is supposed to finish by time  $d_i$ ;
- a set of  $n$  nonnegative weights or penalties  $w_1, w_2, \dots, w_n$ , such that a penalty  $w_i$  is incurred if task  $i$  is not finished by time  $d_i$ .

**Requirement:** find a schedule for  $S$  on a machine within time  $n$  that minimizes the total penalty incurred for missed deadline.

# Properties of a Schedule

Given a schedule  $S$ , Define:

**Early:** a task is **early** in  $S$  if it finishes before its deadline.

**Late:** a task is **late** in  $S$  if it finishes after its deadline.

**Early-First Form:**  $S$  is in the **early-first form** if the early tasks precede the late tasks.

# Properties of a Schedule

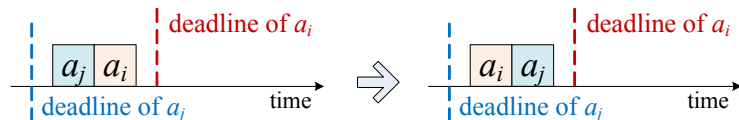
Given a schedule  $S$ , Define:

**Early:** a task is **early** in  $S$  if it finishes before its deadline.

**Late:** a task is **late** in  $S$  if it finishes after its deadline.

**Early-First Form:**  $S$  is in the **early-first form** if the early tasks precede the late tasks.

**Claim:** An arbitrary schedule can always be put into *early-first form* without changing its penalty value.





## Properties of a Schedule (2)

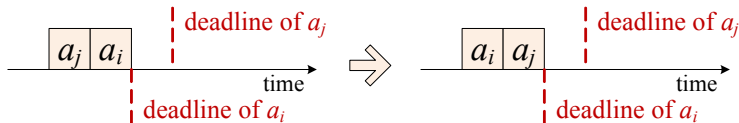
**Canonical Form:** An arbitrary schedule can always be transformed into *canonical form*, in which the early tasks precede the late tasks and are scheduled in order of monotonically increasing deadlines.

## Properties of a Schedule (2)

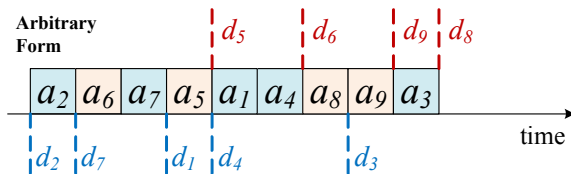
**Canonical Form:** An arbitrary schedule can always be transformed into *canonical form*, in which the early tasks precede the late tasks and are scheduled in order of monotonically increasing deadlines.

First put the schedule into early-first form.

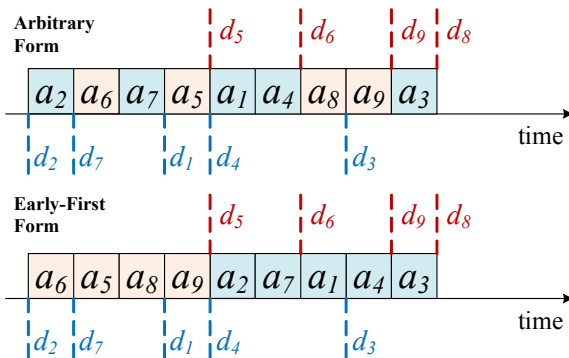
Then swap the position of any consecutive early tasks  $a_i$  and  $a_j$  if  $d_j > d_i$  but  $a_j$  appears before  $a_i$ .



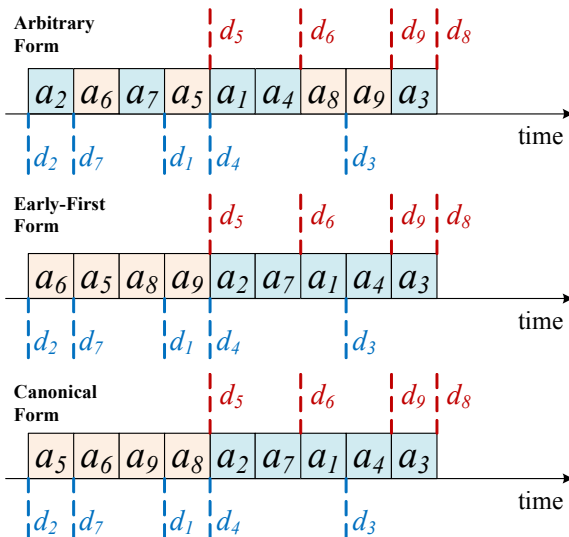
# An Example



# An Example



# An Example



# Outline

- 1 Matroid
  - Independent System
  - Matroid
- 2 Greedy Algorithm on Matroid
  - $u(F)$  and  $v(F)$
  - Greedy-MAX Algorithm
- 3 Task Scheduling Problem
  - Unit-Time Task Scheduling
  - Greedy Approach

# Reduction

The search for an optimal schedule  $S$  thus reduces to finding a set  $A$  of tasks that we assign to be early in the optimal schedule.

# Reduction

The search for an optimal schedule  $S$  thus reduces to finding a set  $A$  of tasks that we assign to be early in the optimal schedule.

To determine  $A$ , we can create the actual schedule by listing the elements of  $A$  in order of monotonically increasing deadlines, then listing the late tasks (i.e.,  $S - A$ ) in any order, producing a canonical ordering of the optimal schedule.



# Independence

**Independent:** A set of tasks  $A$  is independent if there exists a schedule for these tasks without penalty.

Clearly, the set of early tasks for a schedule forms an independent set of tasks. Let  $\mathbf{C}$  denote the set of all independent sets of tasks.

For  $t = 0, 1, 2, \dots, n$ , let

$N_t(A)$  denote the number of tasks in  $A$  whose deadline is  $t$  or earlier.

Note that  $N_0(A) = 0$  for any set  $A$ .

# Lemma

**Lemma:** For any set of tasks  $A$ , the statements (1)-(3) are equivalent.

(1). The set  $A$  is independent.

(2). For  $t = 0, 1, 2, \dots, n$ ,  $N_t(A) \leq t$ .

(3). If the tasks in  $A$  are scheduled in order of monotonically increasing deadlines, then no task is late.

# Lemma

**Lemma:** For any set of tasks  $A$ , the statements (1)-(3) are equivalent.

(1). The set  $A$  is independent.

(2). For  $t = 0, 1, 2, \dots, n$ ,  $N_t(A) \leq t$ .

(3). If the tasks in  $A$  are scheduled in order of monotonically increasing deadlines, then no task is late.

**Proof:**

$\neg(2) \Rightarrow \neg(1)$ : if  $N_t(A) > t$  for some  $t$ , then there is no way to make a schedule with no late tasks for set  $A$ , because more than  $t$  tasks must finish before time  $t$ . Therefore, (1) implies (2).

$(2) \Rightarrow (3)$ : there is no way to “get stuck” when scheduling the tasks in order of monotonically increasing deadlines, since (2) implies that the  $i$ th largest deadline is at least  $i$ .

$(3) \Rightarrow (1)$ : trivial.

□

# Greedy Approach

Use the previous lemma, we can easily compute whether or not a given set of tasks is independent.

# Greedy Approach

Use the previous lemma, we can easily compute whether or not a given set of tasks is independent.

The problem of **minimizing** the sum of the penalties of the late tasks is the same as the problem of **maximizing** the sum of the penalties of the early tasks.

# Greedy Approach

Use the previous lemma, we can easily compute whether or not a given set of tasks is independent.

The problem of **minimizing** the sum of the penalties of the late tasks is the same as the problem of **maximizing** the sum of the penalties of the early tasks.

Thus if  $(S, \mathbf{C})$  is a matroid, then we can use Greedy-MAX to find an independent set  $A$  of tasks with the maximum total penalty, which is proved to be an optimal solution.

# Matroid Theorem

**Theorem:** Let  $S$  be a set of unit-time tasks with deadlines and  $\mathbf{C}$  the set of all independent tasks of  $S$ . Then  $(S, \mathbf{C})$  is a matroid.

# Matroid Theorem

**Theorem:** Let  $S$  be a set of unit-time tasks with deadlines and  $\mathbf{C}$  the set of all independent tasks of  $S$ . Then  $(S, \mathbf{C})$  is a matroid.

**Proof:** (Hereditary): Trivial.

(Exchange Property): Consider two independent sets  $A$  and  $B$  with  $|A| < |B|$ . Let  $k$  be the largest  $t$  such that  $N_t(A) \geq N_t(B)$ . Then  $k < n$  and  $N_t(A) < N_t(B)$  for  $k + 1 \leq t \leq n$ . Choose  $x \in \{i \in B \setminus A \mid d_i = k + 1\}$ .

$$\text{Then, } N_t(A \cup \{x\}) = N_t(A) \leq t, \quad \text{for } 1 \leq t \leq k,$$

$$\text{and } N_t(A \cup \{x\}) = N_t(A) + 1 \leq N_t(B) \leq t, \quad \text{for } k + 1 \leq t \leq n.$$

Thus  $A \cup \{x\} \in \mathbf{C}$ . □



# The Algorithm

Implementing Greedy-MAX, for any given set of tasks  $S$ , we could sort them by penalties and determine the best selections.

# The Algorithm

Implementing Greedy-MAX, for any given set of tasks  $S$ , we could sort them by penalties and determine the best selections.

**Time Complexity:**  $O(n^2)$ .

Sort the tasks takes  $O(n \log n)$ .

Check whether  $A \cup \{x\} \in \mathbf{C}$  takes  $O(n)$ .

There are totally  $O(n)$  iterations of independence check.

Thus the finally complexity is  $O(n \log n + n \cdot n) \rightarrow O(n^2)$ .

# An Example

Given an instance of 7 tasks with deadlines and penalties as follows:

$a_i$	1	2	3	4	5	6	7
$d_i$	4	2	4	3	1	4	6
$w_i$	70	60	50	40	30	20	10

# An Example

Given an instance of 7 tasks with deadlines and penalties as follows:

$a_i$	1	2	3	4	5	6	7
$d_i$	4	2	4	3	1	4	6
$w_i$	70	60	50	40	30	20	10

Greedy-MAX selects  $a_1, a_2, a_3, a_4$ , then rejects  $a_5, a_6$ , and finally accepts  $a_7$ .

The final schedule is  $\langle a_2, a_4, a_1, a_3, a_7, a_5, a_6 \rangle$ .

The optimal penalty is  $w_5 + w_6 = 50$ .