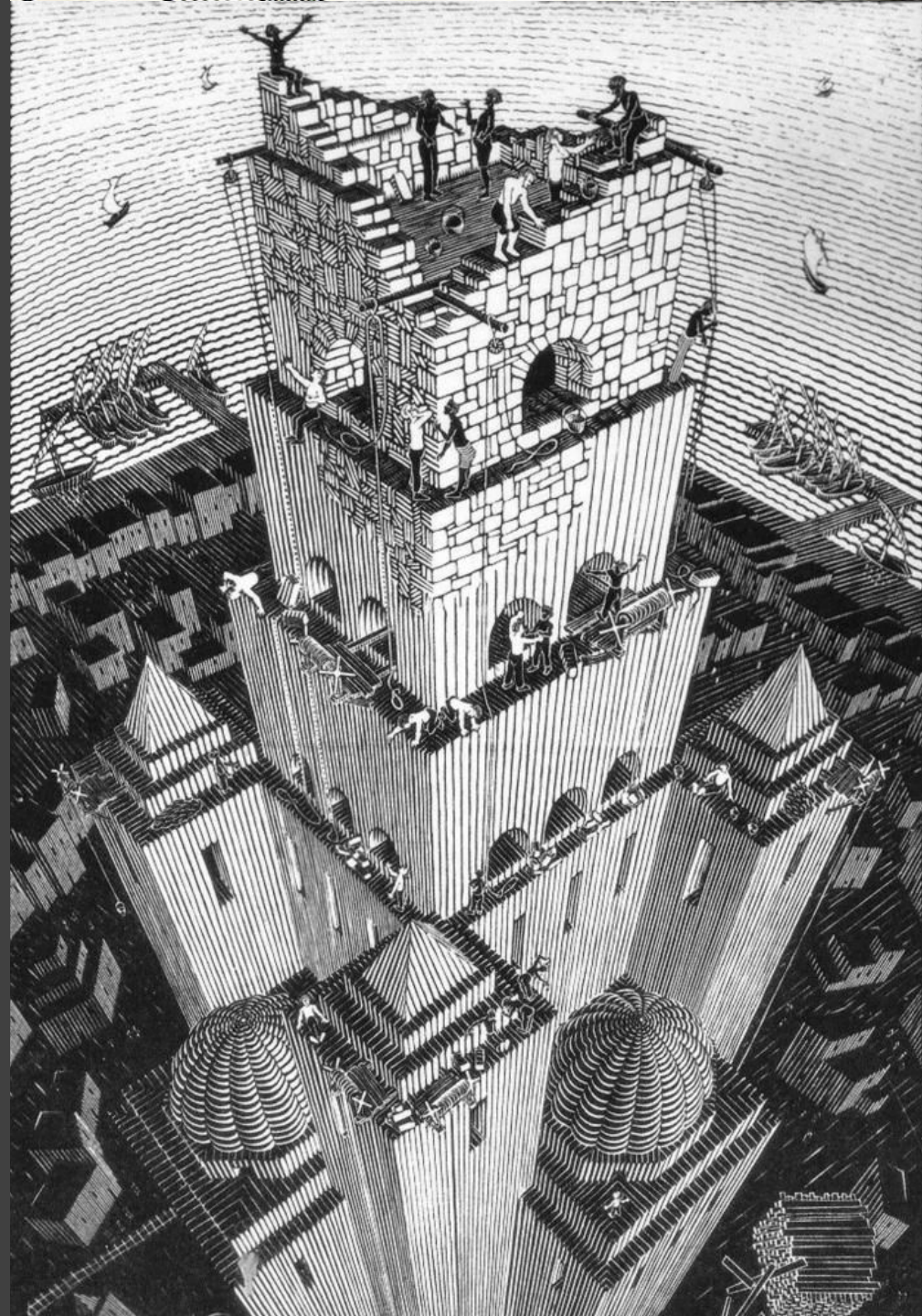
Università Roma Tre – Dipartimento di Matematica e Fisica
IN440 – Ottimizzazione Combinatoria
Prof. Marco Liverani

Algoritmi approssimanti per problemi NP-completi



Algoritmi per la soluzione di problemi super-polinomiali

- Spesso è necessario implementare un algoritmo per la soluzione di problemi NP-completi o per i quali non si conosce comunque un algoritmo risolutivo efficiente di complessità polinomiale
- In questi casi ci si presentano tre alternative:
  1. implementare un algoritmo di **ricerca esaustiva** della soluzione esatta del problema: questa strada è praticabile se l'istanza del problema da risolvere è di piccola dimensione; in tal caso anche una complessità computazionale super-polinomiale può risultare accettabile;
  2. individuare algoritmi di complessità polinomiale per la **soluzione di specifiche famiglie di istanze del problema** (casi particolari);
  3. produrre **soluzioni approssimate**, «quasi ottime», calcolabili in tempo polinomiale
- Quest'ultima opzione è accettabile solo se l'approssimazione è buona o comunque compatibile con l'applicazione per cui la si intende utilizzare

Algoritmi approssimanti

- Sia (A, f) un'istanza di un problema di ottimizzazione e sia $a^* \in A$ una soluzione ottima del problema; sia $f^* = f(a^*)$ il valore della funzione obiettivo per una soluzione ottima del problema; indichiamo come al solito con n la dimensione dell'istanza del problema
- Risulta quindi $f^* = \min_{a \in A} f(a)$ se (A, f) è un problema di minimizzazione e $f^* = \max_{a \in A} f(a)$ se è un problema di massimizzazione
- Sia $a' \in A$ una **soluzione approssimata** dello stesso problema; indichiamo con $f' = f(a')$ il valore della funzione obiettivo per tale soluzione

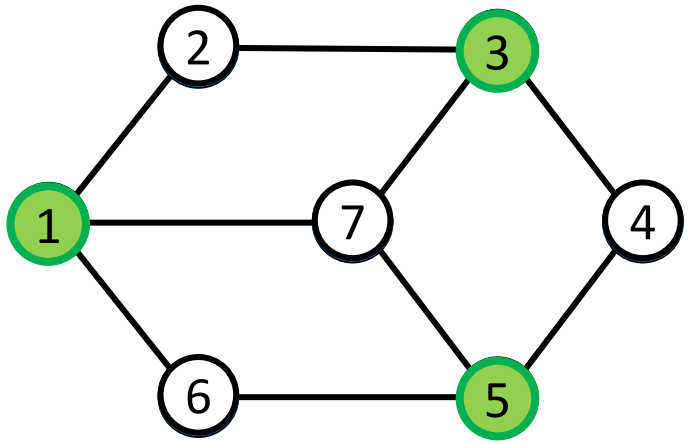
  con il termine *soluzione approssimata* indichiamo una configurazione delle variabili del problema per cui la funzione obiettivo non assume il valore ottimo (il valore massimo o minimo), ma un valore che si avvicina al valore della funzione obiettivo per una soluzione ottima
- Risulterà $f^* / f' \geq 1$ se il problema è di massimizzazione e $f' / f^* \geq 1$ se il problema è, al contrario, un problema di minimizzazione
- Il **rapporto di approssimazione** dell'algoritmo approssimato è $\rho(n) = \max\{ f^*/f', f'/f^* \}$ e tale algoritmo si dirà algoritmo **$\rho(n)$-approssimato**

Algoritmi approssimanti

- Un algoritmo $\rho(n)$-approssimato è dunque un algoritmo in grado di produrre una soluzione approssimata del problema di ottimizzazione (A, f), con rapporto di approssimazione $\rho(n)$
- Nella progettazione di un algoritmo approssimato è importante riuscire a stimare a priori il rapporto di approssimazione che tale algoritmo è in grado di ottenere nel caso peggiore
- L'ideale è progettare un algoritmo approssimato che possa produrre una soluzione con un rapporto di approssimazione arbitrario, stabilito di volta in volta fornendo un parametro $\varepsilon = \rho(n) - 1$
- Si vuole quindi progettare un algoritmo in grado di produrre soluzioni $(1 + \varepsilon)$-approssimate; naturalmente, in generale, quanto più piccolo sarà il valore di ε, tanto maggiore sarà il numero di operazioni eseguite dall'algoritmo
- Vediamo di seguito un paio di esempi applicati alla soluzione approssimata di problemi NP-completi

Copertura di vertici

- Dato un grafo $G = (V, E)$, un sottoinsieme $V' \subseteq V$ è una **copertura di vertici** per G se per ogni $(u, v) \in E$ risulta $u \in V'$ oppure $v \in V'$
- Problema di ottimizzazione (minimizzazione): dato $G = (V, E)$ trovare la copertura di vertici di minima cardinalità



- Il problema posto in forma decisionale è NP-completo
- La ricerca esaustiva della soluzione ottima del problema di ottimizzazione ha una complessità computazionale super-polinomiale: $O(m\ 2^n)$

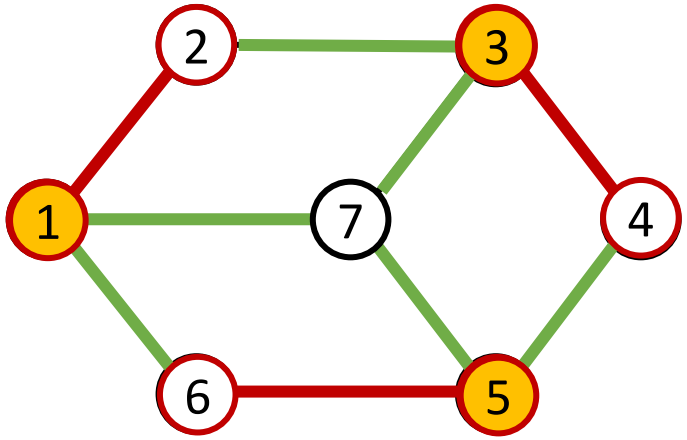
Algoritmo 53 VERTEXCOVERESAUSTIVO(G)

Input: Un grafo $G, = (V, E)$
Output: Una copertura di vertici $C \subseteq V$ di cardinalità minima

```
1:  C_min = V
2:  per ogni C ∈ P(V) ripeti
3:      flag = true
4:      per ogni e ∈ E(G) e flag = true ripeti
5:          sia e = (u, v)
6:          se u ∉ C e v ∉ C allora
7:              flag = false
8:          fine-condizione
9:      fine-ciclo
10:     se |C| < |C_min| allora
11:         C_min = C
12:     fine-condizione
13: fine-ciclo
14: return C_min
```

Copertura di vertici approssimata

- Dato un grafo $G = (V, E)$, un sottoinsieme $V' \subseteq V$ è una **copertura di vertici** per G se per ogni $(u, v) \in E$ risulta $u \in V'$ oppure $v \in V'$
- Problema di ottimizzazione (minimizzazione): dato $G = (V, E)$ trovare la copertura di vertici di minima cardinalità
- Possiamo proporre un algoritmo approssimante:



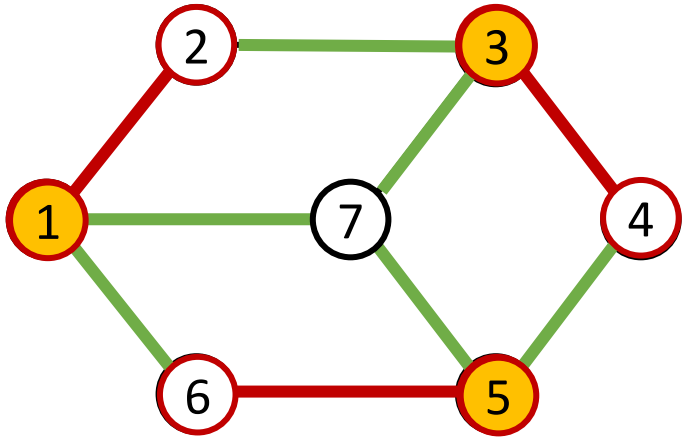
Algoritmo 40 VERTEXCOVERAPPROSSIMATO(G)

Input: Un grafo $G, = (V, E)$
Output: Una copertura di vertici $C \subseteq V$ al massimo di dimensione doppia rispetto alla soluzione ottima

1: $C = \emptyset$, $E' = E(G)$
2: **fintanto che** $E' \neq \emptyset$ **ripeti**
3: sia $(u, v) \in E'$ uno spigolo arbitrario
4: $C = C \cup \{u, v\}$
5: rimuovi da E' ogni spigolo incidente in u e in v
6: **fine-ciclo**

Copertura di vertici approssimata

- **L'algoritmo è polinomiale**: esegue m iterazioni del ciclo 2-6 e per ogni spigolo (u, v) esegue un'operazione per ogni spigolo incidente su u e v
- **L'algoritmo è corretto**: restituisce una copertura di vertici; infatti termina quando tutti gli spigoli sono stati «coperti» da qualche vertice (la condizione di fine-ciclo è proprio questa)
- L'algoritmo è 2-approssimante: $\rho(n) = 2$



Algoritmo 40 VERTEXCOVERAPPROSSIMATO(G)

Input: Un grafo $G, = (V, E)$
Output: Una copertura di vertici $C \subseteq V$ al massimo di dimensione doppia rispetto alla soluzione ottima

1: $C = \emptyset$, $E' = E(G)$
2: **fintanto che** $E' \neq \emptyset$ **ripeti**
3: sia $(u, v) \in E'$ uno spigolo arbitrario
4: $C = C \cup \{u, v\}$
5: rimuovi da E' ogni spigolo incidente in u e in v
6: **fine-ciclo**

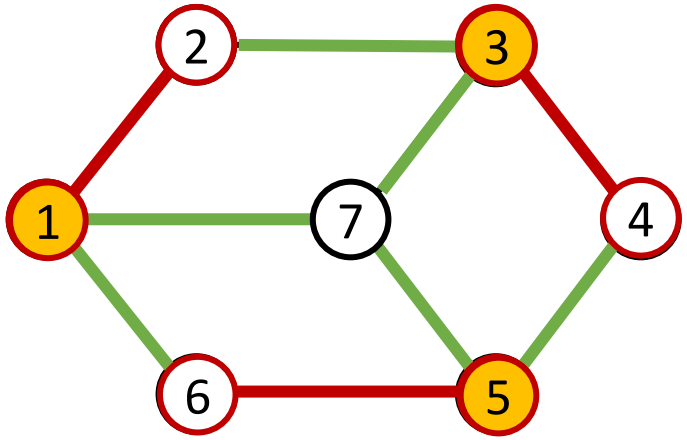
Copertura di vertici approssimata

- **Teorema**. La copertura di vertici restituita dall'algoritmo approssimante ha una dimensione al più doppia rispetto ad una copertura ottima

  Dimostrazione. Sia $G = (V, E)$ un grafo. Sia $A \subseteq E$ l'insieme di spigoli scelti per costruire la copertura C.

  Una copertura di A è costituita dall'insieme composto da un estremo per ciascuno spigolo; due spigoli infatti non possono avere nessun estremo in comune perché dopo aver scelto lo spigolo (u, v) a riga 3, vengono rimossi tutti gli spigoli incidenti sia su u che su v (riga 5).

  Quindi la copertura ottima C^* è tale che $|C^*| \geq |A|$. Inoltre $|C| = 2\,|A|$ per come sono stati costruiti A e C. Quindi $|C| = 2\,|A| \leq 2\,|C^*|$ ■



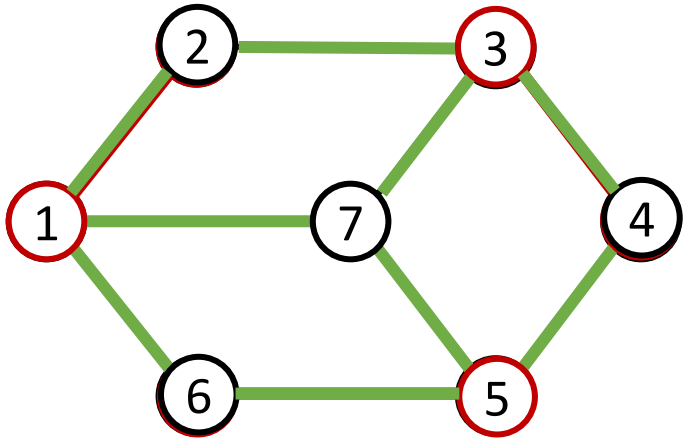
Algoritmo 40 VERTEXCOVERAPPROSSIMATO(G)

Input: Un grafo $G, = (V, E)$
Output: Una copertura di vertici $C \subseteq V$ al massimo di dimensione doppia rispetto alla soluzione ottima

1: $C = \emptyset$, $E' = E(G)$
2: **fintanto che** $E' \neq \emptyset$ **ripeti**
3: sia $(u, v) \in E'$ uno spigolo arbitrario
4: $C = C \cup \{u, v\}$
5: rimuovi da E' ogni spigolo incidente in u e in v
6: **fine-ciclo**

Copertura di vertici approssimata (migliorata)

- Possiamo provare a migliorare l'efficacia dell'algoritmo, aggiungendo un procedimento euristico (righe 7–11) per ridurre il numero di vertici presenti nella copertura [Liverani 2021 ☺]
- Anche questo algoritmo ha complessità polinomiale (7–11 è polinomiale ⇒ la soluzione è approssimata, altrimenti ... $\mathrm{P} = \mathrm{NP}$! ☺)



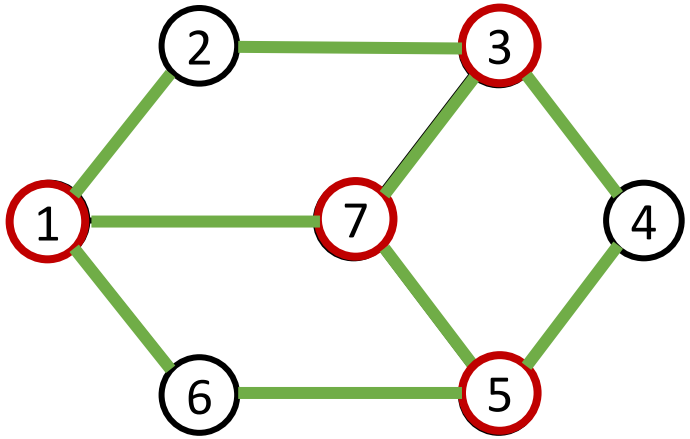
Algoritmo 41 VertexCover(G)

1: $C = \emptyset$, $E' = E(G)$
2: **fintanto che** $E' \neq \emptyset$ **ripeti**
3: sia $(u, v) \in E'$ uno spigolo arbitrario
4: $C = C \cup \{u, v\}$
5: rimuovi da E' ogni spigolo incidente su u o v
6: **fine-ciclo**
7: **per ogni** $u \in C$ in ordine di grado non decrescente su G **ripeti**
8: **se** ogni v adiacente a u è in C **allora**
9: rimuovi il vertice u da C
10: **fine-condizione**
11: **fine-ciclo**

Copertura di vertici golosa

- Una diversa strategia è basata su una **tecnica greedy**: ad ogni passo si elimina dal grafo e si inserisce nella copertura il vertice di grado massimo del «grafo residuo»

- Anche in questo caso l'algoritmo è polinomiale e il risultato è una copertura, ma non è garantito che sia minimale



Algoritmo 42 GREEDYVERTEXCOVERAPPROSSIMATO(G)

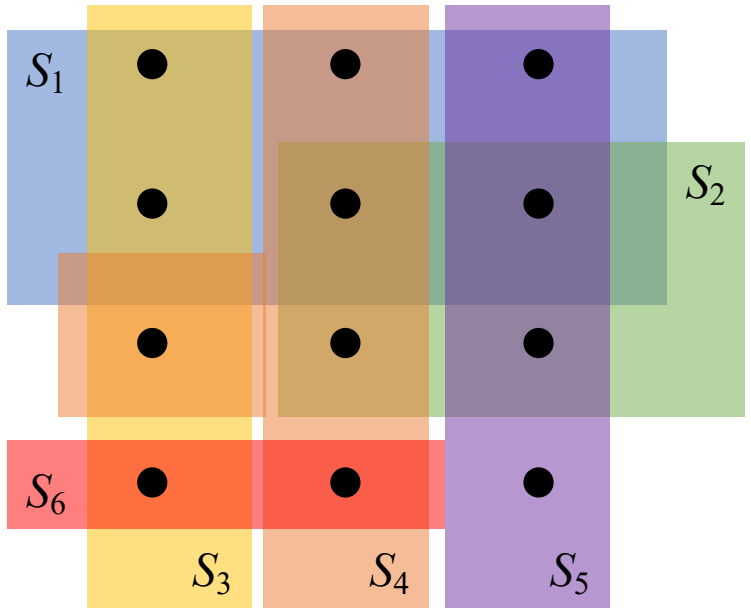
Input: Un grafo $G, = (V, E)$

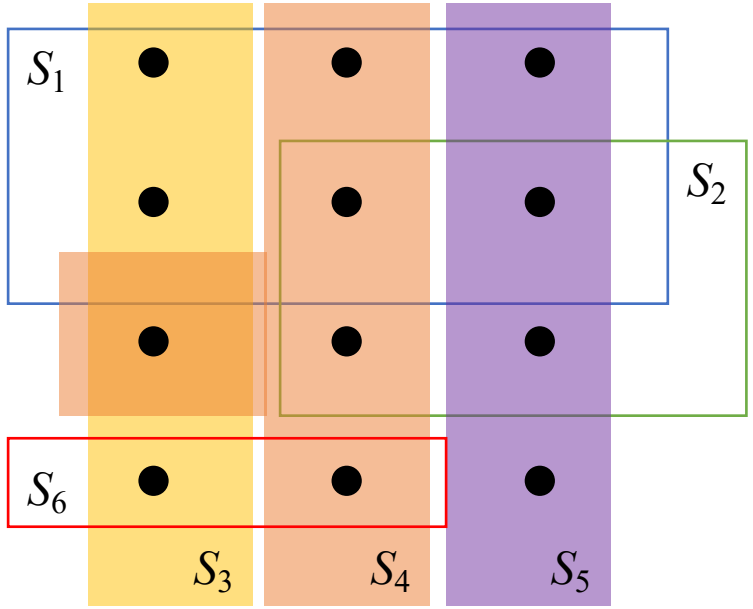
Output: Una copertura di vertici $C \subseteq V$ al massimo di dimensione doppia rispetto alla soluzione ottima

1: $C := \emptyset$, $E' := E(G)$, $V' := V(G)$, $G' := (V', E')$
2: **fintanto che** $E' \neq \emptyset$ **ripeti**
3: sia $u \in V'$ il vertice di grado massimo in G'
4: $C := C \cup \{u\}$
5: rimuovi da E' ogni spigolo incidente in u; rimuovi u da V'
6: **fine-ciclo**

Copertura di insiemi

- **Set Cover**: consideriamo un insieme X e una famiglia $\mathcal{F}$ di sottoinsiemi di X tale che ogni elemento di X appartenga ad almeno un insieme della famiglia $\mathcal{F}$
- Si vuole trovare $\mathcal{C} \subseteq \mathcal{F}$ di cardinalità minima, i cui membri coprano X
- Esempio: $\mathcal{F} = \{S_1, S_2, S_3, S_4, S_5, S_6\}$ Soluzione ottima: $\mathcal{C} = \{S_3, S_4, S_5\}$





Copertura di insiemi: ricerca esaustiva della soluzione ottima

- **Set Cover**: consideriamo un insieme X e una famiglia $\mathscr{F}$ di sottoinsiemi di X tale che ogni elemento di X appartenga ad almeno un insieme della famiglia $\mathscr{F}$
- Si vuole trovare $\mathcal{C} \subseteq \mathscr{F}$ di cardinalità minima, i cui membri coprano X
- Il problema Set Cover è NP-completo
- La ricerca esaustiva della soluzione ha complessità computazionale $O(|X|\, 2^{|\mathcal{F}|})$

Algoritmo 54 SETCOVERESAUSTIVO($X, \mathcal{F} \subseteq \mathcal{P}(X)$)

Input: L'insieme X ed una famiglia $\mathcal{F}$ di sottoinsiemi che coprono X
Output: Un sottoinsieme $\mathcal{C}$ di $\mathcal{F}$ di cardinalità minima che copre X

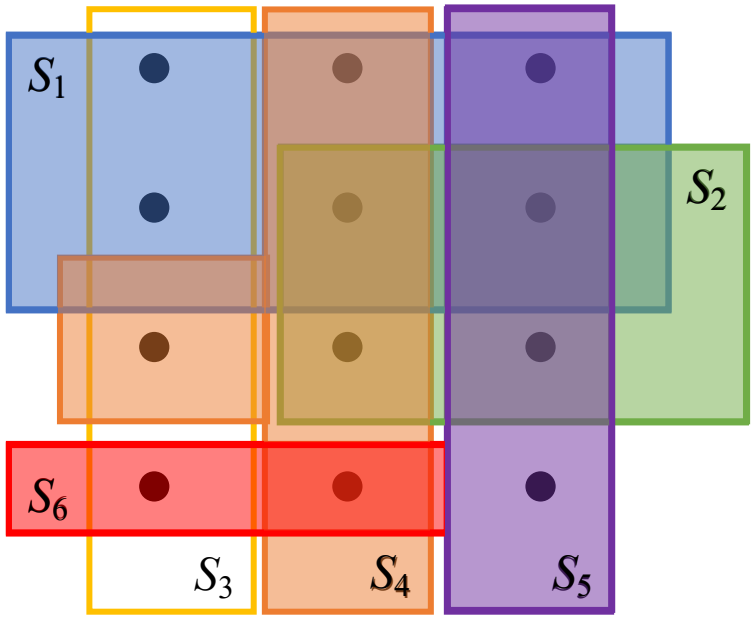
```
1: sia F_min = F
2: per ogni F ∈ P(F) ripeti
3:     sia F = {S_1, ..., S_k}
4:     sia Y = X
5:     per ogni S_i ∈ F ripeti
6:         sia Y = Y \ S_i
7:     fine-ciclo
8:     se Y = ∅ e |F| < |F_min| allora
9:         F_min = F
10:    fine-condizione
11: fine-ciclo
12: return F_min
```

Copertura di insiemi: algoritmo goloso approssimato

- **Set Cover**: consideriamo un insieme X e una famiglia $\mathcal{F}$ di sottoinsiemi di X tale che ogni elemento di X appartenga ad almeno un insieme della famiglia $\mathcal{F}$
- Si vuole trovare $\mathcal{C} \subseteq \mathcal{F}$ di cardinalità minima, i cui membri coprano X
- Il problema Set Cover è NP-completo
- Anche in questo caso proponiamo un algoritmo approssimante basato sulla **tecnica greedy**
- Esempio: $\mathcal{F} = \{S_1, S_2, S_3, S_4, S_5, S_6\}$

Soluzione ottima: $\mathcal{C} = \{S_3, S_4, S_5\}$

Soluzione approssimata: $\mathcal{C} = \{S_1 \;\; S_2 \;\; S_4 \;\; S_6 \;\; S_5\}$



Algoritmo 43 GREEDYSETCOVER$(X, \mathcal{F})$

Input: L'insieme X ed una famiglia $\mathcal{F}$ di sottoinsiemi che coprono X
Output: Un sottoinsieme $\mathcal{C}$ di $\mathcal{F}$ di cardinalità minima che copre X

1: $U := X$, $\mathcal{C} := \emptyset$
2: **fintanto che** $U \neq \emptyset$ **ripeti**
3: seleziona $S \in \mathcal{F}$ che massimizza $|S \cap U|$
4: $U := U \setminus S$
5: $\mathcal{C} := \mathcal{C} \cup \{S\}$
6: **fine-ciclo**

Riferimenti bibliografici

- Cormen, Leiserson, Rivest, Stein, «*Introduzione agli algoritmi e strutture dati*», terza edizione, McGraw-Hill (Cap. 35, 35.1, 35.3)
- Michael R. Garey, David S. Johnson, «*Computer and Intractability. A Guide to the Theory of NP-Completeness*», Freeman and Company (1979)