Advanced Core in Algorithm Design #3
算法設計要論 第 3 回

Yasushi Kawase
河瀬 康志

Oct. 18th, 2022

last update: 4:30pm, October 16, 2022

Schedule

Lec. #	Date	Topics
1	10/4	Introduction, Stable matching
2	10/11	Basics of Algorithm Analysis, Greedy Algorithms (1/2)
3	10/18	Greedy Algorithms (2/2)
4	10/25	Divide and Conquer (1/2)
5	11/1	Divide and Conquer (2/2)
6	11/8	Dynamic Programming (1/2)
7	11/15	Dynamic Programming (2/2)
—	11/22	Thursday Classes
8	11/29	Network Flow (1/2)
9	12/6	Network Flow (2/2)
10	12/13	NP and Computational Intractability
11	12/20	Approximation Algorithms (1/2)
12	12/27	Approximation Algorithms (2/2)
13	1/10	Randomized Algorithms

Outline

1. Minimum Spanning Tree Problem
2. Job Scheduling Problem
3. Matroids

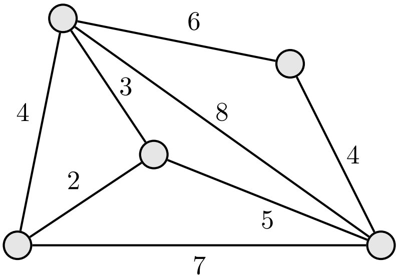
Minimum spanning problem

Problem

- Input: Connected undirected graph $G = (V, E)$, weight $w_e \geq 0$ $(e \in E)$
- Goal: Compute a minimum cost spanning tree (MST)

subgraph that is both connected and acyclic

Example



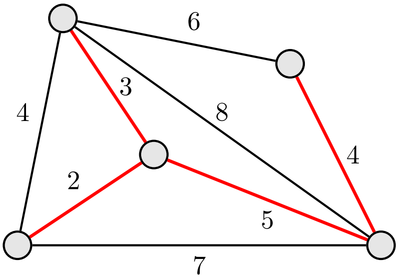
Minimum spanning problem

Problem

- Input: Connected undirected graph $G = (V, E)$, weight $w_e \geq 0$ $(e \in E)$
- Goal: Compute a minimum cost spanning tree (MST)

subgraph that is both connected and acyclic

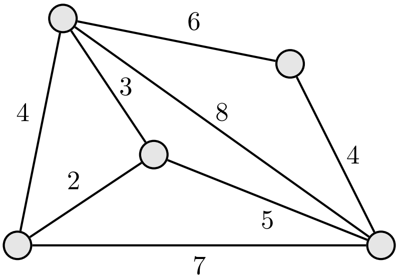
Example   minimum cost $= 14$



Kruskal's algorithm

Algorithm

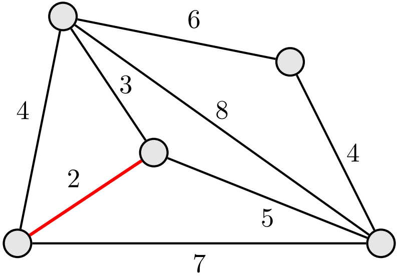
$F \leftarrow \emptyset$;
Sort the edges E by weight;
foreach $e \in E$ *in increasing order of weight* **do**
  if $F \cup \{e\}$ *has no cycle* **then** $F \leftarrow F \cup \{e\}$;
Return (V, F);



Kruskal's algorithm

Algorithm

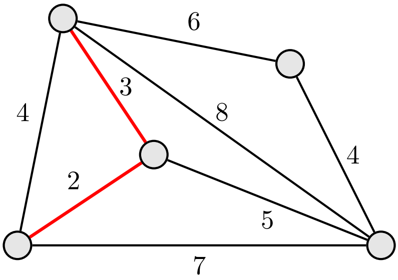
$F \leftarrow \emptyset$;
Sort the edges E by weight;
foreach $e \in E$ *in increasing order of weight* **do**
  if $F \cup \{e\}$ *has no cycle* **then** $F \leftarrow F \cup \{e\}$;
Return (V, F);



Kruskal's algorithm

Algorithm

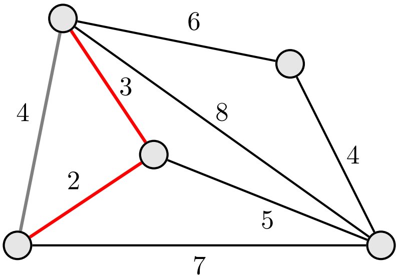
$F \leftarrow \emptyset$;
Sort the edges E by weight;
foreach $e \in E$ *in increasing order of weight* **do**
  if $F \cup \{e\}$ *has no cycle* **then** $F \leftarrow F \cup \{e\}$;
Return (V, F);



Kruskal's algorithm

Algorithm

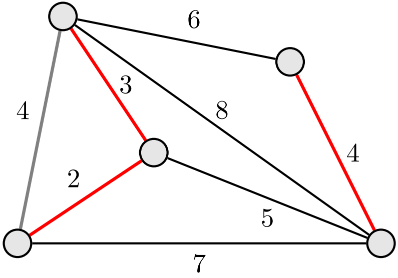
$F \leftarrow \emptyset$;
Sort the edges E by weight;
foreach $e \in E$ *in increasing order of weight* **do**
  if $F \cup \{e\}$ *has no cycle* **then** $F \leftarrow F \cup \{e\}$;
Return (V, F);



Kruskal's algorithm

Algorithm

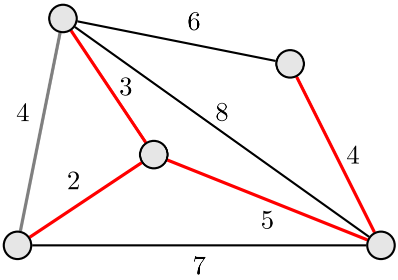
$F \leftarrow \emptyset$;
Sort the edges E by weight;
foreach $e \in E$ *in increasing order of weight* **do**
  if $F \cup \{e\}$ *has no cycle* **then** $F \leftarrow F \cup \{e\}$;
Return (V, F);



Kruskal's algorithm

Algorithm

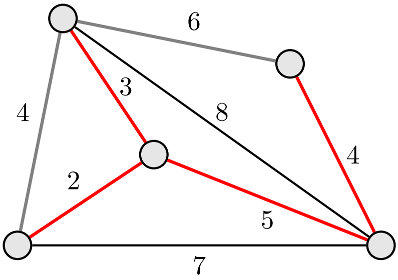
$F \leftarrow \emptyset$;
Sort the edges E by weight;
foreach $e \in E$ *in increasing order of weight* **do**
  if $F \cup \{e\}$ *has no cycle* **then** $F \leftarrow F \cup \{e\}$;
Return (V, F);



Kruskal's algorithm

Algorithm

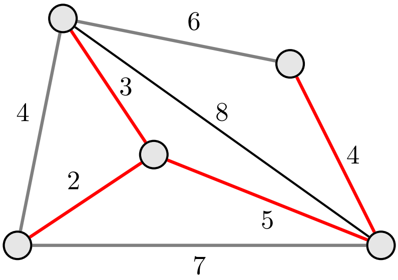
$F \leftarrow \emptyset$;
Sort the edges E by weight;
foreach $e \in E$ *in increasing order of weight* **do**
  if $F \cup \{e\}$ *has no cycle* **then** $F \leftarrow F \cup \{e\}$;
Return (V, F);



Kruskal's algorithm

Algorithm

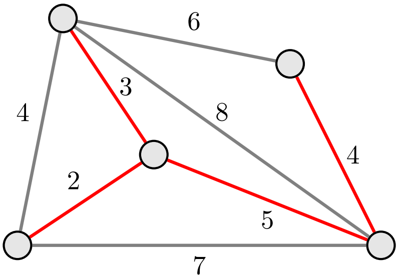
$F \leftarrow \emptyset$;
Sort the edges E by weight;
foreach $e \in E$ *in increasing order of weight* **do**
    if $F \cup \{e\}$ *has no cycle* **then** $F \leftarrow F \cup \{e\}$;
Return (V, F);



Kruskal's algorithm

Algorithm

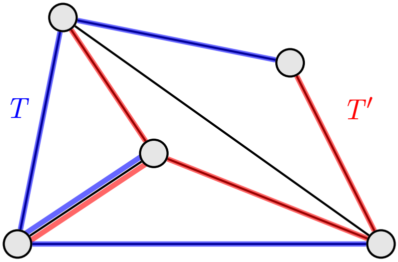
$F \leftarrow \emptyset$;
Sort the edges E by weight;
foreach $e \in E$ *in increasing order of weight* **do**
  if $F \cup \{e\}$ *has no cycle* **then** $F \leftarrow F \cup \{e\}$;
Return (V, F);



Structure of Spanning Trees

Lemma for spanning trees T, T'

$\forall e \in T \setminus T'$, $\exists f \in T' \setminus T$, $T' \cup \{e\} \setminus \{f\}$ is a spanning tree



- There is a cycle C in $(V, T' \cup \{e\})$
- Since T is a tree, $C \not\subseteq T$, and hence $\exists f \in C \setminus T \subseteq T' \setminus T$
- $T' \cup \{e\} \setminus \{f\}$ is a spanning tree

Correctness

Theorem

Kruskal's algorithm outputs a minimum spanning tree

Proof by contradiction

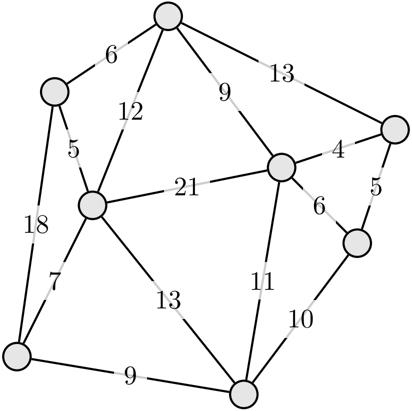
- T: output of Kruskal's algorithm
- T^*: MST with maximum $|T \cap T^*|$ ($T^* \neq T$ by assumption)
- $e \in T \setminus T^*$: the edge not in T^* that the algorithm firstly choose
- $\exists f \in T^* \setminus T$ such that T^{**} is a spanning tree (by lemma)

  T^{**}: $T^* \cup \{e\} \setminus \{f\}$

- the algorithm is greedy → $c_e \leq c_f$
- $c(T^{**}) = c(T^*) + c_e - c_f \leq c(T^*)$ → T^{**} is MST
- $|T \cap T^{**}| = |T \cap T^*| + 1$ → contradicts to the definition of T^*

Quiz

Compute a minimum spanning tree



Outline

Scheduling to Minimize Lateness

Problem

- Input: n unit-time jobs $J = \{1, 2, \ldots, n\}$
  job j has deadline $d_j \in \mathbb{Z}_+$ and penalty p_j
- Goal: minimum penalty schedule (permutation) for J

incurred for missed deadlines

Example

j	1	2	3	4	5	6	7
d_j	1	2	2	3	4	4	6
p_j	3	5	6	7	2	5	1

Scheduling to Minimize Lateness

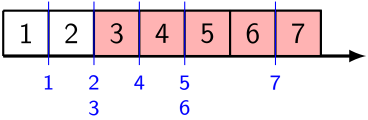
Problem

- Input: n unit-time jobs $J = \{1, 2, \ldots, n\}$
  job j has deadline $d_j \in \mathbb{Z}_+$ and penalty p_j
- Goal: minimum penalty schedule (permutation) for J

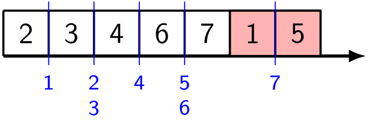
incurred for missed deadlines

Example

j	1	2	3	4	5	6	7
d_j	1	2	2	3	4	4	6
p_j	3	5	6	7	2	5	1



penalty $= 6 + 7 + 2 + 5 + 1 = 21$



penalty $= 3 + 2 = 5$

Canonical form

Definition

For a given schedule, a job is

- early if it finishes before its deadline
- late if it finishes after its deadline

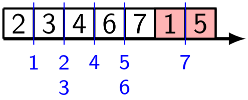
Definition

A schedule is called canonical if

- the early jobs precede the late jobs
- the early jobs are scheduled in increasing order of deadlines

Observation

every schedule can be put into canonical form



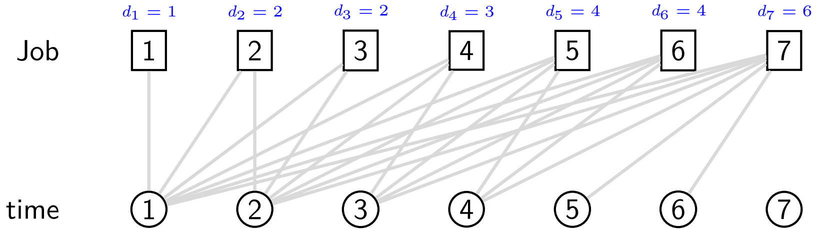
Processable

Definition

A set of jobs $S \subseteq J$ is processable if S can be scheduled as early jobs

Observation

A set of jobs S is processable iff $|\{j \in S : d_j \leq t\}| \leq t$ $(\forall t = 0, 1, \ldots, |S|)$



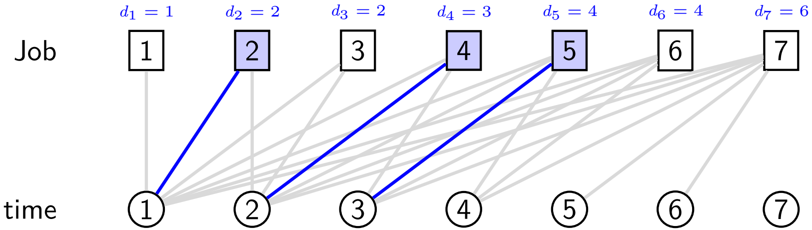
Processable

Definition

A set of jobs $S \subseteq J$ is processable if S can be scheduled as early jobs

Observation

A set of jobs S is processable iff $|\{j \in S : d_j \le t\}| \le t$ $(\forall t = 0, 1, \dots, |S|)$



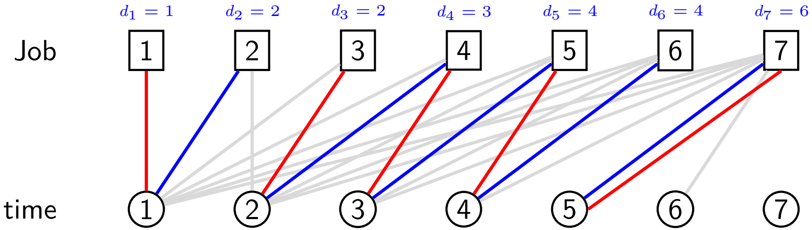
Structure of Processable Sets

Observation

Every maximal processable set has the same size

Lemma for maximal processable sets S, T

$\forall s \in S \setminus T$, $\exists t \in T \setminus S$, $T \cup \{s\} \setminus \{t\}$ is processable



Algorithm

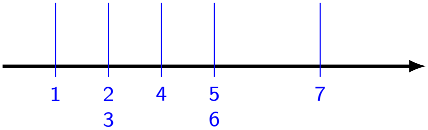
Greedy Algorithm

$F \leftarrow \emptyset$;
Sort the jobs J by penalty;
foreach $j \in J$ *in decreasing order of penalty* **do**
  if $F \cup \{j\}$ *is processable* **then** $F \leftarrow F \cup \{j\}$;
Return a canonical schedule in which every $j \in F$ is early;

Example

j	1	2	3	4	5	6	7
d_j	1	2	2	3	4	4	6
p_j	3	5	6	7	2	5	1

decreasing order: 4, 3, 2, 6, 1, 5, 7



Algorithm

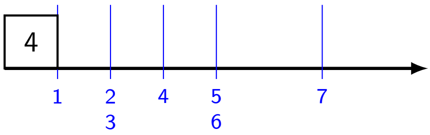
Greedy Algorithm

$F \leftarrow \emptyset$;
Sort the jobs J by penalty;
foreach $j \in J$ *in decreasing order of penalty* **do**
  if $F \cup \{j\}$ *is processable* **then** $F \leftarrow F \cup \{j\}$;
Return a canonical schedule in which every $j \in F$ is early;

Example

j	1	2	3	4	5	6	7
d_j	1	2	2	3	4	4	6
p_j	3	5	6	7	2	5	1

decreasing order: 4, 3, 2, 6, 1, 5, 7



Algorithm

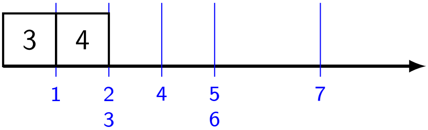
Greedy Algorithm

$F \leftarrow \emptyset$;
Sort the jobs J by penalty;
foreach $j \in J$ *in decreasing order of penalty* **do**
  if $F \cup \{j\}$ *is processable* **then** $F \leftarrow F \cup \{j\}$;
Return a canonical schedule in which every $j \in F$ is early;

Example

j	1	2	3	4	5	6	7
d_j	1	2	2	3	4	4	6
p_j	3	5	6	7	2	5	1

decreasing order: 4, 3, 2, 6, 1, 5, 7



Algorithm

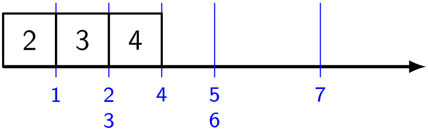
Greedy Algorithm

$F \leftarrow \emptyset$;
Sort the jobs J by penalty;
foreach $j \in J$ *in decreasing order of penalty* **do**
  if $F \cup \{j\}$ *is processable* **then** $F \leftarrow F \cup \{j\}$;
Return a canonical schedule in which every $j \in F$ is early;

Example

j	1	2	3	4	5	6	7
d_j	1	2	2	3	4	4	6
p_j	3	5	6	7	2	5	1

decreasing order: 4, 3, 2, 6, 1, 5, 7



Algorithm

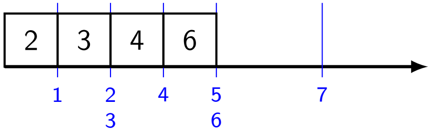
Greedy Algorithm

$F \leftarrow \emptyset$;
Sort the jobs J by penalty;
foreach $j \in J$ *in decreasing order of penalty* **do**
  if $F \cup \{j\}$ *is processable* **then** $F \leftarrow F \cup \{j\}$;
Return a canonical schedule in which every $j \in F$ is early;

Example

j	1	2	3	4	5	6	7
d_j	1	2	2	3	4	4	6
p_j	3	5	6	7	2	5	1

decreasing order: 4, 3, 2, 6, 1, 5, 7



Algorithm

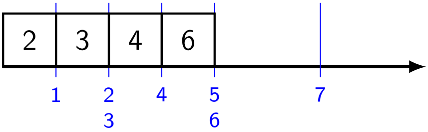
Greedy Algorithm

$F \leftarrow \emptyset$;
Sort the jobs J by penalty;
foreach $j \in J$ *in decreasing order of penalty* **do**
  if $F \cup \{j\}$ *is processable* **then** $F \leftarrow F \cup \{j\}$;
Return a canonical schedule in which every $j \in F$ is early;

Example

j	1	2	3	4	5	6	7
d_j	1	2	2	3	4	4	6
p_j	3	5	6	7	2	5	1

decreasing order: 4, 3, 2, 6, 1, 5, 7



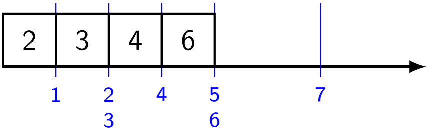
Algorithm

Greedy Algorithm

$F \leftarrow \emptyset$;
Sort the jobs J by penalty;
foreach $j \in J$ *in decreasing order of penalty* **do**
  if $F \cup \{j\}$ *is processable* **then** $F \leftarrow F \cup \{j\}$;
Return a canonical schedule in which every $j \in F$ is early;

Example

j	1	2	3	4	5	6	7
d_j	1	2	2	3	4	4	6
p_j	3	5	6	7	2	5	1



decreasing order: 4, 3, 2, 6, 1, 5, 7

Algorithm

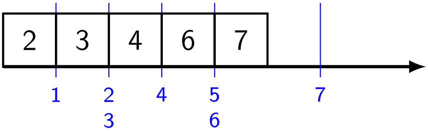
Greedy Algorithm

$F \leftarrow \emptyset$;
Sort the jobs J by penalty;
foreach $j \in J$ *in decreasing order of penalty* **do**
  if $F \cup \{j\}$ *is processable* **then** $F \leftarrow F \cup \{j\}$;
Return a canonical schedule in which every $j \in F$ is early;

Example

j	1	2	3	4	5	6	7
d_j	1	2	2	3	4	4	6
p_j	3	5	6	7	2	5	1

decreasing order: 4, 3, 2, 6, 1, 5, 7



Algorithm

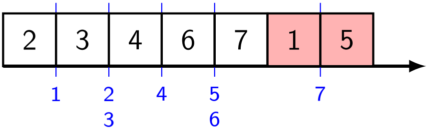
Greedy Algorithm

$F \leftarrow \emptyset$;
Sort the jobs J by penalty;
foreach $j \in J$ *in decreasing order of penalty* **do**
  if $F \cup \{j\}$ *is processable* **then** $F \leftarrow F \cup \{j\}$;
Return a canonical schedule in which every $j \in F$ is early;

Example

j	1	2	3	4	5	6	7
d_j	1	2	2	3	4	4	6
p_j	3	5	6	7	2	5	1

decreasing order: 4, 3, 2, 6, 1, 5, 7



Correctness

Theorem

The greedy algorithm outputs an optimal schedule

Proof by contradiction

- F: the early jobs of the greedy algorithm
  → penalty of the schedule is $\sum_{j \in J \setminus F} p_j = p(J) - p(F)$
- F^*: the early jobs of the optimal schedule with maximum $|F \cap F^*|$
  → penalty of the schedule is $\sum_{j \in J \setminus F^*} p_j = p(J) - p(F^*)$
- $s \in F \setminus F^*$: the job not in F^* that the algorithm firstly choose
- $\exists t \in F^* \setminus F$ such that F^{**} is processable
  (F^{**}: $F^* \cup \{s\} \setminus \{t\}$ is processable (by lemma))
- the algorithm is greedy → $p_s \geq p_t$
- $p(F^{**}) = p(F^*) + p_s - p_t \geq p(F^*)$ → F^{**} implies an opt. schedule
- $|F \cap F^{**}| = |F \cap F^*| + 1$ → contradicts to the definition of F^*

Quiz

What is the minimum penalty of a schedule?

j	1	2	3	4	5	6	7	8
d_j	1	2	2	3	3	4	4	5
p_j	3	5	6	7	2	5	4	1

Outline

Matroids

Definition

For a finite set E and a subset family $\mathcal{I} \subseteq 2^E$, $(E, \mathcal{I})$ is a matroid if

- $\emptyset \in \mathcal{I}$
- $X \subseteq Y \in \mathcal{I} \Rightarrow X \in \mathcal{I}$
- $X, Y \in \mathcal{I}$, $|X| > |Y| \Rightarrow \exists x \in X \setminus Y,\ Y \cup \{x\} \in \mathcal{I}$

$X \in \mathcal{I}$ is called independent set

Simple Examples

- $E = \{1, 2\}$, $\mathcal{I} = \big\{\emptyset, \{1\}, \{2\}\big\}$ (matroid)
- $E = \{1, 2, 3\}$, $\mathcal{I} = \big\{\emptyset, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}\big\}$ (matroid)
- $E = \{1, 2, 3\}$, $\mathcal{I} = \big\{\emptyset, \{1\}, \{2\}, \{1, 2\}, \{1, 2, 3\}\big\}$ (not matroid)
- $E = \{1, 2, 3, 4\}$, $\mathcal{I} = \big\{\emptyset, \{1\}, \{2\}, \{3\}, \{4\}, \{1, 2\}, \{3, 4\}\big\}$ (not matroid)

Uniform matroid, Partition matroid

Proposition (Uniform matroid)

For any natural number $r \geq 0$, $(E,\ \{X \subseteq E \mid |X| \leq r\})$ is a matroid

Example

- $E = \{1, 2, 3, 4\},\ r = 2$
- $\mathcal{I} = \{\emptyset, \{1\}, \{2\}, \{3\}, \{4\}, \{1, 2\}, \{1, 3\}, \{1, 4\}, \{2, 3\}, \{2, 4\}, \{3, 4\}\}$

Proposition (Partition matroid)

For any partition $(S_1, \ldots, S_k)$ of E and $q_1, \ldots, q_k \in \mathbb{Z}_{++}$,
$(E,\ \{X \subseteq E \mid |X \cap S_i| \leq q_i\ (\forall i = 1, \ldots, k)\})$ is a matroid

Example

- $E = \{1, 2, 3, 4, 5, 6\},\ S_1 = \{1, 2, 3\},\ q_1 = 1,\ S_2 = \{4, 5, 6\},\ q_2 = 2$
- $\mathcal{I} = \{\emptyset, \{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{1, 4\}, \{1, 5\}, \{1, 6\}, \ldots, \{3, 5, 6\}\}$

Linear matroid

Proposition

$\mathbb{F}$ is a field

For $a_1, a_2, \ldots, a_n \in \mathbb{F}^m$ and $E = \{a_1, a_2, \ldots, a_n\}$,
$(E,\ \{X \subseteq E \mid X \text{ is linearly independent}\})$ is a matroid

Example

- $a_1 = \begin{pmatrix}1\\0\end{pmatrix}$, $a_2 = \begin{pmatrix}0\\1\end{pmatrix}$, $a_3 = \begin{pmatrix}1\\2\end{pmatrix}$, $a_4 = \begin{pmatrix}-1\\0\end{pmatrix}$, $\mathbb{F} = \mathbb{R}$
- $E = \{a_1, a_2, a_3, a_4\}$
- $\mathcal{I} = \{\emptyset, \{a_1\}, \{a_2\}, \{a_3\}, \{a_4\}, \{a_1, a_2\}, \{a_1, a_3\}, \{a_2, a_3\}, \{a_2, a_4\}, \{a_3, a_4\}\}$

Graphic matroid (cycle matroid)

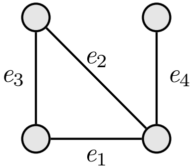
Proposition

For an undirected graph $G = (V, E)$,
$(E,\ \{X \subseteq E \mid X \text{ does not contain a cycle}\})$ is a matroid

a graphic matroid is a linear matroid ($\mathbb{F} = \mathbb{Z}_2$)

Example

- $E = \{e_1, e_2, e_3, e_4\}$
- $\mathcal{I} = \left\{ \begin{matrix} \emptyset, \{e_1\}, \{e_2\}, \{e_3\}, \{e_4\}, \{e_1, e_2\}, \{e_1, e_3\}, \{e_1, e_4\}, \{e_2, e_3\} \\ \{e_2, e_4\}, \{e_3, e_4\}, \{e_1, e_2, e_4\}, \{e_1, e_3, e_4\}, \{e_2, e_3, e_4\} \end{matrix} \right\}$



Transversal matroid

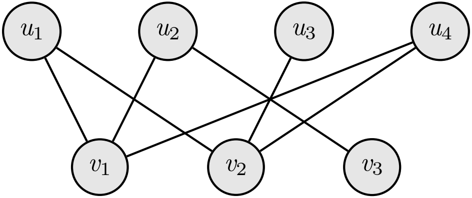
Proposition

For a bipartite graph $G = (U, V; E)$,
$(U, \{X \subseteq U \mid \text{there exists a matching that covers } X\})$ is a matroid

a transversal matroid is a linear matroid (e.g. $\mathbb{F} = \mathbb{R}$)

Example

- $U = \{u_1, u_2, u_3, u_4\}$
- $\mathcal{I} = \left\{\begin{array}{c} \emptyset, \{u_1\}, \{u_2\}, \{u_3\}, \{u_4\}, \{u_1, u_2\}, \{u_1, u_3\}, \{u_1, u_4\}, \{u_2, u_3\} \\ \{u_2, u_4\}, \{u_3, u_4\}, \{u_1, u_2, u_3\}, \{u_1, u_2, u_4\}, \{u_2, u_3, u_4\} \end{array}\right\}$



Base

Definition

For a matroid $(E, \mathcal{I})$, $B \in \mathcal{I}$ is called base if $\forall e \in E \setminus B$, $B \cup \{e\} \notin \mathcal{I}$

Proposition

All the bases of a matroid have the same size.

Example

- $E = \{e_1, e_2, e_3, e_4\}$
- $\mathcal{I} = \left\{ \begin{matrix} \emptyset, \{e_1\}, \{e_2\}, \{e_3\}, \{e_4\}, \{e_1, e_2\}, \{e_1, e_3\}, \{e_1, a_4\}, \{e_2, e_3\} \\ \{e_2, e_4\}, \{e_3, e_4\}, \{e_1, e_2, e_4\}, \{e_1, e_3, e_4\}, \{e_2, e_3, e_4\} \end{matrix} \right\}$
- $\mathcal{B} = \{\{e_1, e_2, e_4\}, \{e_1, e_3, e_4\}, \{e_2, e_3, e_4\}\}$

the set of bases

Basis axiom

Definition (basis axioms)

For a finite set E and a subset family $\mathcal{B} \subseteq 2^E$,

- $\mathcal{B} \neq \emptyset$
- $B, B' \in \mathcal{B}$ and $x \in B \setminus B' \Rightarrow \exists y \in B' \setminus B$ such that $B \setminus \{x\} \cup \{y\} \in \mathcal{B}$

Theorem

- $(E, \mathcal{I})$ is a matroid $\Rightarrow$ the set of bases satisfies the basis axioms
- $(E, \mathcal{B})$ satisfies the basis axioms $\Rightarrow$ $(E, \bigcup_{B \in \mathcal{B}} 2^B)$ is a matroid

Minimum cost base problem

Problem

- Input: matroid $(E, \mathcal{I})$, cost $c\colon E \to \mathbb{R}$
- Goal: minimize $\sum_{e \in X} c(e)$ subject to X is a base of $(E, \mathcal{I})$

Greedy algorithm

$I \leftarrow \emptyset$ and sort the elements E by cost;
foreach $e \in E$ *in increasing order of cost* **do**
    if $I \cup \{e\} \in \mathcal{I}$ **then** $I \leftarrow I \cup \{e\}$;
Return I;

Theorem

The greedy algorithm outputs a minimum cost base

The proof is the same as the MST case (graphic matroid)

Maximum weight independent set problem

Problem

- Input: matroid $(E, \mathcal{I})$, weight $w\colon E \to \mathbb{R}_+$
- Goal: maximize $\sum_{e \in X} w(e)$ subject to $X \in \mathcal{I}$

Greedy algorithm

$I \leftarrow \emptyset$ and sort the elements E by weight;
foreach $e \in E$ *in decreasing order of cost* **do**
    if $I \cup \{e\} \in \mathcal{I}$ **then** $I \leftarrow I \cup \{e\}$;
Return I;

Theorem

The greedy algorithm outputs a maximum weight independent set

$\because$ the algorithm outputs a base X that minimizes $\sum_{e \in X} -w(e)$

Matroids and Greedy algorithm (1/2)

Problem

$\emptyset \in \mathcal{I}$ and $Y \subseteq X \in \mathcal{I} \Rightarrow Y \in \mathcal{I}$

- Input: independence system $(E, \mathcal{I})$, weight $w\colon E \to \mathbb{R}_+$
- Goal: maximize $\sum_{e \in X} w(e)$ subject to $X \in \mathcal{I}$

Greedy algorithm

$I \leftarrow \emptyset$ and sort the edges E by weight;
foreach $e \in E$ *in decreasing order of cost* **do**
  if $I \cup \{e\} \in \mathcal{I}$ **then** $I \leftarrow I \cup \{e\}$;
Return I;

Theorem

For independence system $(E, \mathcal{I})$, the following two are equivalent

(i) for any $w\colon E \to \mathbb{R}_+$, the greedy algorithm outputs an optimal solution

(ii) $(E, \mathcal{I})$ is a matroid

Matroids and Greedy algorithm (2/2)

Theorem

For independence system $(E, \mathcal{I})$, the following two are equivalent

(i) for any $w\colon E \to \mathbb{R}_+$, the greedy algorithm outputs an optimal solution

(ii) $(E, \mathcal{I})$ is a matroid

Proof

- We only prove $\overline{(ii)} \Rightarrow \overline{(i)}$ since $(ii) \Rightarrow (i)$ is already shown
- Suppose that $(E, \mathcal{I})$ is not a matroid. Then, we have $\exists X, Y \in \mathcal{I}$ s.t. $|X| > |Y|$ and $\forall e \in X \setminus Y,\ Y \cup \{e\} \notin \mathcal{I}$
- The greedy algorithm does not output an optimal solution when

$$w(e) = \begin{cases} 1+\epsilon & \text{if } e \in Y \\ 1 & \text{if } e \in X \setminus Y \\ 0 & \text{otherwise} \end{cases}$$