# Algoritmi e Strutture Dati

Elaborato a.a. 2020-2021

# Problema: calcolo degli hitting set minimali

- Input: una collezione finita N di insiemi finiti dove gli elementi di ogni insieme appartengono al dominio M
- Output: tutti gli hitting set minimali di N

# Hitting set e hitting set minimali

- Hitting Set (HS) di una collezione N di insiemi definita sul dominio M = insieme di elementi appartenenti a M che presenta una intersezione non vuota con ciascun insieme della collezione N
- Hitting Set minimale (MHS) = HS tale che nessun suo sottoinsieme è un HS

#### Calcolo dei MHS

- Il problema di decisione teso a stabilire se esista un hitting set di dimensione ≤ k per una collezione di insiemi data è NP completo [Karp 1972] [Garey and Johnson 1979]
- In questo tema per elaborato vengono proposti solo algoritmi esatti completi (cioè che calcolano tutti e soli i MHS relativi alla collezione data)

### Calcolo dei MHS: esempio

```
N = \{ \{B3, B4\}, \}
       {A1,A2,B4},
        {A2,A5,B3,B4} }
M = \{A1, A2, A5, B3, B4\}
MHS = \{ \{B4\}, \}
          {A1,B3},
          {A2,B3}}
```

N.B. M coincide con l'unione degli insiemi della collezione N oppure è un superinsieme di tale unione

#### Rilevanza dei MHS

- Usati (non solo) per
  - Convertire un'espressione booleana dalla forma normale disgiuntiva (DNF) a quella congiuntiva (CNF) e viceversa
  - Colpire un insieme di landmark nel planning Al
  - Isolare difetti nel codice di programmi sw
  - Determinare le diagnosi (esse sono i MHS dei conflitti) negli approcci diagnostici AI basati sui modelli

#### Input: matrice

Ogni elemento di M è univocamente identificato da un indice intero appartenente all'intervallo [1 .. |M|]

#### Esempio

- $M = \{A1,A2,B3,B4,A5\}$
- 1 2 3 4 5

### Input: matrice (cont.)

Analogamente ogni elemento di N è univocamente identificato da un indice intero appartenente all'intervallo [1 .. |N|]

#### Esempio

```
N = { {B3,B4}, 1 {A1,A2,B4}, 2 {A2,A5,B3,B4} }
```

### Input: matrice (cont.)

- I dati d'ingresso del problema possono essere rappresentati come una matrice A<sub>N,M</sub>, dove il valore del componente a<sub>i,j</sub> della matrice è 1 se l'elemento (di M) di indice j appartiene all'insieme (in N) di indice i, 0 altrimenti
- Assunzione: ogni riga della matrice contiene almeno un 1 (cioè nessun insieme della collezione N è vuoto)

# Esempio: matrice

		{A1, A2, B3, B4, A5}					
		1	2	3	4	5	
• {B3,B4}	1	0	0	1	1	0	
• {A1,A2,B4}	2	1	1	0	1	0	
• {A2,B3,B4,A5}	3	0	1	1	1	1	

# Ordine lessicografico

- A ogni elemento di M è stato fatto corrispondere un valore intero, secondo un ordine crescente da sinistra a destra. In questo modo si è stabilito un ordinamento totale fra gli elementi del dominio, in virtù del quale è possibile confrontare fra loro sia singoli elementi contenuti in M sia sottoinsiemi di M
- Secondo l'esempio della pagina precedente, le seguenti condizioni che usano l'operatore di confronto < (e le duali, che usano l'operatore >) sono tutte vere (così come molte altre non riportate):
  - A1 < A2,</li>
  - B3 < A5,
  - {A2, B4} < {B4, B5}
  - {A1, B3} < {A2}

### Ordine lessicografico (cont.)

- In virtù dell'ordinamento totale degli elementi di M, dato un (sotto)insieme di M, è possibile indicarne il valore minimo (min) e massimo (max). Facendo sempre riferimento all'esempio di pag. 10, sono vere (fra le altre) le seguenti uguaglianze
  - min(M) = A1
  - max(M) = A5
  - $min(\{A2, B4\}) = A2$
  - $max({A2, B3, B4}) = B4$

# Ordine lessicografico (cont.)

- In virtù dell'ordinamento totale degli elementi di M è inoltre possibile, dato un elemento di M, determinarne il successore (succ) e il predecessore (pred). Facendo sempre riferimento all'esempio di pag. 10, sono vere (fra le altre) le seguenti uguaglianze
  - succ(A1) = A2
  - pred(B4) = B3
- Convenzionalmente sono vere le uguaglianze
  - pred(min(M)) =  $\varepsilon_{min}$
  - $succ(max(M)) = \varepsilon_{max}$

dove  $\epsilon_{\min}$  è un elemento nullo che, secondo l'ordine lessicografico, precede ogni elemento di M e  $\epsilon_{\max}$  è un elemento nullo che, secondo l'ordine lessicografico, segue ogni elemento di M

• Convenzionalmente è vero che min( $\varnothing$ ) = max( $\varnothing$ ) =  $\varepsilon_{\min}$ 

#### **ALGORITMO DI BASE**

### Algoritmo MBASE

- Lo pseudocodice di un semplice algoritmo per il calcolo di tutti e soli i MHS che <u>sfrutta l'ordine lessicografico</u> degli elementi del dominio M è fornito a pag. 17
- MBASE genera (riga 6) e analizza (riga 7) uno per uno, in ordine lessicografico, i sottoinsiemi del dominio M
- Alcuni dei sottoinsiemi generati (quelli classificati come KO) saranno scartati, altri (quelli classificati come MHS) verranno prodotti in uscita, i rimanenti (classificati come OK) verranno inseriti, nell'ordine in cui sono stati generati (quello lessicografico), in una coda (Q)

### Algoritmo MBASE (cont.)

- MBASE è una sorta di template, con un funzionamento che cambia a seconda dell'implementazione del modulo CHECK che esso invoca
- CHECK analizza un sottoinsieme non vuoto  $\Gamma$  di M e ritorna un valore scalare fra tre possibili (OK, KO, MHS), dove
  - OK significa che  $\Gamma$  non è un MHS ma forse potrebbe diventarlo, aggiungendogli ulteriori elementi di M
  - KO significa che  $\Gamma$  non è un MHS né può diventarlo
  - MHS significa che  $\Gamma$  è un MHS

### Algoritmo MBASE: pseudocodice

#### Algorithm 1 Main program ▷ A è la matrice 1: procedure MBase(A) Enqueue( $Q, \emptyset$ ) prima di questo inserimento, la coda Q era vuota 2: while Q is not empty do 3: $\Lambda \leftarrow \text{Dequeue}(Q)$ 4:for $e \leftarrow succ(max(\Lambda))$ to max(M) do 5: $\Gamma \leftarrow \Lambda \cup \{e\}$ 6: result $\leftarrow$ Check( $\Gamma$ ) 7:if (result = OK) $\land$ (e $\neq$ max(M)) then 8: Enqueue( $Q, \Gamma$ ) 9: else if result = MHS then 10: $Output(\Gamma)$ 11:

#### Modulo OUTPUT

- Output semplicemente si occupa di annoverare fra i MHS relativi all'istanza corrente del problema il parametro  $\Gamma$  che gli viene passato. L'implementazione di Output può essere la più varia: potrebbe trattarsi semplicemente di scrivere (il contenuto di)  $\Gamma$  sullo standard output o su un altro dispositivo di uscita oppure di inserire  $\Gamma$  in una lista che, al termine dell'esecuzione di MBASE, viene ritornata al chiamante, ecc.
- Ciascun MHS, essendo un insieme definito sul dominio M, può essere rappresentato come avviene per un insieme della collezione N, cioè come la riga di una matrice avente lo stesso numero di colonne della matrice d'ingresso A. In tal modo si ottiene una matrice d'uscita, il cui formato può coincidere con quello della matrice in ingresso
- Per il formato della matrice in ingresso, si veda la sezione «Sperimentazione» di questo documento

#### Vettori rappresentativi

- A ciascun (sotto)insieme  $\Sigma$  di M compete un vettore rappresentativo  $C_{\Sigma}$  il cui numero di cella è pari a |N|. Il contenuto di ciascuna cella  $\sigma_i$  del vettore è
  - un elemento η∈Σ; questo significa che η è l'unico elemento di Σ che colpisce  $N_i$
  - il valore convenzionale x, dove x  $\notin$  M; questo significa che in  $\Sigma$  esistono almeno due elementi che colpiscono N<sub>i</sub>
  - il valore 0; questo significa che in  $\Sigma$  non esiste alcun elemento che colpisce  $N_i$
- Indichiamo con  $P(C_{\Sigma})$  la proiezione del contenuto di  $C_{\Sigma}$  su  $\Sigma$ , cioè l'insieme di elementi  $\eta \in \Sigma$  contenuti in  $C_{\Sigma}$

# Regola

- 1) Se  $P(C_{\Sigma})=\Sigma$  e in  $C_{\Sigma}$  non esiste alcuno 0, allora  $\Sigma$  è un MHS
- 2) Se  $P(C_{\Sigma})=\Sigma$  e in  $C_{\Sigma}$  esiste qualche 0, allora  $\Sigma$  è OK
- 3) Se  $P(C_{\Sigma}) \neq \Sigma$ , allora  $\Sigma$  è KO

#### Vettori rappresentativi

- Come si calcola il vettore rappresentativo  $C_{\Sigma}$ ?
- C<sub>∅</sub> è un vettore di tutti zeri
- Ciascun elemento  $\sigma_i$  di  $C_{\Sigma}$ , dove  $\Sigma \neq \emptyset$ ,  $\Sigma = \Sigma_1 \cup \Sigma_2$ ,  $\Sigma_1 \cap \Sigma_2 = \emptyset$ , max $(\Sigma_1)$  < min $(\Sigma_2)$ , si ottiene come segue  $(\sigma_{1i} \in \sigma_{2i} \text{ sono})$  l'elemento i-mo rispettivamente di  $C_{\Sigma_1} \in C_{\Sigma_2}$ :
  - Se ( $\sigma_{1i}$ = $\eta_1$ , con  $\eta_1$ ∈ $\Sigma_1$ , AND  $\sigma_{2i}$ = $\eta_2$ , con  $\eta_2$ ∈ $\Sigma_2$ ), allora  $\sigma_i$ =x
  - Se  $(\sigma_{1i}=x \text{ OR } \sigma_{2i}=x)$ , allora  $\sigma_{i}=x$
  - Se ( $\sigma_{1i}$ = $\eta_1$ , con  $\eta_1$ ∈ $\Sigma_1$ , AND  $\sigma_{2i}$ =0), allora  $\sigma_i$ =  $\eta_1$
  - Se  $(\sigma_{1i}=0 \text{ AND } \sigma_{2i}=\eta_2, \text{ con } \eta_2 \in \Sigma_2)$ , allora  $\sigma_i=\eta_2$
  - Se ( $\sigma_{1i}$ =0 AND  $\sigma_{2i}$ = 0), allora  $\sigma_{i}$ =0
- Si noti che dal punto precedente discende che  $C_{\{e\}}$ ,  $e \in M$ , è un vettore che uguaglia la colonna della matrice A relativa all'elemento  $e \in M$ , in cui a ciascun 1 è stato sostituito (idealmente) e

#### Modulo CHECK

- La specifica di CHECK è stata fornita pag. 16. Si riportano di seguito dei suggerimenti per realizzare una fra le tante implementazioni possibili di CHECK. Si può scegliere di adottare un'altra implementazione, purché funzionalmente equivalente
- Per quanto riguarda l'invocazione Check( $\Gamma$ ) entro MBASE, sappiamo che  $\Gamma = \Lambda \cup \{e\}$  dove, per costruzione,  $\Gamma \neq \emptyset$ ,  $\Lambda \cap \{e\} = \emptyset$ , max $\{\Lambda\} < e$ . Pertanto possiamo sfruttare quanto riportato nella pagina precedente per calcolare il vettore rappresentativo di  $\Gamma$  e sfruttare la regola di cui a pag. 20 per calcolare il valore di ritorno di Check( $\Gamma$ )

#### Modulo CHECK: alternative

- Il modo di realizzare CHECK illustrato nella pagina precedente lascia spazio a due alternative:
  - per ogni insieme inserito nella coda Q si salva non solo il contenuto dell'insieme stesso ma anche il corrispondente vettore rappresentativo (in tal caso, per il calcolo di  $C_{\Gamma}$ , si disporrebbe di  $C_{\Lambda}$ , quindi si tratterebbe di «comporre»  $C_{\Lambda}$  con  $C_{\{e\}}$  secondo le indicazioni della pagina precedente)
  - per ogni insieme inserito nella coda Q si salva unicamente il contenuto dell'insieme stesso (in tal caso si dovrebbe eseguire il calcolo di  $C_{\Gamma}$  da zero, componendo prima  $C_{\varnothing}$  con il vettore rappresentativo dell'insieme singoletto contenente il minimo di  $\Gamma$ , poi componendo il vettore risultante con il vettore rappresentativo corrispondente all'elemento successivo di  $\Gamma$ , e così via, in ordine lessicografico, fino all'esaurimento degli elementi di  $\Gamma$ )

# Modulo CHECK: prima alternativa

- La prima alternativa di cui alla pagina precedente prevede una maggiore occupazione di memoria e tempi di calcolo più rapidi rispetto alla seconda. Se si abbraccia la prima alternativa, è bene adottare strutture dati (sia per il contenuto di ciascun insieme sia per il vettore rappresentativo) che limitino l'occupazione di memoria centrale
- Abbracciando la prima alternativa, anche usando strutture dati «leggere», si rischia presto di eccedere la capacità della memoria centrale. Tale occupazione si potrebbe ridurre effettuando salvataggi su memoria di massa, appesantendo però i tempi di esecuzione, a causa degli elevati tempi di accesso alla memoria di massa (quindi vanificando il vantaggio sopra associato a questa alternativa ma consentendo di risolvere istanze di dimensioni elevate)

#### Modulo CHECK: seconda alternativa

- Se si abbraccia la seconda alternativa, è bene adottare meccanismi che contengano lo sforzo di calcolo
- Per contenere lo sforzo del calcolo del vettore rappresentativo di un insieme  $\Gamma$ , si potrebbe evitare di calcolare i vettori rappresentativi degli insiemi singoletti a partire dal vettore rappresentativo dell'insieme vuoto
- Per contenere lo sforzo del calcolo del vettore rappresentativo di un insieme  $\Gamma$  si può inoltre tenere conto del fatto che, se il valore di una cella del vettore rappresentativo di un sottoinsieme di  $\Gamma$  è x, allora tale cella assume il valore x anche per  $\Gamma$

#### Vettori rappresentativi dei singoletti

 Si noti che, quale che sia l'alternativa adottata, per tutta l'esecuzione di MBASE è necessario disporre dei vettori rappresentativi degli insiemi singoletti (o calcolare ogni volta il vettore rappresentativo di un singoletto accedendo alla relativa colonna della matrice)

#### Esempio di esecuzione di MBASE

	а	b	С	d	е	f		а	b	С	d	е	f
1	1	1	1				1	а	b	С			
2		1	1	1		1	2		b	С	d		f
3		1	1				3		b	С			
4	1		1			1	4	a		С			f
5	1	1	1	1			5	a	b	С	d		
6		1	1	1		1	6		b	С	d		f
7			1	1			7			С	d		

 Sia A la matrice a sinistra (il parametro d'ingresso dell'algoritmo MBASE). I vettori rappresentativi dei singoletti sono indicati nelle colonne della matrice di destra

# Prima esecuzione del ciclo for (riga 5 dello pseudocodice)

- a
- b
- c mhs
- d
- <u>•</u> e KO
- f non viene inserito
   perché non vi si potranno
   aggiungere ulteriori
   colonne per farlo
   diventare un MHS

	а	b	С	d	е	f
1	а	b	С			
2		b	С	d		f
3		b	С			
4	a		С			f
5	a	b	С	d		
6		b	С	d		f
7			С	d		

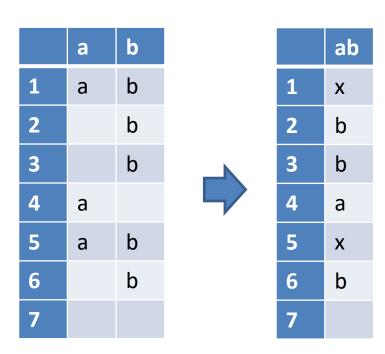
mhs trovati:
{c}

La colonna a sinistra indica gli insiemi (qui singoletto) via via generati (riga 6) e controllati (riga 7). Per brevità, in tutte le pagine dedicate agli esempi di esecuzione, gli identificatori degli elementi di un insieme (ciascuno è una lettera singola) vengono scritti uno di seguito all'altro, senza separatori, e l'insieme non viene racchiuso entro parentesi graffe. Gli insiemi non barrati costituiscono il contenuto ordinato di Q (a fianco degli insiemi cancellati, viene indicata la ragione per cui non stati inseriti in Q)

#### Seconda esecuzione del ciclo for



- b
- C
- ab

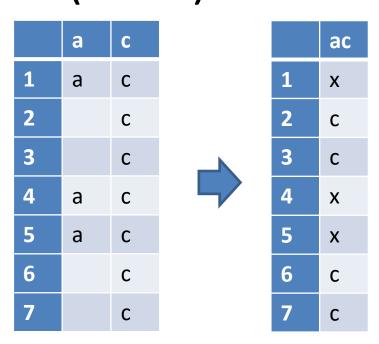


mhs trovati:
{c}

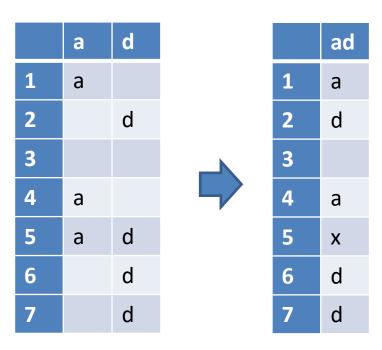
L'insieme su sfondo grigio è quello appena estratto da Q (quindi non appartiene più a Q), è cioè l'insieme  $\Lambda$  di cui si stanno generando i figli. La barra orizzontale separa gli insiemi singoletto contenuti ordinatamente in Q dagli insiemi di cardinalità 2 (cioè separa un livello dal successivo).

L'immagine illustra il calcolo del vettore rappresentativo di un insieme  $\Gamma$  di cardinalità 2 a partire dai vettori rappresentativi di due singoletti.

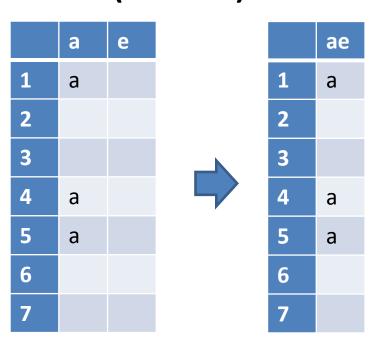
- a
- b
- C
- ab
- ac KO



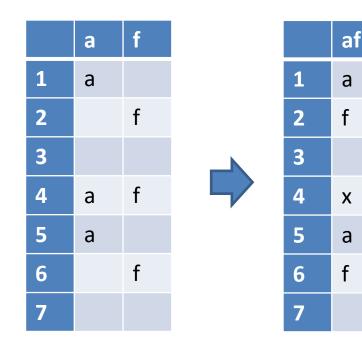
- a
- h
- d
- ab
- ad



- a
- b
- C
- ab
- ad
- ae KO

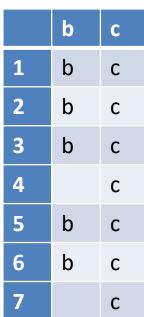


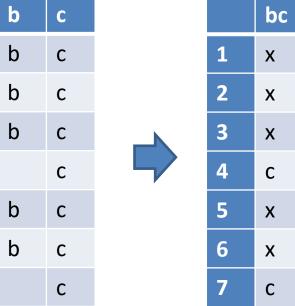
- a
- b
- 0
- ab
- ad
- af non viene
   inserito perché non
   vi si potranno
   aggiungere
   ulteriori colonne
   per farlo diventare
   un MHS



#### Terza esecuzione del ciclo for

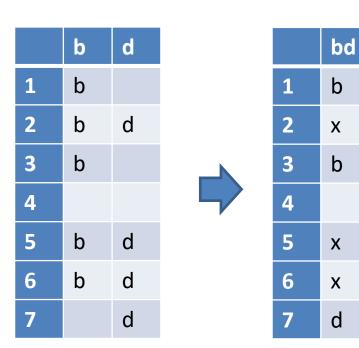
- ab
- ad
- bc KO





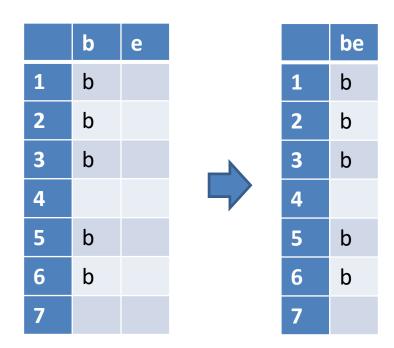
#### Terza esecuzione del ciclo for (cont.)

- b
- d
- ab
- ad
- bd



#### Terza esecuzione del ciclo for (cont.)

- b
- C
- ab
- ad
- bd
- be KO

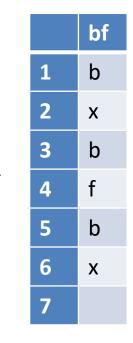


## Terza esecuzione del ciclo for (cont.)



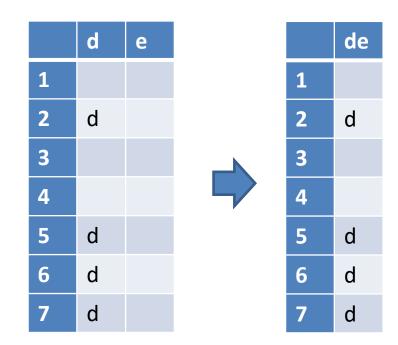
- d
- ab
- ad
- bd
- bf non viene
   inserito perché
   non vi si potranno
   aggiungere
   ulteriori colonne
   per farlo diventare
   un MHS

	b	f
1	b	
1	D	
2	b	f
3	b	
4		f
5	b	
6	b	f
7		



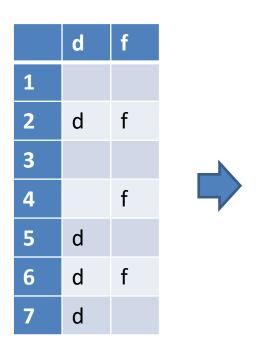
## Quarta esecuzione del ciclo for

- d
- ab
- ad
- bd
- de KO





- ab
- ad
- bd
- df non viene
   inserito perché
   non vi si potranno
   aggiungere
   ulteriori colonne
   per farlo
   diventare un MHS



mhs trovati:
{c}

df

Χ

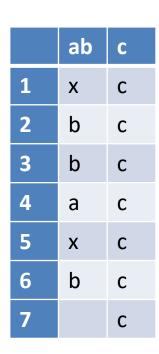
d

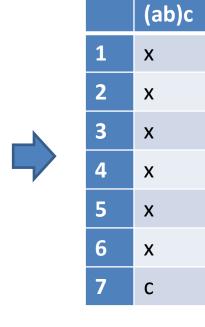
Χ

d

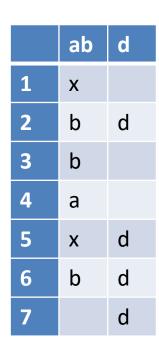
## Quinta esecuzione del ciclo for

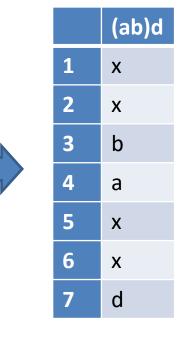
- ab
- ad
- bd
- (ab)c KC



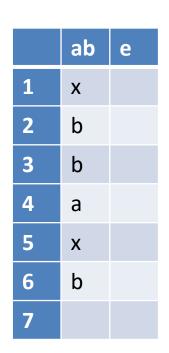


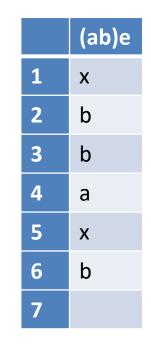
- ab
- ad
- bd
- (ab)d mhs





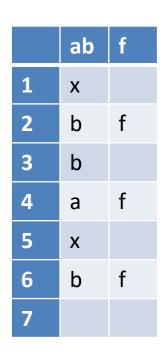
- ab
- ad
- bd
- (ab)e KO

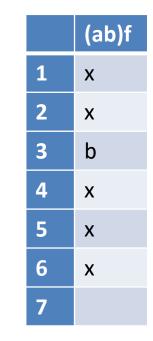






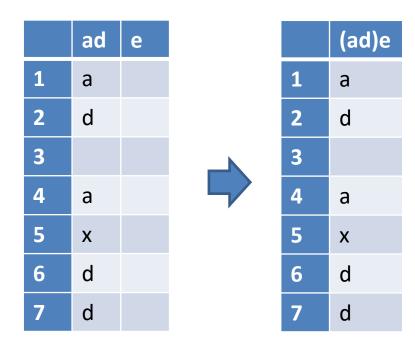
- ad
- bd
- (ab)f KO





### Sesta esecuzione del ciclo for

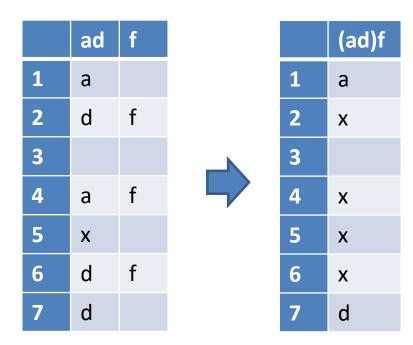
- ad
- bd
- (ad)e KO





bd

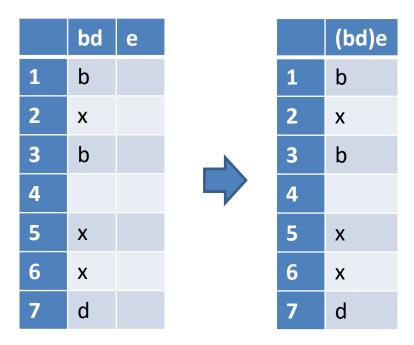
• (ad)f KC



## Settima (e ultima) esecuzione del ciclo for

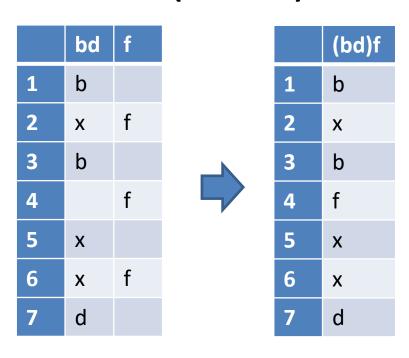
• bd

(bd)e KC



# Settima (e ultima) esecuzione del ciclo for (cont.)

- bd
- (bd)f mhs
- FINE



mhs trovati: {c}, {a, b, d}, {b, d, f}

Numero totale di iterazioni del ciclo **for** eseguite = 25

## Coda Q

- Livello di Q: elenco (ordinato lessicograficamente) di insiemi contenuti in Q aventi tutti la medesima cardinalità
- Durante l'esecuzione di MBASE (o di ciascuna delle sue due varianti) gli insiemi in Q sono disposti al più in due livelli, dove gli insiemi contenuti nel primo hanno una cardinalità di una unità inferiore rispetto al secondo
- Figlio di un insieme  $\Lambda$  (dove  $\Lambda \subset M$ ): insieme ottenuto (dall'algoritmo MBASE, riga 6) unendo a  $\Lambda$  un insieme singoletto {e}, dove  $e \in M$ , con {e}> $\Lambda$
- Fratello di un insieme  $\Gamma$ : figlio dello stesso padre di  $\Gamma$

## Compito

- Realizzare una applicazione software modulare che incarni l'algoritmo MBASE
- Per quanto riguarda i formati di I/O dell'applicazione si faccia riferimento alla sezione «Sperimentazione» di questo documento
- Si eviti la presenza di cloni nel codice dell'applicazione

### **PRIMA VARIANTE**

## Algoritmo MVariant1

- Per evitare di generare e analizzare (per poi scartare) superinsiemi di MHS già trovati, si può ricorrere a una prima variante, MVariant1, dell'algoritmo di base. Lo pseudocodice della variante è fornito a pag. 55 (esso differisce da quello di MBASE per l'aggiunta delle righe 6 e 13 e per la modifica della chiamata alla riga 10)
- Sia questa variante, sia la successiva assumono che ciascun insieme  $\Lambda$  che si trova nella coda Q disponga di una lista tabu, progressivamente aggiornata. Tale lista elenca dei sottoinsiemi di M, nessuno dei quali deve essere unito a  $\Lambda$  per generare un nuovo insieme (da analizzare)
- La creazione di un insieme  $\Gamma \leftarrow \Lambda \cup \{e\}$  avviene solo se  $\{e\}$  non appartiene alla lista tabu di  $\Lambda$  (riga 6)
- Contestualmente all'accodamento di un insieme  $\Gamma$  a Q avviene l'inizializzazione della sua lista tabu (nuova chiamata alla riga 10, su cui si ritornerà anche nella successiva sezione «Gestione delle liste tabu»)
- Ogni volta che l'algoritmo trova un MHS  $\Gamma$ , aggiorna (riga 13) le liste tabu degli insiemi in Q che potrebbero diventare superinsiemi di  $\Gamma$
- La lista tabu di Ø è vuota (riga 2)

### Modulo ISTABU

- Indicata con  $L_{\Lambda}$  la lista tabu dell'insieme  $\Lambda$ , l'invocazione IsTabu( $\Lambda$ ,e) ritorna il valore booleano true se  $\{e\} \in L_{\Lambda}$ , false altrimenti
- Attenzione: se  $L_{\Lambda}$  non contiene {e} e contiene un superinsieme proprio di {e}, allora IsTABU( $\Lambda$ ,e) ritorna il valore *false*

## Modulo Enqueue\_&\_Inherit

- Quando le due varianti dell'algoritmo MBASE accodano a Q un insieme Γ, figlio di Λ, tale operazione è accompagnata dall'inizializzazione della lista tabu di Γ, che deve avvenire in questo modo: Γ «eredita» la lista tabu di Λ, privata però di tutti gli insiemi ≤{max(Γ)} e sottraendo {max(Γ)} a ciascuno degli insiemi rimasti
- Ad esempio, assumendo che l'ordine lessicografico coincida con quello alfabetico, se  $\Lambda=\{a,c\}$  e  $L_{\Lambda}=\langle\{d,f\},\{g\},\{m,p,q\},\{m,r\},\{t\}\rangle$  e  $\Gamma=\{a,c,m\}$ , allora  $L_{\Gamma}=\langle\{p,q\},\{r\},\{t\}\rangle$

### Modulo Propagate Tabu

- La chiamata Propagate Tabu (Q,  $\Gamma$ ) invidua gli insiemi contenuti in Q le cui liste tabu sono da aggiornare al fine di evitare che essi evolvano in superinsiemi di  $\Gamma$  e aggiorna suddette liste
- Una implementazione naïf di PropagateTabu(Q,  $\Gamma$ ) si limita a passare in rassegna tutti gli insiemi  $\Lambda$  contenuti in Q, valutando se  $\Lambda$  possa diventare un superinsieme di  $\Gamma$  e, nel caso sia così, inserendo l'insieme  $\Gamma \backslash \Lambda$  in L $_{\Lambda}$
- Una implementazione più attenta di PROPAGATETABU è discussa nella sezione «Gestione delle liste tabu» di questo documento

## Prima variante: pseudocodice

```
Algorithm 2 Variant with tabu list (from MHSs only)

    procedure MVariant1(A)

                                                                                  ▶ A è la matrice
                                         ▷ prima di questo inserimento, la coda Q era vuota
        Enqueue(Q, \emptyset)
 2:
        while Q is not empty do
 3:
            \Lambda \leftarrow \text{Dequeue}(Q)
 4:
            for e \leftarrow succ(max(\Lambda)) to max(M) do
 5:
                if ! isTabu(\Lambda, e) then
 6:
                    \Gamma \leftarrow \Lambda \cup \{e\}
 7:
                    result \leftarrow Check(\Gamma)
 8:
                    if (result = OK) \land (e \neq max(M)) then
 9:
                        Enqueue_&_InheritTabu(Q, \Gamma)
10:
                    else if result = MHS then
11:
                        Output(\Gamma)
12:
                        Propagate_Tabu(Q,\Gamma)
13:
```

# Esempio di esecuzione di MVARIANT1: prima esecuzione del ciclo for

- a
- b
- e mhs: si
   aggiunge
   alle liste
   tabu di a e b

	а	b	С	d	е	f
1	а	b	С			
2		b	С	d		f
3		b	С			
4	a		С			f
5	a	b	С	d		
6		b	С	d		f
7			С	d		

mhs trovati:
{c}

Si riusa la stessa istanza di pag. 27

### Prima esecuzione del ciclo for

• a tabu: c

• b tabu: c

• d

<u>•</u> • KO

f non viene
 inserito perché
 non vi si potranno
 aggiungere
 ulteriori colonne
 per farlo
 diventare un MHS

	а	b	С	d	е	f
1	а	b	С			
2		b	С	d		f
3		b	С			
4	а		С			f
5	а	b	С	d		
6		b	С	d		f
7			С	d		

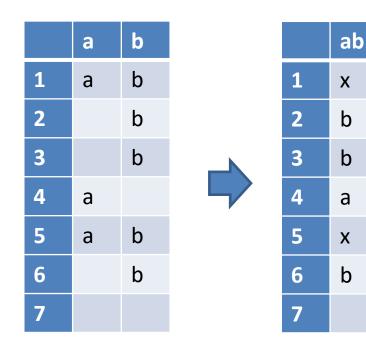
### Seconda esecuzione del ciclo for

• a tabu: c

• b tabu: c

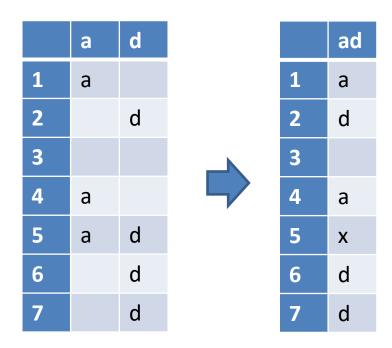
• d

• ab tabu: c



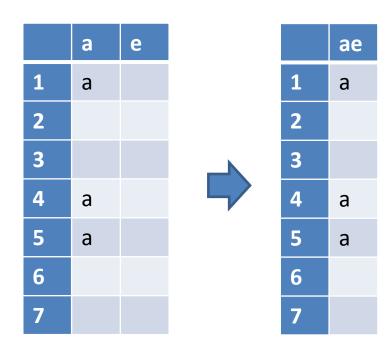
# Seconda esecuzione del ciclo for (cont.)

- a tabu: c
- b tabu: c
- d
- ab tabu: c
- ad



# Seconda esecuzione del ciclo for (cont.)

- a tabu: c
- b tabu: c
- C
- ab
- ad
- ae KO



# Seconda esecuzione del ciclo for (cont.)

• a tabu: c

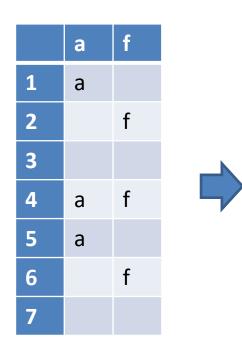
• b tabu: c

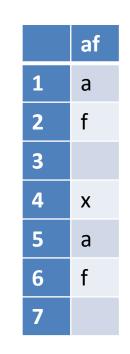
• d

• ab tabu: c

ad

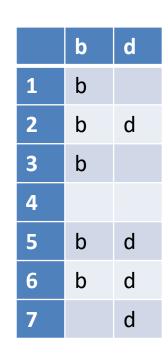
af non viene
 inserito perché
 non vi si potranno
 aggiungere
 ulteriori colonne
 per farlo diventare
 un MHS

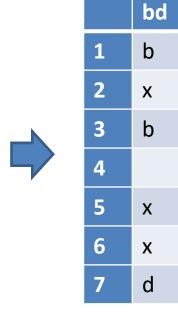




#### Terza esecuzione del ciclo for

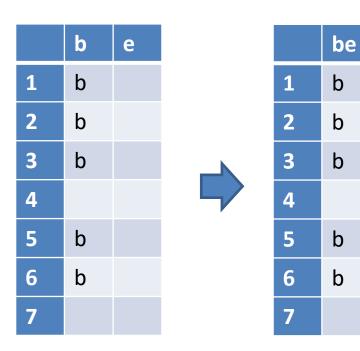
- b tabu: c
- d
- ab tabu: c
- ad
- bd





### Terza esecuzione del ciclo for

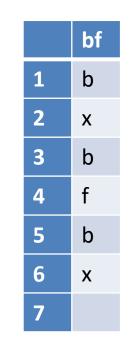
- b tabu: c
- C
- ab tabu: c
- ad
- bd
- be KO



## Terza esecuzione del ciclo for (cont.)

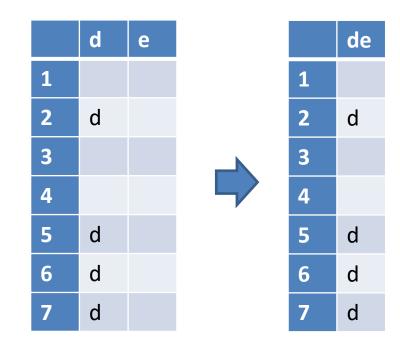
- b tabu: c
- C
- ab tabu: c
- ad
- bd
- bf non viene
   inserito perché
   non vi si potranno
   aggiungere
   ulteriori colonne
   per farlo diventare
   un MHS

	b	f
1	b	
2	b	f
3	b	
4		f
5	b	
6	b	f
7		



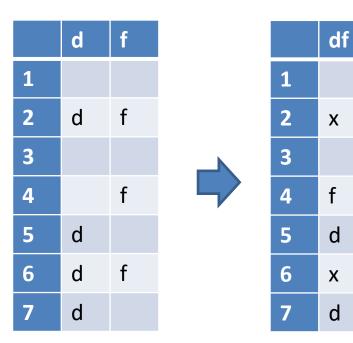
## Quarta esecuzione del ciclo for

- d
- ab tabu: c
- ad
- bd
- de KO



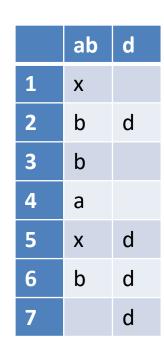


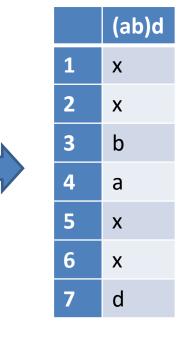
- ab tabu: c
- ad
- bd
- df-non viene
   inserito perché
   non vi si potranno
   aggiungere
   ulteriori colonne
   per farlo
   diventare un MHS



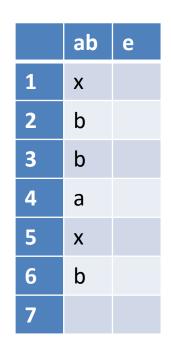
## Quinta esecuzione del ciclo for

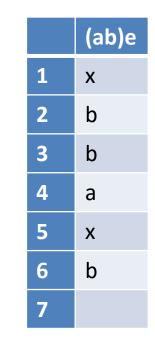
- ab tabu: c
- ad
- bd
- abd mhs



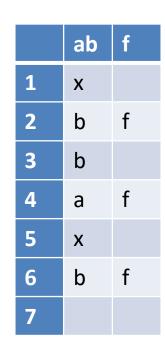


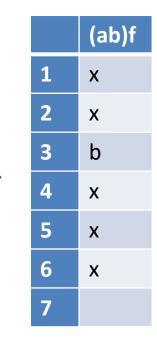
- ab tabu: c
- ad
- bd
- abe KO





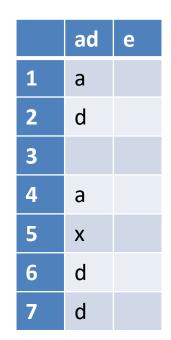
- ab tabu: c
- ad
- bd
- abf KO

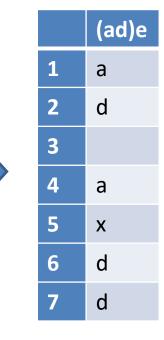




### Sesta esecuzione del ciclo for

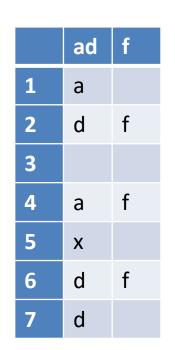
- ad
- bd
- ade KO







- bd
- adf KO

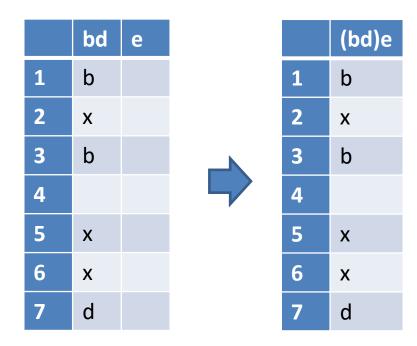




## Settima (e ultima) esecuzione del ciclo for

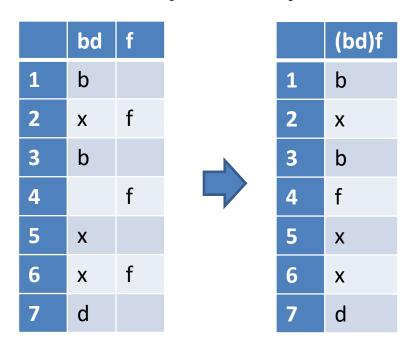
• bd

bde KO



# Settima (e ultima) esecuzione del ciclo for (cont.)

- bd
- bdf mhs
- FINE



mhs trovati: {c}, {a, b, d}, {b, d, f}

Numero totale di iterazioni del ciclo **for** eseguite = 22

### **SECONDA VARIANTE**

# Algoritmo MVariant2

- La seconda variante differisce dalla prima perché riduce ulteriormente il numero di insiemi generati e analizzati, infatti essa evita di generare insiemi che siano superinsiemi non solo di un MHS ma anche di un insieme KO. Pertanto, ogni volta che viene generato un insieme  $\Gamma$  che sia MHS o KO, si aggiornano le liste tabu di tutti gli insiemi che si trovano in Q e che potrebbero diventare superinsiemi di  $\Gamma$  (le liste tabu saranno in generale più lunghe di quelle che si ottengono con MVARIANT1)
- Nello pseudocodice della pagina successiva, le differenze di MVARIANT2 rispetto a MVARIANT1 sono confinate nelle righe 11-14. Si noti che tutti i moduli invocati sono gli stessi di MVARIANT1 (e non necessitano di alcuna modifica)

### Algoritmo MVariant2: pseudocodice

#### Algorithm 3 Variant with tabu list (complete)

```
1: procedure MVariant2(A)

▷ A è la matrice

        Enqueue(Q, \emptyset)
                                         prima di questo inserimento, la coda Q era vuota
        while Q is not empty do
 3:
            \Lambda \leftarrow \text{Dequeue}(Q)
 4:
            for e \leftarrow succ(max(\Lambda)) to max(M) do
 5:
                if ! isTabu(\Lambda, e) then
 6:
                    \Gamma \leftarrow \Lambda \cup \{e\}
 7:
                    result \leftarrow Check(\Gamma)
 8:
                    if (result = OK) \land (e \neq max(M)) then
9:
                        Enqueue_&_Inherit(Q, \Gamma)
10:
                    else if (result = MHS) \lor (result = KO) then
11:
                        Propagate_Tabu(Q,\Gamma)
12:
                        if result = MHS then
13:
                            Output(\Gamma)
14:
```

# Implementazione

 Se si dispone di una applicazione che codifichi l'algoritmo MVARIANT1, estendere la stessa in modo che implementi l'algoritmo MVARIANT2 non dovrebbe richiedere altro che la «modifica» del codice di MVARIANT1 (mentre i moduli invocati non dovrebbero subire alcun cambiamento)

### **GESTIONE DELLE LISTE TABU**

### Lista tabu

 Dal punto di vista della correttezza, la lista tabu associata a ciascun insieme contenuto in Q può essere gestita nella maniera più varia, purché su di essa possano essere compiute le operazioni previste dagli algoritmi descritti in questo documento. Di seguito si danno delle indicazioni per una fra le tante realizzazioni possibili

# Algoritmo Propagate Tabu

- La chiamata Propagate Tabu (Q,  $\Gamma$ ) invidua gli insiemi contenuti in Q le cui liste tabu sono da aggiornare e delega l'aggiornamento vero e proprio a Insert Tabu
- Dato il parametro  $\Gamma$  di Propagate Tabu, le liste tabu che devono essere aggiornate sono solo quelle di (una selezione di) insiemi che già sono presenti in Q e che si trovano nel livello contraddistinto dal valore di cardinalità  $|\Gamma|$
- Lo pseudocodice dell'algoritmo Propagate Tabu è fornito nella pagina successiva

# Algoritmo Propagate Tabu: pseudocodice

```
Algorithm 4 Finding the tabu lists to be updated

    procedure Propagate_Tabu(Q, Γ)

        for all s siblings of \Gamma in Q do
                                                                       'siblings' sta per 'fratelli'
 2:
            Insert_Tabu(s, \{max(\Gamma)\}\)
 3:
        for all S_L sublayers preceding the siblings of \Gamma do
                                                                                ▷ 'sublayers' sta per
 4:
    'sottolivelli'
            for all \Phi in S_L starting from the first one do
 5:
                \Delta \leftarrow \Gamma \setminus \Phi
 6:
                if min(\Delta) > max(\Phi) then
 7:
                    Insert_Tabu(\Phi, \Delta)
 8:
                                                                 \,\triangleright\mid\Delta\midindica la cardinalità di\Delta
                    if |\Delta| = 1 then break
 9:
                else break ▷ ciascun break determina l'uscita dal ciclo for più annidato
10:
```

# Algoritmo InsertTabu

- L'algoritmo INSERTTABU assume che la lista tabu  $L_{\Lambda}$  associata a ciascun insieme  $\Lambda$  presente in Q sia totalmente ordinata lessicograficamente e che nessun insieme nell'elenco sia contenuto nell'insieme successivo secondo tale ordine (quindi se  $S_1$ ,  $S_2 \in L_{\Lambda}$ ,  $S_1 < S_2$ , allora  $S_1 \not\subset S_2$ ). Ogni operazione compiuta sulla lista non deve infrangere questo invariante
- Inoltre ciascuna lista può essere vista come suddivisa in sottoliste, deve ogni sottolista contiene insiemi che condividono lo stesso elemento minimo
- Si noti che se una sottolista contiene un singoletto, allora, in virtù dell'invariante, esso è l'unico insieme della sottolista

# Algoritmo InsertTabu

#### Algorithm 5 Updating a tabu list

```
1: procedure Insert_Tabu(\Phi, \Delta)
           \alpha \leftarrow \min\{\Delta\}
 2:
           if \exists \Upsilon in L_{\Phi,\alpha} s.t. \Upsilon \geq \Delta then \triangleright L_{\Phi,\alpha} è la sottolista (della lista tabu L_{\Phi})
 3:
     contenente (in ordine lessicografico) gli insiemi il cui elemento minimo è \alpha
                 if (\Delta \neq \Upsilon) \wedge (\Upsilon' \not\subseteq \Delta) then \triangleright \Upsilon' is the element preceding \Upsilon in L_{\Phi,\alpha} (if any)
 4:
                      Replace \Upsilon with \Delta
 5:
                      if \Delta \subset \Upsilon then
 6:
                            for all \Psi in L_{\Phi,\alpha}, \Psi > \Upsilon, \Delta \subset \Psi do
 7:
                                  Remove \Psi from L_{\Phi,\alpha}
 8:
           else
 9:
                 Insert \Delta at the end of L_{\Phi,\alpha}
10:
```

# Modulo Enqueue\_&\_Inherit

 Attenzione: se per ciascuna lista tabu vale l'invariante prima enunciato (la lista è ordinata lessicograficamente e nessun insieme nell'elenco è contenuto nell'insieme successivo secondo tale ordine), allora l'algoritmo ENQUEUE & INHERIT(Q,  $\Gamma$ ), per modificare la lista «ereditata» dal padre  $\Lambda$  di  $\Gamma$ , deve cercare il primo insieme  $\Psi$ >{max( $\Gamma$ )}, rimuovere tutti gli elementi che precedono lo stesso e sottrarre  $\{\max(\Gamma)\}\$  dall'insieme  $\Psi$  e da quelli successivi. Al primo insieme incontrato, a partire da  $\Psi$ , che non contiene l'elemento max $(\Gamma)$ , l'operazione si conclude (perché sicuramente anche tutti gli elementi successivi non contengono  $max(\Gamma)$ )

### **PRE-ELABORAZIONE**

### Pre-elaborazione

- Esistono delle operazioni di preelaborazione che possono ridurre l'istanza (N, M) del problema di calcolo dei MHS assegnata a una nuova istanza (N', M'), con  $|N'| \le |N|$  e  $|M'| \le |M|$
- Indicato con MHS(R, S) l'insieme delle soluzioni relativo a una generica istanza (R, S), data un'operazione di preelaborazione pre(N, M) = (N', M'), si possono verificare due casi
  - MHS(N', M') = MHS(N, M), pertanto non è necessario compiere alcuna operazione di post-elaborazione, oppure
  - MHS(N', M') ≠ MHS(N, M), pertanto è necessario compiere un'operazione di post-elaborazione post (dipendente dalla specifica operazione di pre-elaborazione effettuata) tale che post(MHS(N', M')) = MHS(N, M)
- Qui ci occuperemo solo di operazioni di pre-elaborazione che cadono nel primo caso

# Operazioni di preelaborazione che non richiedono alcuna post-elaborazione

- Operazione di preelaborazione togli\_righe: rimozione di ciascun insieme N<sub>i</sub> della collezione per il quale esista già nella collezione un insieme N<sub>j</sub> (con j ≠ i) tale che N<sub>i</sub> ⊆ N<sub>i</sub>
- Operazione di preelaborazione togli\_colonne: rimozione di ciascun elemento del dominio che non interseca alcun insieme della collezione

# Implementazione

- Attenzione: le operazioni di cui alla pagina precedente devono essere eseguite (entrambe prima di invocare l'algoritmo di calcolo dei MHS sulla nuova istanza (N', M')) secondo l'ordine indicato, cioè togli\_righe seguita da togli\_colonne, perché la soppressione di alcune righe della matrice d'ingresso determina una nuova collezione N' e per quest'ultima il numero di colonne vuote (cioè contenenti solo valori 0) può essere superiore rispetto a quello della collezione N
- Sebbene il valore di |M'| possa essere inferiore rispetto a |M|, si abbia cura di rappresentare i MHS in uscita mediante una matrice a |M| colonne (secondo lo stesso ordine delle colonne della matrice in ingresso originale)

### **SPERIMENTAZIONE**

### Benchmark

- Sono resi disponibili due benchmark, ciascuno contenente file di testo (rispettivamente 1371 e 1400) di tipo matrix (attenzione: numerosi file sono condivisi fra i due)
- Ogni file rappresenta una matrice d'ingresso (le cui dimensioni possono arrivare fino a qualche migliaio di colonne)
- Ogni file .matrix è univocamente individuato dal suo nome (ad es. 74L85.026): la descrizione delle prove sperimentali condotte acquisendo in ingresso tale file devono riportare questo nome

### Formato interno dei file .matrix

commenti

righe della matrice; qui |N| = 2

<u>indice intero</u> assegnato a ciascun <u>elemento di M</u> (fra parentesi); qui | M | = 33

sepàratori delle righe della matrice

### **RICHIESTE**

# Gruppi di lavoro

- Ogni gruppo, costituito da <u>due</u> studenti, deve realizzare una applicazione software che codifichi l'algoritmo principale e
  - a) specializzarla nelle due varianti proposte (la gestione delle liste tabu non deve necessariamente seguire i suggerimenti dati nell'omonima sezione), oppure
  - b) dotarla delle funzionalità di pre-elaborazione proposte (dopo avere descritto gli algoritmi relativi a tali funzionalità in pseudocodice).

Inoltre, ogni gruppo deve redigere una relazione che illustri il lavoro svolto, le scelte implementative compiute e la sperimentazione condotta. Tale sperimentazione deve avvenire su una selezione (a discrezione del gruppo) dei file .matrix forniti.

# Alternativa a)

- Se il gruppo di lavoro sceglie l'alternativa a) della pagina precedente, il primo fine della sperimentazione è quello di affrontare ciascuna istanza di problema (file .matrix) considerata con i tre algoritmi realizzati (MBASE, MVARIANT1 e MVARIANT2) per confrontare (possibilmente automaticamente) i risultati ottenuti: ogni differenza denuncia la presenza di difetti nella specifica (fornita dal docente) e/o nella implementazione
- Il secondo fine della sperimentazione è quello di registrare le prestazioni (spaziali e temporali) delle prove condotte
- È inoltre opportuno registrare le dimensioni di ciascuna istanza di problema affrontata

# Alternativa b)

- Se il gruppo di lavoro sceglie l'alternativa b) di pag. 93, il primo fine della sperimentazione è quello di affrontare ciascuna istanza di problema (file .matrix) considerata con due invocazioni dell'algoritmo MBASE, la prima non preceduta da alcuna pre-elaborazione, la seconda preceduta da pre-elaborazione, per confrontare (possibilmente automaticamente) i risultati ottenuti: ogni differenza denuncia la presenza di difetti nella specifica (fornita dal docente) e/o nella implementazione
- Il secondo fine della sperimentazione è quello di registrare le prestazioni (spaziali e temporali) delle prove condotte; in particolare, si devono registrare separatamente i tempi di esecuzione di MBASE senza preelaborazione, quelli di pre-elaborazione e quelli della esecuzione successiva di MBASE
- È inoltre opportuno registrare le dimensioni di ciascuna istanza di problema affrontata (cioè i valori di |N|, |M|, |N'| e |M'|); si possono eventualmente indicare anche i numeri d'ordine delle colonne vuote 'soppresse'

### Lavoro

• È positivo che l'applicazione sviluppata registri informazioni aggiuntive che siano di interesse per la valutazione sperimentale, quali il numero di MHS calcolati per ciascuna istanza e la cardinalità minima e massima di tali MHS

### Lavoro e relazione

- Particolare attenzione deve essere dedicata alla scelta di strutture dati volte a estendere le dimensioni delle matrici che possono essere elaborate nonché di altri accorgimenti aventi lo stesso obiettivo. La relazione deve documentare tali scelte e accorgimenti
- Sono naturalmente apprezzati gli sforzi tesi a ridurre il costo temporale della computazione, che devono anch'essi essere documentati
- La relazione deve contenere ogni indicazione ritenuta utile al fine di consentire l'utilizzo dei programmi realizzati e la conduzione di ulteriori sperimentazioni
- La relazione deve evidenziare tutte le limitazioni riscontrate nelle prove di esecuzione effettuate

## Requisiti funzionali

- L'applicazione software sviluppata deve:
  - per ogni istanza in ingresso (file .matrix), generare in uscita i MHS relativi nonché le informazioni riassuntive di cui si è già parlato
  - consentire all'utente di interrompere il calcolo prima che esso sia concluso (o eventualmente di fissare una durata massima per l'elaborazione), fornendo in uscita i risultati parziali già calcolati, accompagnati anch'essi dalle informazioni citate al punto precedente, esplicitando però che il calcolo non è stato completato

# Requisiti non funzionali

- Non è richiesta la realizzazione di GUI: l'elaborazione può felicemente essere batch (I/O solo da/per file)
- Non è imposto un linguaggio di programmazione, né un ambiente di sviluppo né un ambiente di destinazione

# Consegna del materiale

- Ai fini del superamento della prova orale è necessario consegnare, entro i tempi indicati nelle note relative agli appelli, una cartella elettronica compressa contenente codice sorgente, eventuale codice eseguibile e relazione (in formato sia "sorgente", sia pfd)
- La consegna deve avvenire inviando via email il link a tale cartella, link creato usando un sistema di condivisione (ad es. dropbox), oppure attraverso la piattaforma Moodle