

Tópicos de Análise de Algoritmos

Método guloso

Sec 4.2 do KT e Sec 16.4 do CLRS

Mais um problema de escalonamento

d_i : deadline da tarefa i

t_i : tempo de processamento da tarefa i

Escalonamento: permutação π , onde $\pi(i)$ denota a posição da tarefa i na ordem de execução.

Mais um problema de escalonamento

d_i : deadline da tarefa i

t_i : tempo de processamento da tarefa i

Escalonamento: permutação π , onde $\pi(i)$ denota a posição da tarefa i na ordem de execução.

Dado π , o instante de início da tarefa i é

$$s_i = \sum_{j=1}^{\pi(i)-1} t_{\pi^{-1}(j)},$$

Mais um problema de escalonamento

d_i : deadline da tarefa i

t_i : tempo de processamento da tarefa i

Escalonamento: permutação π , onde $\pi(i)$ denota a posição da tarefa i na ordem de execução.

Dado π , o instante de início da tarefa i é

$$s_i = \sum_{j=1}^{\pi(i)-1} t_{\pi^{-1}(j)},$$

ou seja, é a soma das durações das tarefas anteriores a i .

Mais um problema de escalonamento

d_i : deadline da tarefa i

t_i : tempo de processamento da tarefa i

Escalonamento: permutação π , onde $\pi(i)$ denota a posição da tarefa i na ordem de execução.

Dado π , o instante de início da tarefa i é

$$s_i = \sum_{j=1}^{\pi(i)-1} t_{\pi^{-1}(j)},$$

ou seja, é a soma das durações das tarefas anteriores a i .

Instante de término: $f_i = s_i + t_i$.

Mais um problema de escalonamento

d_i : deadline da tarefa i

t_i : tempo de processamento da tarefa i

Escalonamento: permutação π , onde $\pi(i)$ denota a posição da tarefa i na ordem de execução.

Dado π , o instante de início da tarefa i é

$$s_i = \sum_{j=1}^{\pi(i)-1} t_{\pi^{-1}(j)},$$

ou seja, é a soma das durações das tarefas anteriores a i .

Instante de término: $f_i = s_i + t_i$.

Atraso ℓ_i : 0 se $f_i \leq d_i$ e $f_i - d_i$ se $f_i > d_i$.

Mais um problema de escalonamento

d_i : deadline da tarefa i

t_i : tempo de processamento da tarefa i

Dado escalonamento π ,

s_i : início do processamento da tarefa i

f_i : fim do processamento da tarefa i

ℓ_i : atraso da tarefa i

Mais um problema de escalonamento

d_i : deadline da tarefa i

t_i : tempo de processamento da tarefa i

Dado escalonamento π ,

s_i : início do processamento da tarefa i

f_i : fim do processamento da tarefa i

ℓ_i : atraso da tarefa i

Problema: Dados d e t , encontrar π cujo atraso máximo é mínimo.

Mais um problema de escalonamento

d_i : deadline da tarefa i

t_i : tempo de processamento da tarefa i

Dado escalonamento π ,

s_i : início do processamento da tarefa i

f_i : fim do processamento da tarefa i

ℓ_i : atraso da tarefa i

Problema: Dados d e t , encontrar π cujo atraso máximo é mínimo.

Exemplo 1: $t_1 = 1$ e $d_1 = 2$, $t_2 = 2$ e $d_2 = 4$, $t_3 = 3$ e $d_3 = 6$.

Mais um problema de escalonamento

d_i : deadline da tarefa i

t_i : tempo de processamento da tarefa i

Dado escalonamento π ,

s_i : início do processamento da tarefa i

f_i : fim do processamento da tarefa i

ℓ_i : atraso da tarefa i

Problema: Dados d e t , encontrar π cujo atraso máximo é mínimo.

Exemplo 2: $t_1 = 1$ e $d_1 = 4$, $t_2 = 2$ e $d_2 = 5$, $t_3 = 3$ e $d_3 = 3$.

Possíveis critérios gulosos

- ▶ SPT - shortest processing time (menor t_i primeiro)

Funciona?

Possíveis critérios gulosos

- ▶ SPT - shortest processing time (menor t_i primeiro)

Funciona? Não...

Exemplo: $d_1 = 9$ e $t_1 = 1$, $d_2 = 8$ e $t_2 = 8$.

Possíveis critérios gulosos

- ▶ SPT - shortest processing time (menor t_i primeiro)

Funciona? Não...

Exemplo: $d_1 = 9$ e $t_1 = 1$, $d_2 = 8$ e $t_2 = 8$.



Possíveis critérios gulosos

- ▶ SPT - shortest processing time (menor t_i primeiro)

Funciona? Não...

Exemplo: $d_1 = 9$ e $t_1 = 1$, $d_2 = 8$ e $t_2 = 8$.



- ▶ LST - least slack time (menor $d_i - t_i$ primeiro)

Funciona?

Possíveis critérios gulosos

- ▶ SPT - shortest processing time (menor t_i primeiro)

Funciona? Não...

Exemplo: $d_1 = 9$ e $t_1 = 1$, $d_2 = 8$ e $t_2 = 8$.



- ▶ LST - least slack time (menor $d_i - t_i$ primeiro)

Funciona? Também não...

Exemplo: $d_1 = 2$ e $t_1 = 1$, $d_2 = 8$ e $t_2 = 8$.

Possíveis critérios gulosos

- ▶ SPT - shortest processing time (menor t_i primeiro)

Funciona? Não...

Exemplo: $d_1 = 9$ e $t_1 = 1$, $d_2 = 8$ e $t_2 = 8$.



- ▶ LST - least slack time (menor $d_i - t_i$ primeiro)

Funciona? Também não...

Exemplo: $d_1 = 2$ e $t_1 = 1$, $d_2 = 8$ e $t_2 = 8$.



Possíveis critérios gulosos

- ▶ LST - least slack time (menor $d_i - t_i$ primeiro)

Funciona? Também não...

Exemplo: $d_1 = 2$ e $t_1 = 1$, $d_2 = 8$ e $t_2 = 8$.



- ▶ EDD - earliest due date (menor d_i primeiro)

Funciona?

Algoritmo resultante

GULOSO (d, t, n)

- 1 seja π permutação tq $d[\pi^{-1}(1)] \leq \dots \leq d[\pi^{-1}(n)]$
- 2 **devolva** π

Nem olhamos o t ... Será que isso funciona???

Algoritmo resultante

GULOSO (d, t, n)

- 1 seja π permutação tq $d[\pi^{-1}(1)] \leq \dots \leq d[\pi^{-1}(n)]$
- 2 **devolva** π

Note que não há tempo ocioso em uma solução.

Algoritmo resultante

GULOSO (d, t, n)

- 1 seja π permutação tq $d[\pi^{-1}(1)] \leq \dots \leq d[\pi^{-1}(n)]$
- 2 **devolva** π

Note que não há tempo ocioso em uma solução.

Dado um escalonamento,

uma **inversão** é um par de índices i e j tais que $\pi(i) < \pi(j)$ e $d_i > d_j$.

Algoritmo resultante

GULOSO (d, t, n)

- 1 seja π permutação tq $d[\pi^{-1}(1)] \leq \dots \leq d[\pi^{-1}(n)]$
- 2 **devolva** π

Note que não há tempo ocioso em uma solução.

Dado um escalonamento,

uma **inversão** é um par de índices i e j tais que $\pi(i) < \pi(j)$ e $d_i > d_j$.

Afirmção 1:

Dois escalonamentos sem inversão têm o mesmo atraso máximo.

Algoritmo resultante

GULOSO (d, t, n)

- 1 seja π permutação tq $d[\pi^{-1}(1)] \leq \dots \leq d[\pi^{-1}(n)]$
- 2 **devolva** π

Note que não há tempo ocioso em uma solução.

Dado um escalonamento,

uma **inversão** é um par de índices i e j tais que $\pi(i) < \pi(j)$ e $d_i > d_j$.

Afirmção 1:

Dois escalonamentos sem inversão têm o mesmo atraso máximo.

Prova: Qualquer que seja a ordem das tarefas com deadline d , elas, juntas, consomem o mesmo tempo para serem executadas.

Em cada bloco de tarefas com um mesmo deadline d , a mais atrasada é a que foi escalonada por último.

O atraso máximo nesse bloco é sempre o atraso da última.

Algoritmo resultante

GULOSO (d, t, n)

- 1 seja π permutação tq $d[\pi^{-1}(1)] \leq \dots \leq d[\pi^{-1}(n)]$
- 2 **devolva** π

Note que não há tempo ocioso em uma solução.

Dado um escalonamento,

uma **inversão** é um par de índices i e j tais que $\pi(i) < \pi(j)$ e $d_i > d_j$.

Afirmção 1:

Dois escalonamentos sem inversão têm o mesmo atraso máximo.

Afirmção 2:

Existe uma solução que não tem nenhuma inversão.

Prova: Se houver uma inversão, há uma entre tarefas consecutivas.

Inverta duas tarefas consecutivas que formam uma inversão.

Isso não aumenta o atraso máximo. Repita até não haver inversão.

Algoritmo resultante

GULOSO (d, t, n)

- 1 seja π permutação tq $d[\pi^{-1}(1)] \leq \dots \leq d[\pi^{-1}(n)]$
- 2 **devolva** π

Note que não há tempo ocioso em uma solução.

Dado um escalonamento,

uma **inversão** é um par de índices i e j tais que $\pi(i) < \pi(j)$ e $d_i > d_j$.

Afirmiação 1:

Dois escalonamentos sem inversão têm o mesmo atraso máximo.

Afirmiação 2:

Existe uma solução que não tem nenhuma inversão.

Disso segue que o algoritmo funciona.

Matroides e o método guloso

U : conjunto finito arbitrário.

$\mathcal{C}$: família não vazia de subconjuntos de U hereditária.

(subconjuntos de conjuntos em $\mathcal{C}$ estão em $\mathcal{C}$)

Matroides e o método guloso

U : conjunto finito arbitrário.

$\mathcal{C}$: família não vazia de subconjuntos de U hereditária.

(subconjuntos de conjuntos em $\mathcal{C}$ estão em $\mathcal{C}$)

Problema 1: Dados U e $\mathcal{C}$,
encontrar um conjunto de $\mathcal{C}$ de tamanho máximo.

Matroides e o método guloso

U : conjunto finito arbitrário.

$\mathcal{C}$: família não vazia de subconjuntos de U hereditária.

(subconjuntos de conjuntos em $\mathcal{C}$ estão em $\mathcal{C}$)

Problema 1: Dados U e $\mathcal{C}$,
encontrar um conjunto de $\mathcal{C}$ de tamanho máximo.

GULOSO ($U, \mathcal{C}$)

- 1 $S \leftarrow \emptyset$
- 2 **enquanto** existe e em U tal que $S \cup \{e\} \in \mathcal{C}$ **faça**
- 3 $S \leftarrow S \cup \{e\}$
- 4 **devolva** S

Matroides e o método guloso

U : conjunto finito arbitrário.

$\mathcal{C}$: família não vazia de subconjuntos de U hereditária.

(subconjuntos de conjuntos em $\mathcal{C}$ estão em $\mathcal{C}$)

Problema 1: Dados U e $\mathcal{C}$,
encontrar um conjunto de $\mathcal{C}$ de tamanho máximo.

GULOSO ($U, \mathcal{C}$)

- 1 $S \leftarrow \emptyset$
- 2 **enquanto** existe e em U tal que $S \cup \{e\} \in \mathcal{C}$ **faça**
- 3 $S \leftarrow S \cup \{e\}$
- 4 **devolva** S

GULOSO encontra um conjunto **maximal** de $\mathcal{C}$.

Matroides e o método guloso

U : conjunto finito arbitrário.

$\mathcal{C}$: família não vazia de subconjuntos de U hereditária.

(subconjuntos de conjuntos em $\mathcal{C}$ estão em $\mathcal{C}$)

Problema 1: Dados U e $\mathcal{C}$,
encontrar um conjunto de $\mathcal{C}$ de tamanho máximo.

GULOSO ($U, \mathcal{C}$)

- 1 $S \leftarrow \emptyset$
- 2 **enquanto** existe e em U tal que $S \cup \{e\} \in \mathcal{C}$ **faça**
- 3 $S \leftarrow S \cup \{e\}$
- 4 **devolva** S

GULOSO encontra um conjunto **maximal** de $\mathcal{C}$.

Se $\mathcal{C}$ é um **matroide** então

todo conjunto maximal de $\mathcal{C}$ tem o mesmo tamanho.

Definição de matroides

U : conjunto finito arbitrário.

$\mathcal{C}$: família não vazia de subconjuntos de U hereditária.
(subconjuntos de conjuntos em $\mathcal{C}$ estão em $\mathcal{C}$)

Se $\mathcal{C}$ é um **matroide** então

GULOSO sempre encontra um conjunto de $\mathcal{C}$ de tamanho máximo.

Definição de matroides

U : conjunto finito arbitrário.

$\mathcal{C}$: família não vazia de subconjuntos de U hereditária.

(subconjuntos de conjuntos em $\mathcal{C}$ estão em $\mathcal{C}$)

Se $\mathcal{C}$ é um **matroide** então

GULOSO sempre encontra um conjunto de $\mathcal{C}$ de tamanho máximo.

$\mathcal{C}$ é um **matroide** se, para todo par A, B de conjuntos de $\mathcal{C}$
para os quais $|A| < |B|$, existe $e \in B \setminus A$ tal que $A \cup \{e\} \in \mathcal{C}$.

Definição de matroides

U : conjunto finito arbitrário.

$\mathcal{C}$: família não vazia de subconjuntos de U hereditária.

(subconjuntos de conjuntos em $\mathcal{C}$ estão em $\mathcal{C}$)

Se $\mathcal{C}$ é um **matroide** então

GULOSO sempre encontra um conjunto de $\mathcal{C}$ de tamanho máximo.

$\mathcal{C}$ é um **matroide** se, para todo par A, B de conjuntos de $\mathcal{C}$
para os quais $|A| < |B|$, existe $e \in B \setminus A$ tal que $A \cup \{e\} \in \mathcal{C}$.

Essa é a definição do CLRS.

Exemplos de matroides

Dado um espaço vetorial, a coleção de todos os conjuntos de vetores LI deste espaço é um matroide.

Exemplos de matroides

Dado um espaço vetorial, a coleção de todos os **conjuntos de vetores LI** deste espaço é um matroide.

Dado um grafo G , a coleção de todos os conjuntos de arestas de **florestas** de G é um matroide.

Exemplos de matroides

Dado um espaço vetorial, a coleção de todos os **conjuntos de vetores LI** deste espaço é um matroide.

Dado um grafo G , a coleção de todos os conjuntos de arestas de **florestas** de G é um matroide.

Grafo bipartido $G = (U \cup V, E)$.

M_U : todos os conjuntos S de arestas de G tq no máximo uma aresta de S é incidente a cada vértice de U .

M_U é um matroide.

Exemplos de matroides

Dado um espaço vetorial, a coleção de todos os conjuntos de vetores LI deste espaço é um matroide.

Dado um grafo G , a coleção de todos os conjuntos de arestas de florestas de G é um matroide.

Grafo bipartido $G = (U \cup V, E)$.

M_U : todos os conjuntos S de arestas de G tq no máximo uma aresta de S é incidente a cada vértice de U .

M_U é um matroide.

M_V : a coleção análoga com V no lugar de U .

Claro que M_V também é um matroide.

Exemplos de matroides

Dado um espaço vetorial, a coleção de todos os conjuntos de vetores LI deste espaço é um matroide.

Dado um grafo G , a coleção de todos os conjuntos de arestas de florestas de G é um matroide.

Grafo bipartido $G = (U \cup V, E)$.

M_U : todos os conjuntos S de arestas de G tq no máximo uma aresta de S é incidente a cada vértice de U .

M_U é um matroide.

M_V : a coleção análoga com V no lugar de U .

Claro que M_V também é um matroide.

E $M_U \cap M_V$? É ou não é um matroide?

Matroides e o método guloso

U : conjunto finito.

$\mathcal{C}$: família não vazia de subconjuntos de U hereditária.

Pesos positivos para os elementos de U .

Problema 2: Encontrar um conjunto de $\mathcal{C}$ de peso máximo.

Matroides e o método guloso

U : conjunto finito.

$\mathcal{C}$: família não vazia de subconjuntos de U hereditária.

Pesos positivos para os elementos de U .

Problema 2: Encontrar um conjunto de $\mathcal{C}$ de peso máximo.

GULOSO ($U, w, \mathcal{C}$)

- 1 $(e_1, \dots, e_n) \leftarrow \text{ORDENE}(U, w)$ $\triangleright$ ordem decrescente dos pesos
- 2 $S \leftarrow \emptyset$
- 3 **para** $i \leftarrow 1$ **até** n **faça**
- 4 se $S \cup \{e_i\} \in \mathcal{C}$
- 5 **então** $S \leftarrow S \cup \{e_i\}$
- 6 **devolva** S

Matroides e o método guloso.

U : conjunto finito.

$\mathcal{C}$: família não vazia de subconjuntos de U hereditária.

Pesos positivos para os elementos de U .

Problema 2: Encontrar um conjunto de $\mathcal{C}$ de peso máximo.

GULOSO ($U, w, \mathcal{C}$)

```
1   $(e_1, \dots, e_n) \leftarrow \text{ORDENE}(U, w)$   $\triangleright$  ordem decrescente dos pesos
2   $S \leftarrow \emptyset$ 
3  para  $i \leftarrow 1$  até  $n$  faça
4      se  $S \cup \{e_i\} \in \mathcal{C}$ 
5          então  $S \leftarrow S \cup \{e_i\}$ 
6  devolva  $S$ 
```

Teorema: Se $\mathcal{C}$ é um matroide,
então o algoritmo acima resolve o **Problema 2**.

Prova do teorema

Teorema: Se $\mathcal{C}$ é um matroide,
então o algoritmo anterior resolve o **Problema 2**.

Esboço da prova:

Seja O um conjunto de $\mathcal{C}$ de peso máximo.

Seja S o conjunto devolvido pelo algoritmo.

Prova do teorema

Teorema: Se $\mathcal{C}$ é um matroide,
então o algoritmo anterior resolve o **Problema 2**.

Esboço da prova:

Seja O um conjunto de $\mathcal{C}$ de peso máximo.

Seja S o conjunto devolvido pelo algoritmo.

Se $O = S$, não há nada a provar. Senão, note que $S \not\subseteq O$.

Prova do teorema

Teorema: Se $\mathcal{C}$ é um matroide,
então o algoritmo anterior resolve o **Problema 2**.

Esboço da prova:

Seja O um conjunto de $\mathcal{C}$ de peso máximo.

Seja S o conjunto devolvido pelo algoritmo.

Se $O = S$, não há nada a provar. Senão, note que $S \not\subseteq O$.

Seja i o menor possível tal que $e_i \in S \setminus O$.

Prova do teorema

Teorema: Se $\mathcal{C}$ é um matroide, então o algoritmo anterior resolve o **Problema 2**.

Esboço da prova:

Seja O um conjunto de $\mathcal{C}$ de peso máximo.

Seja S o conjunto devolvido pelo algoritmo.

Se $O = S$, não há nada a provar. Senão, note que $S \not\subseteq O$.

Seja i o menor possível tal que $e_i \in S \setminus O$.

Seja $A = (S \cap O) \cup \{e_i\}$. Note que $A \subseteq S$, logo $A \in \mathcal{C}$.

Seja A' um conjunto maximal de $\mathcal{C}$ tal que $A \subseteq A' \subseteq O \cup \{e_i\}$.

Prova do teorema

Teorema: Se $\mathcal{C}$ é um matroide, então o algoritmo anterior resolve o **Problema 2**.

Esboço da prova:

Seja O um conjunto de $\mathcal{C}$ de peso máximo.

Seja S o conjunto devolvido pelo algoritmo.

Se $O = S$, não há nada a provar. Senão, note que $S \not\subseteq O$.

Seja i o menor possível tal que $e_i \in S \setminus O$.

Seja $A = (S \cap O) \cup \{e_i\}$. Note que $A \subseteq S$, logo $A \in \mathcal{C}$.

Seja A' um conjunto maximal de $\mathcal{C}$ tal que $A \subseteq A' \subseteq O \cup \{e_i\}$.

Como $\mathcal{C}$ é um matroide, $|A'| = |O|$.

De fato:

$\mathcal{C}$ é um **matroide** se, para todo par A, B de conjuntos de $\mathcal{C}$ para os quais $|A| < |B|$, existe $e \in B \setminus A$ tal que $A \cup \{e\} \in \mathcal{C}$.

Prova do teorema

Teorema: Se $\mathcal{C}$ é um matroide,
então o algoritmo anterior resolve o **Problema 2**.

Esboço da prova:

Seja O um conjunto de $\mathcal{C}$ de peso máximo.

Seja S o conjunto devolvido pelo algoritmo.

Se $O = S$, não há nada a provar. Senão, note que $S \not\subseteq O$.

Seja i o menor possível tal que $e_i \in S \setminus O$.

Seja $A = (S \cap O) \cup \{e_i\}$. Note que $A \subseteq S$, logo $A \in \mathcal{C}$.

Seja A' um conjunto maximal de $\mathcal{C}$ tal que $A \subseteq A' \subseteq O \cup \{e_i\}$.

Como $\mathcal{C}$ é um matroide, $|A'| = |O|$.

Logo $A' = O \setminus \{e_j\} \cup \{e_i\}$, com $j > i$.

Portanto o peso de A' é maior ou igual ao peso de O ,
ou seja, A' tem peso máximo.

Prova do teorema

Teorema: Se $\mathcal{C}$ é um matroide, então o algoritmo anterior resolve o **Problema 2**.

Esboço da prova:

Seja O um conjunto de $\mathcal{C}$ de peso máximo.

Seja S o conjunto devolvido pelo algoritmo.

Se $O = S$, não há nada a provar. Senão, note que $S \not\subseteq O$.

Seja i o menor possível tal que $e_i \in S \setminus O$.

Seja $A = (S \cap O) \cup \{e_i\}$. Note que $A \subseteq S$, logo $A \in \mathcal{C}$.

Seja A' um conjunto maximal de $\mathcal{C}$ tal que $A \subseteq A' \subseteq O \cup \{e_i\}$.

Como $\mathcal{C}$ é um matroide, $|A'| = |O|$.

Logo $A' = O \setminus \{e_j\} \cup \{e_i\}$, com $j > i$.

Portanto o peso de A' é maior ou igual ao peso de O , ou seja, A' tem peso máximo.

Repetindo esse processo, provamos que S e O têm o mesmo peso.

Consequência nos exemplos

Dado um espaço vetorial, a coleção de todos os **conjuntos de vetores LI** deste espaço é um matroide.

Se cada vetor tiver um peso, podemos encontrar, com o algoritmo guloso, uma base de peso máximo.

Consequência nos exemplos

Dado um espaço vetorial, a coleção de todos os **conjuntos de vetores LI** deste espaço é um matroide.

Se cada vetor tiver um peso, podemos encontrar, com o algoritmo guloso, uma base de peso máximo.

Dado um grafo G , a coleção de todos os conjuntos de arestas de **florestas** de G é um matroide.

Se cada aresta tiver um peso, podemos encontrar, com o algoritmo guloso, uma floresta de peso máximo (Borůvka/Kruskal).

Exemplos de matroides

Grafo bipartido $G = (U \times V, E)$.

M_U : todos os conjuntos S de arestas de G tq no máximo uma aresta de S é incidente a cada vértice de U .

M_V : a coleção analoga com V no lugar de U .

M_U e M_V são matroides.

Exemplos de matroides

Grafo bipartido $G = (U \times V, E)$.

M_U : todos os conjuntos S de arestas de G tq no máximo uma aresta de S é incidente a cada vértice de U .

M_V : a coleção analoga com V no lugar de U .

M_U e M_V são matroides.

O que é $M_U \cap M_V$?

O que é um conjunto da coleção $M_U \cap M_V$ em G ?

Exemplos de matroides

Grafo bipartido $G = (U \times V, E)$.

M_U : todos os conjuntos S de arestas de G tq no máximo uma aresta de S é incidente a cada vértice de U .

M_V : a coleção analoga com V no lugar de U .

M_U e M_V são matroides.

O que é $M_U \cap M_V$?

O que é um conjunto da coleção $M_U \cap M_V$ em G ?

Cada conjunto de $M_U \cap M_V$ é um **emparelhamento** em G .

Exemplos de matroides

Grafo bipartido $G = (U \times V, E)$.

M_U : todos os conjuntos S de arestas de G tq no máximo uma aresta de S é incidente a cada vértice de U .

M_V : a coleção analoga com V no lugar de U .

M_U e M_V são matroides.

O que é $M_U \cap M_V$?

O que é um conjunto da coleção $M_U \cap M_V$ em G ?

Cada conjunto de $M_U \cap M_V$ é um **emparelhamento** em G .

Esta coleção é ou não é um matroide?

Exemplos de matroides

Grafo bipartido $G = (U \times V, E)$.

M_U : todos os conjuntos S de arestas de G tq no máximo uma aresta de S é incidente a cada vértice de U .

M_V : a coleção analoga com V no lugar de U .

M_U e M_V são matroides.

O que é $M_U \cap M_V$?

O que é um conjunto da coleção $M_U \cap M_V$ em G ?

Cada conjunto de $M_U \cap M_V$ é um **emparelhamento** em G .

Esta coleção é ou não é um matroide?

Não é... (nem todo emparelhamento maximal é máximo).

Mas é a **interseção** de dois matroides.

Exemplos de matroides

Grafo bipartido $G = (U \times V, E)$.

M_U : todos os conjuntos S de arestas de G tq no máximo uma aresta de S é incidente a cada vértice de U .

M_V : a coleção analoga com V no lugar de U .

M_U e M_V são matroides.

$M_U \cap M_V$ é a coleção dos **emparelhamentos** de G .

Esta coleção não é um matroide.

Mas é a **interseção** de dois matroides.

Exemplos de matroides

Grafo bipartido $G = (U \times V, E)$.

M_U : todos os conjuntos S de arestas de G tq no máximo uma aresta de S é incidente a cada vértice de U .

M_V : a coleção analoga com V no lugar de U .

M_U e M_V são matroides.

$M_U \cap M_V$ é a coleção dos **emparelhamentos** de G .

Esta coleção não é um matroide.

Mas é a **interseção** de dois matroides.

Existe algoritmo polinomial para encontrar um conjunto (de peso) máximo na interseção de dois matroides (Edmonds).