Algoritmi e Strutture Dati

Elaborato a.a. 2021-2022

Prof.ssa Marina Zanella

Problema: exact cover

- Input: una collezione finita N di insiemi finiti (distinti) dove gli elementi di ogni insieme appartengono al dominio M (si assume che M sia l'unione di tutti gli insiemi della collezione N)
- Output: tutte le partizioni (o coperture esatte) di M dove ciascuna parte è un insieme della collezione N

Problema: exact cover

- Partizione di M (vincolata da N): (sotto)insieme della collezione N costituito da insiemi tutti reciprocamente disgiunti e tali che la loro unione è M

Calcolo delle partizioni: esempio

N = { {b3},
{a1,a2,b4},
{a2,b3,b4,a5},
{a5},
{a1,a2,b3},
{b4,a5} }

M = {a1,a2,b3,b4,a5}

Partizioni = { {{b3},{a1,a2,b4},{a5}},
{{a1,a2,b3},{b4,a5}} }

Problema di decisione corrispondente

- Il problema di decisione teso a stabilire se esista una partizione di M vincolata da N è uno dei 21 problemi classificati come NPC da Karp nel celebre articolo del 1972

REDUCIBILITY AMONG COMBINATORIAL PROBLEMS

Richard M. Karp

University of California at Berkeley

Proprietà che, se vera, porta ad accettare l'istanza in ingresso al problema di decisione

14. EXACT COVER
INPUT: family $\{S_j\}$ of subsets of a set $\{u_i, i = 1,2,\ldots,t\}$
PROPERTY: There is a subfamily $\{T_h\} \subseteq \{S_j\}$ such that the sets T_h are disjoint and $\cup T_h = \cup S_j = \{u_i, i = 1,2,\ldots,t\}$.

Rilevanza di exact cover

- Sudoku (9x9), Hexadoku (16x16), Kanoodle (5x11) e Pentomino tiling possono essere ridotti a exact cover (si veda ad es. http://www.ams.org/publicoutreach/feature-column/fcarc-kanoodle)
- Esiste un algoritmo risolvente proposto da Knuth che calcola tutte le partizioni (l'implementazione di tale algoritmo può convenientemente fare uso di strutture dati denominate *dancing links*)
- L'approccio adottato in questo elaborato è diverso rispetto a quello di Knuth, pur condividendo con esso lo scopo di calcolare tutte le partizioni

Input: matrice (ideale)

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

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

Esempio

N = { {b3},	1
{a1,a2,b4},	2
{a2,b3,b4,a5}	3
{a5}	4
{a1,a2,b3}	5
{b4,a5} }	6

Input: matrice (cont.)

- I dati d'ingresso del problema possono essere rappresentati come una matrice $A_{|N|,|M|}$, dove il valore del componente $a_{i,j}$ della matrice è 1 se l'elemento (di M) di indice j appartiene all'insieme (in N) di indice i, 0 altrimenti
- Precisazione: un insieme vuoto può comparire in ogni partizione, tuttavia è preferibile ometterlo. Pertanto, se un insieme delle collezione N fosse accidentalmente vuoto, esso non dovrà comparire in nessuna soluzione del problema exact cover

Esempio: matrice d'ingresso A

- {b3}
- {a1,a2,b4}
- {a2,b3,b4,a5}
- {a5}
- {a1,a2,b3}
- {b4,a5}

{a1,a2, b3,b4,a5}

	1	2	3	4	5
1	0	0	1	0	0
2	1	1	0	1	0
3	0	1	1	1	1
4	0	0	0	0	1
5	1	1	1	0	0
6	0	0	0	1	1

D'ora innanzi si farà riferimento all'insieme i-mo ($1 \leq i \leq |N|$) della collezione N come A[i]

Output

- Insieme (denominato COV) di tutte le partizioni, dove ciascuna partizione è rappresentata da un insieme di identificatori, uno per ciascun insieme appartenente a N che fa parte della partizione

Esempio

N = { {b3}, 1
{a1,a2,b4}, 2
{a2,b3,b4,a5} 3
{a5} 4
{a1,a2,b3} 5
{b4,a5} } 6

Partizioni = { {{b3},{a1,a2,b4},{a5}},
{{a1,a2,b3},{b4,a5}} } $\rightarrow$ COV = {{1,2,4}, {5,6}}

Matrice di compatibilità

- L'approccio proposto fa ampio uso del concetto di matrice di compatibilità, che è relativa agli insiemi della collezione distinti da $\varnothing$ e M (si veda anche la nota a pag. 29)
- Si tratta di una matrice simmetrica, indicata con B, in cui $b_{i,j}$ assume il valore 1 se valgono le tre seguenti condizioni
  - $i \neq j$
  - $A[i] \cap A[j] = \varnothing$
  - $A[i] \cup A[j] \neq M$

  0 altrimenti

Esempio: matrice di compatibilità B

- Della matrice B basta riempire le celle $b_{i,j}$ in cui j > i, come in figura (le celle vuote in realtà contengono il valore 0)
- A[i] e A[j] (con j > i) si dicono insiemi compatibili se $b_{i,j} = 1$ (nel senso che ad essi si possono aggregare ulteriori insiemi al fine di formare delle partizioni)

	1	2	3	4	5
1	0	0	1	0	0
2	1	1	0	1	0
3	0	1	1	1	1
4	0	0	0	0	1
5	1	1	1	0	0
6	0	0	0	1	1

	1	2	3	4	5	6
1		1		1		1
2				1		
3						
4					1	
5						
6						

ALGORITMO: FUNZIONAMENTO

Principi

- L'algoritmo risolvente proposto (in due versioni) sfrutta l'ordine degli insiemi entro la collezione N (detto ordine lessicografico)
- Esso produce incrementalmente la matrice di compatibilità (nella matrice suddetta non si introduce alcuna colonna che sia relativa a un insieme vuoto o a un insieme coincidente con M ma queste colonne si possono anche aggiungere, come rilevato nella nota di pag. 29)
- Mentre crea la matrice di compatibilità, analizza via via gli aggregati di insiemi della collezione N la compatibilità reciproca dei quali sia già stata appurata

Principi (cont.)

- L'operato dell'algoritmo può essere descritto in termini di esplorazione di alberi, dove un nodo d'albero rappresenta un aggregato di uno o più insiemi della collezione
- Nessun aggregato contiene come elemento un insieme vuoto
- Nessun aggregato contiene come elemento un insieme coincidente con M
- Ciascun albero ha per radice un (singolo) insieme (non vuoto né coincidente con M) distinto della collezione e contiene tutti e soli gli aggregati costituiti da tale insieme e da insiemi (non vuoti né coincidenti con M) che lo precedono secondo l'ordine lessicografico degli insiemi

Principi (cont.)

- A ogni nodo d'albero è associato l'insieme degli identificatori degli insiemi che appartengono all'aggregato a cui il nodo si riferisce
- A ciascuna radice è associato un insieme (di identificatori) singoletto {i}, $1 \leq i \leq |N|$, dove A[i] non è l'insieme vuoto né coincide con M: l'albero avente tale radice si dice «radicato in i»
- Gli alberi vengono visitati per valori crescenti degli identificatori delle loro radici (secondo l'ordine lessicografico)

Visita degli aggregati di due insiemi

- Durante la visita dell'albero radicato in i vengono inizializzate le caselle della matrice di compatibilità della colonna relativa a i (prima la casella B[1,i], poi B[2,i], ecc. fino ad assegnare un valore alla casella B[i – 1,i]): questo implica la visita di tutti i nodi corrispondenti ad aggregati di cardinalità 2

Visita degli aggregati di tre o più insiemi

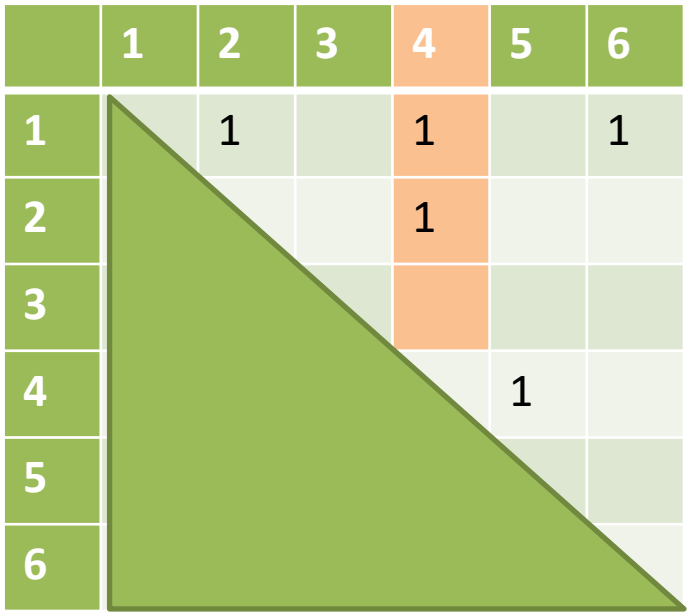
- Ciascun nodo corrispondente a un aggregato di cardinalità > 2 viene invece visitato solo se gli insiemi che lo costituiscono sono compatibili a due a due, secondo quanto riportato nella matrice B
- Ad es., data la matrice B a lato, il nodo 543 non verrà visitato perché la casella B[3,5] contiene uno zero (così come B[3,4])

	1	2	3	4	5	6
1		1		1		1
2				1		
3						
4					1	
5						
6						

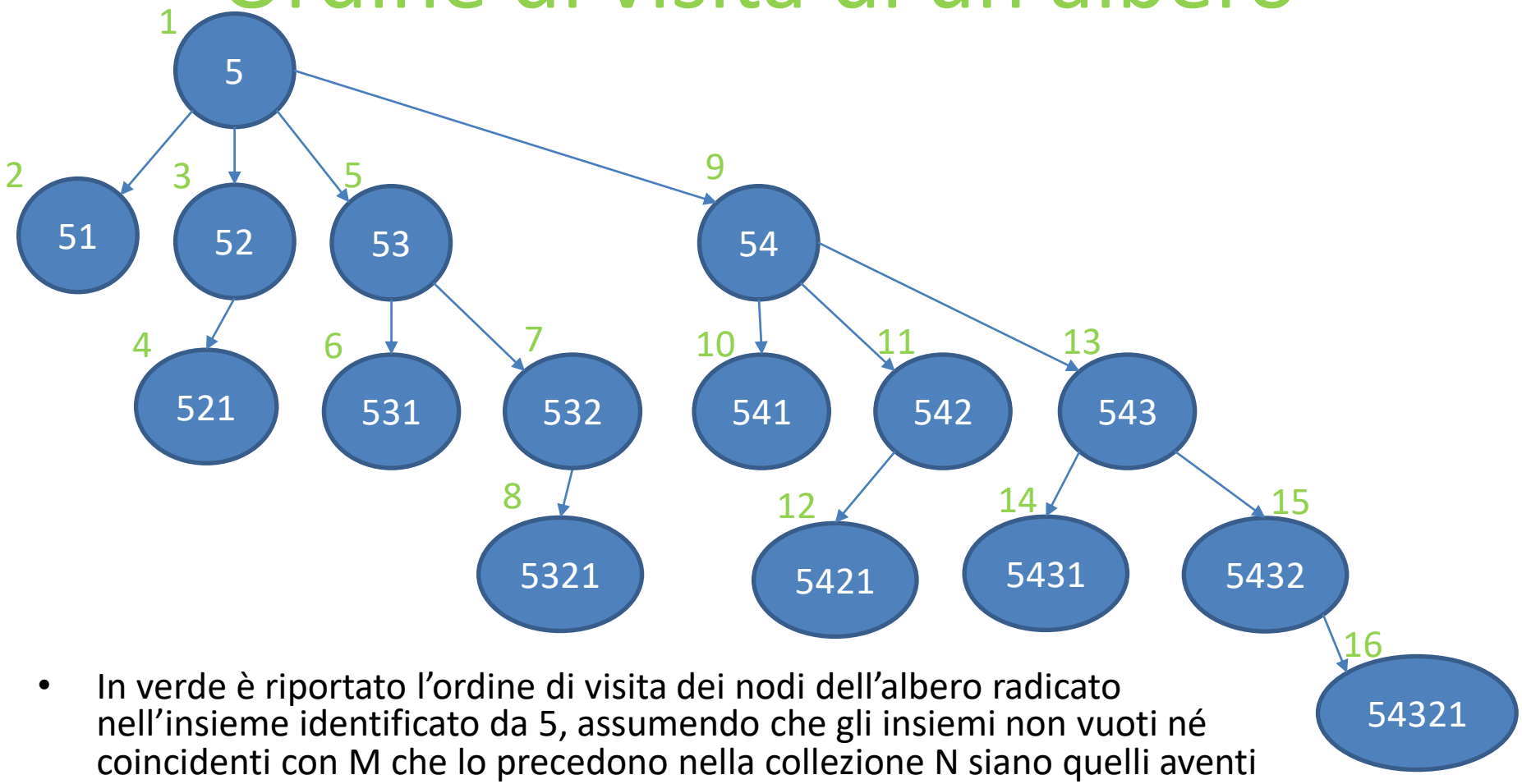
Sotto-colonne

- Nello pseudocodice, con B[1..i, j] si indica la sotto-collezione di N costituita dagli insiemi compatibili con l'insieme A[j] i cui indici vanno da 1 a i (j > i)

- Ad es., B[1..3, 4] si riferisce alle celle evidenziate in rosa e rappresenta l'insieme {1,2} degli indici degli insiemi della collezione N compresi fra 1 e 3 che, secondo la matrice B, sono compatibili con l'insieme A[4]
- Attenzione: B[1..0, h] indica un insieme vuoto. Questa situazione si verifica alla linea 19 dello pseudocodice di pag. 28 e alla linea 20 dello pseudocodice di pag. 70 quando j = 1, nonché alla linea 7 dello pseudocodice sia di pag. 29 sia di pag. 71 quando k = 1

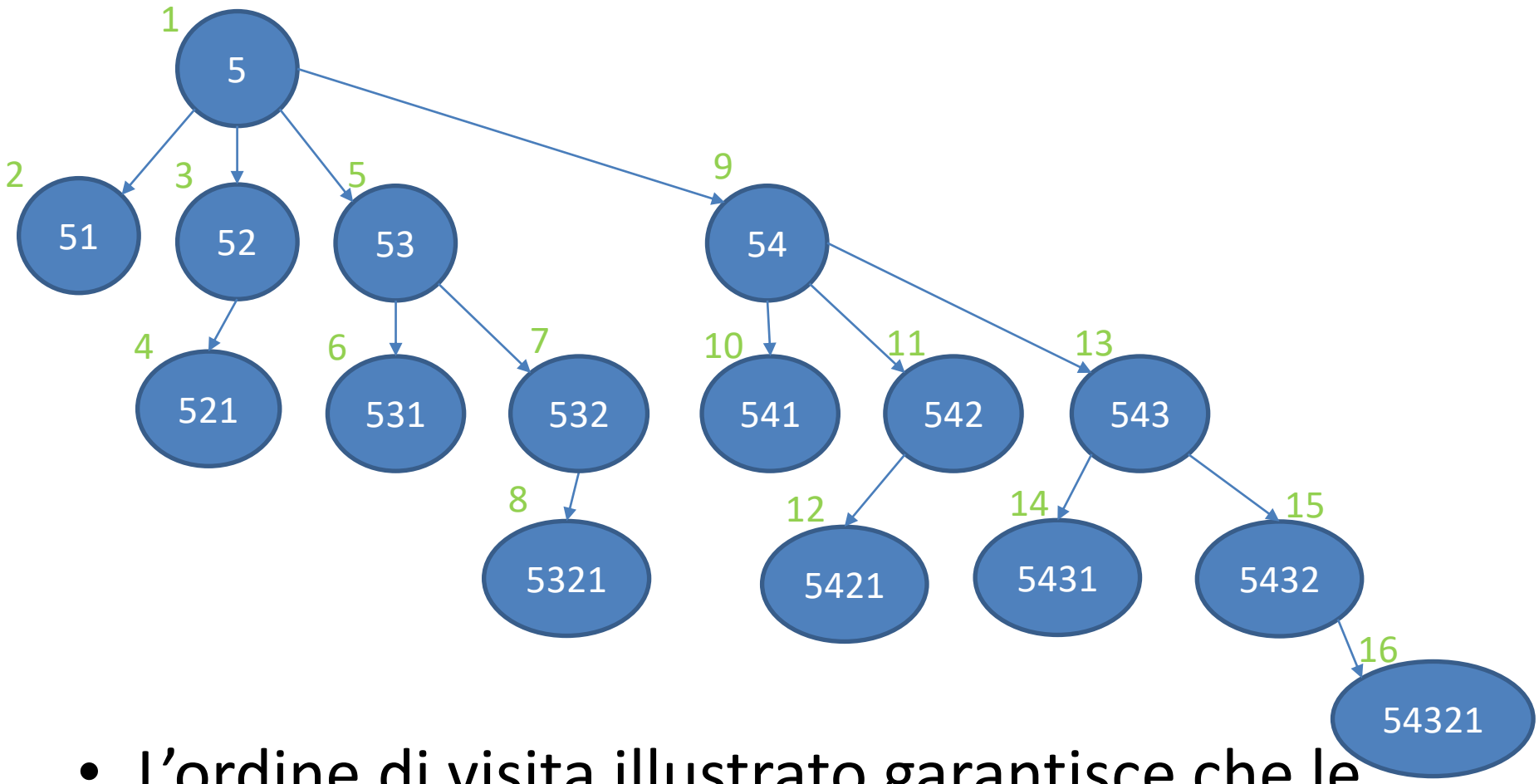


Ordine di visita di un albero



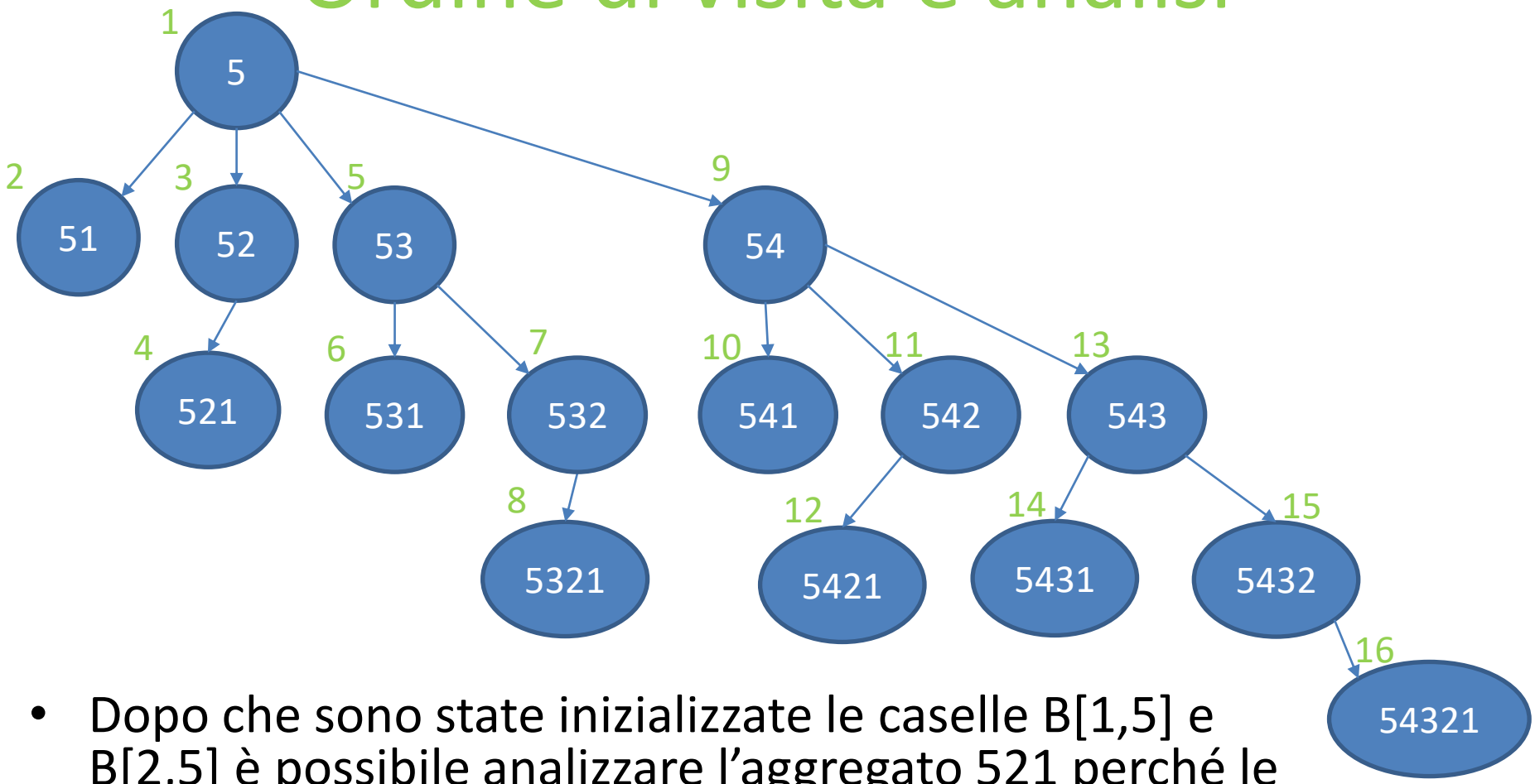
- In verde è riportato l'ordine di visita dei nodi dell'albero radicato nell'insieme identificato da 5, assumendo che gli insiemi non vuoti né coincidenti con M che lo precedono nella collezione N siano quelli aventi gli identificatori da 1 a 4 e che l'ordine entro la collezione rispecchi l'ordine dei valori interi
- Per brevità un aggregato è indicato dalla sequenza degli identificatori (in questo esempio un identificatore è una singola cifra) degli insiemi che lo costituiscono

Ordine di visita e creazione di B



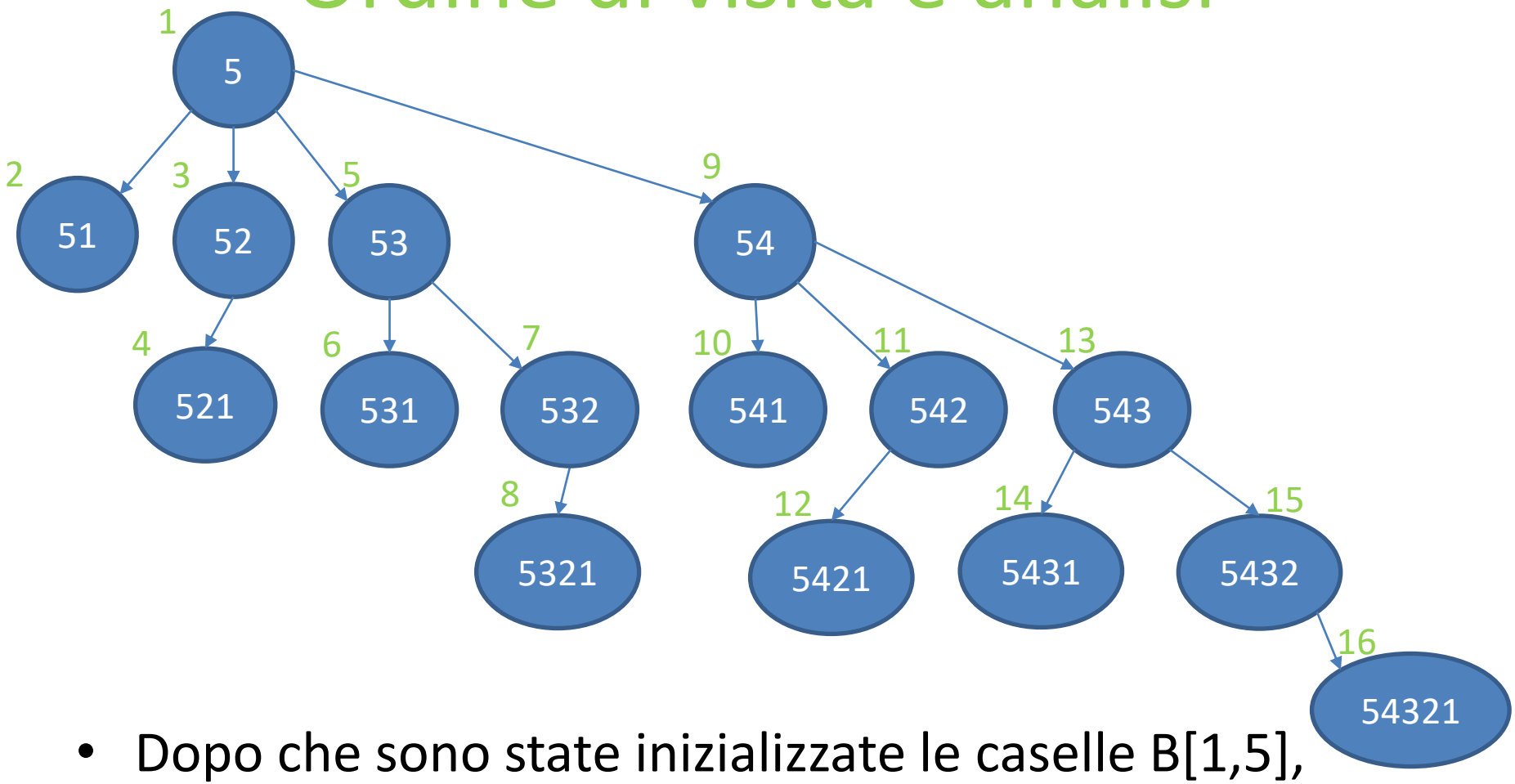
- L'ordine di visita illustrato garantisce che le caselle B[1,5], B[2,5], B[3,5] e B[4,5] vengano inizializzate nell'ordine indicato

Ordine di visita e analisi



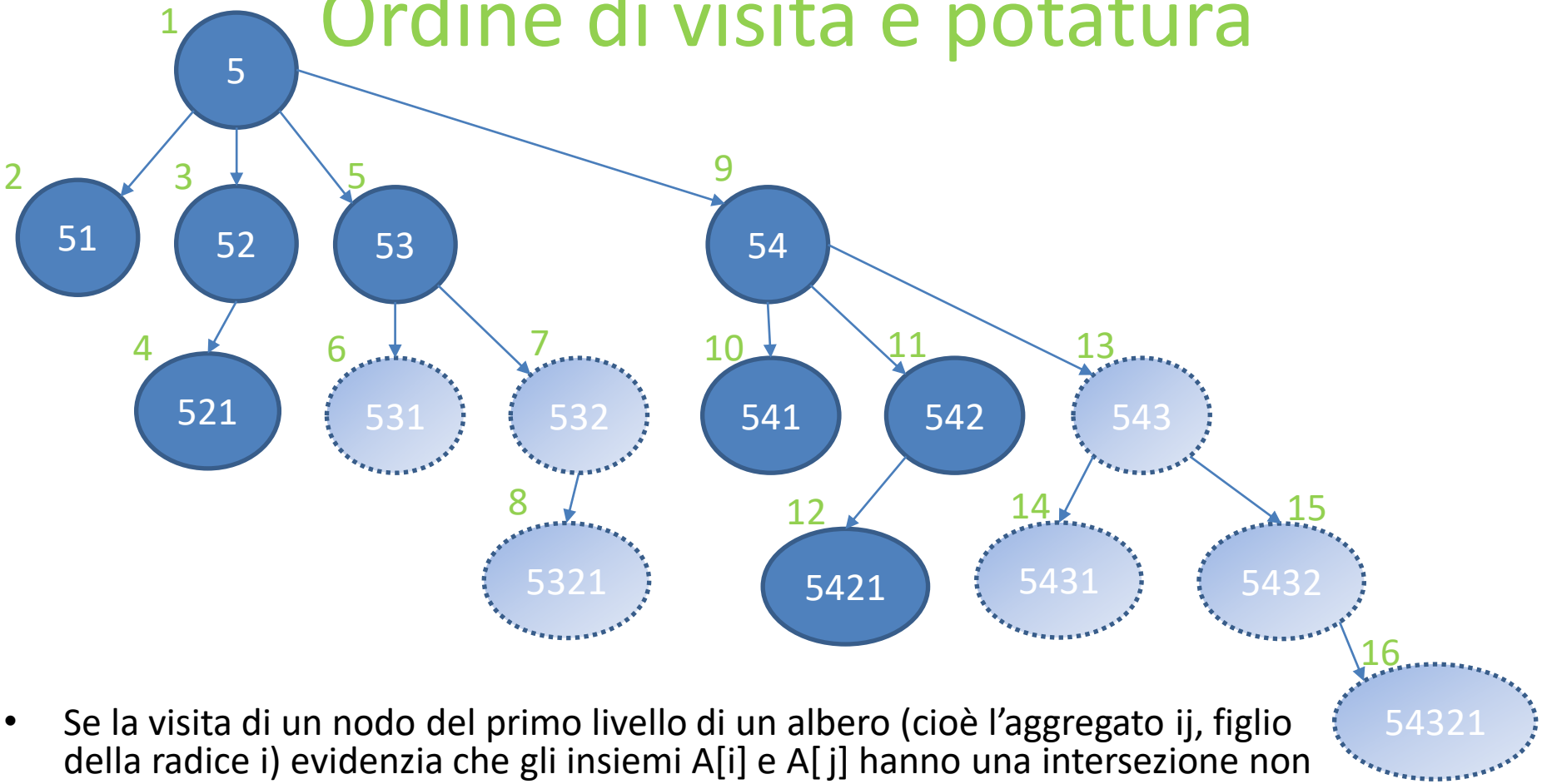
- Dopo che sono state inizializzate le caselle B[1,5] e B[2,5] è possibile analizzare l'aggregato 521 perché le compatibilità degli aggregati 51 e 52 sono note, così come quella dell'aggregato 21, poiché l'albero radicato in 2 è stato visitato prima di quello radicato in 5

Ordine di visita e analisi



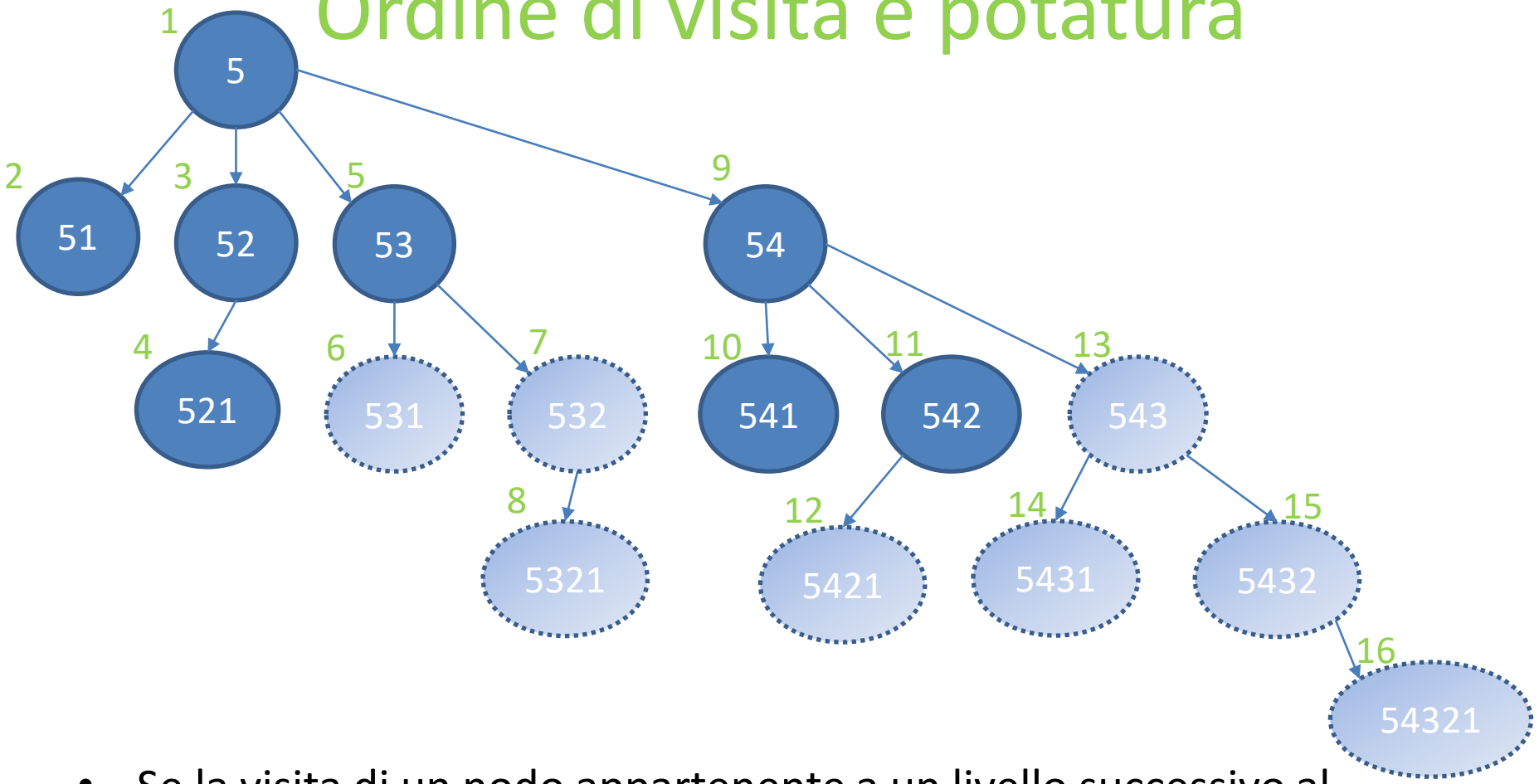
- Dopo che sono state inizializzate le caselle B[1,5], B[2,5] e B[3,5] è possibile analizzare gli aggregati 531 e 532 perché le compatibilità degli aggregati 51, 52 e 53 sono note, così come quelle degli aggregati 31 e 32

Ordine di visita e potatura



- Se la visita di un nodo del primo livello di un albero (cioè l'aggregato ij, figlio della radice i) evidenzia che gli insiemi A[i] e A[j] hanno una intersezione non vuota oppure che la loro unione coincide con M, i discendenti del nodo ij non vengono esplorati così come non vengono esplorati gli aggregati che contengono sia i sia j. Ad es. se l'insieme 5 ha una intersezione non vuota con l'insieme 3 oppure se l'unione degli insiemi 5 e 3 coincide con M, i nodi discendenti del nodo 53 non vengono visitati così come non vengono visitati i nodi appartenenti all'intero sottoalbero radicato in 543, come evidenziato dal nuovo sfondo e contorno di tali nodi

Ordine di visita e potatura



- Se la visita di un nodo appartenente a un livello successivo al primo evidenzia che l'aggregato rappresentato da tale nodo è una partizione, i discendenti di tale nodo non vengono esplorati. Ad es. se l'aggregato 542 è una partizione, i suoi nodi discendenti (ce n'è solo uno) non vengono visitati, come evidenziato dal nuovo sfondo e contorno di tale nodo

ALGORITMO BASE

Pseudocodice

```
1:  procedure EC(A)                ▷ programma principale, A è la matrice d'ingresso
2:      for i ← 1 to rows[A] do
3:          if A[i] == ∅ then
4:              break                          ▷ break termina l'iterazione i-ma
5:          if A[i] == M then
6:              inserire {i} nell'insieme delle partizioni COV, inizialmente vuoto
                                               ▷ COV è una variabile globale
7:              break
8:          in B aggiungere la colonna relativa a i
9:          for j ← 1 to i − 1 do
10:             if A[j] ∩ A[i] ≠ ∅ then
11:                 B[j, i] ← 0
12:             else
13:                 I ← {i, j}, U ← A[i] ∪ A[j]
14:                 if U == M then
15:                     inserire I in COV
16:                     B[j, i] ← 0
17:                 else
18:                     B[j, i] ← 1
19:                     Inter ← B[1..j − 1, i] ∩ B[1..j − 1, j]
20:                     if Inter ≠ ∅ then
21:                         ESPLORA(I, U, Inter)
```

si rammenti pag. 20

Pseudocodice (cont.)

```
1: procedure ESPLORA(I, U, Inter)
2:     for all k ∈ Inter, in ordine lessicografico del valore di k do
3:         Itemp ← I ∪ {k}, Utemp ← U ∪ A[k]
4:         if Utemp == M then
5:             inserire Itemp in COV
6:         else
7:             Intertemp ← Inter ∩ B[1..k − 1, k]
8:             if Intertemp ≠ ∅ then
9:                 ESPLORA(Itemp, Utemp, Intertemp)
```

si rammenti pag. 20

Nella pagina precedente, alla linea 9 (e in quelle successive) per semplicità si è usato i come indice della colonna entro B relativa alle compatibilità dell'insieme A[i], come se gli indici delle righe entro A coincidessero con quelli delle colonne entro B. Attenzione però che tale corrispondenza sussiste solo se la collezione di insiemi rappresentata entro A non contiene insiemi vuoti o coincidenti con M, ai quali, secondo lo pseudocodice, non viene fatta corrispondere alcuna colonna in B. Se si desidera mantenere la corrispondenza degli indici anche in presenza di tali insiemi, è necessario aggiungere per ciascuno di essi in B una colonna e una riga contenente solo valori 0

Implementazione

- Nello pseudocodice delle pagine precedenti, il parametro d'ingresso A è considerato una matrice. Tuttavia nell'implementazione non è necessario che si tratti effettivamente di una matrice intesa come array di array, anche perché tale matrice potrebbe occupare molto spazio di memoria. Pertanto, al di là della struttura dati adottata, della «matrice» A potrebbero essere di volta in volta caricate da file delle porzioni. Il numero di accessi al file aumenterebbe il tempo di esecuzione, riducendo però l'occupazione di memoria centrale

Implementazione

- Nello pseudocodice delle pagine precedenti, la variabile B, corrispondente alla matrice di compatibilità, è considerata una matrice, sebbene della stessa vengano usate solo le caselle sovrastanti la diagonale principale. Nell'implementazione non è necessario che si tratti effettivamente di una matrice intesa come array di array (la scelta della struttura da adottare dovrebbe, come sempre, tenere conto delle operazioni da effettuare sulla stessa)

Implementazione

- La variabile globale COV, che alla fine dell'esecuzione contiene tutte le partizioni (ciascuna descritta sotto forma di insieme degli indici degli insiemi della collezione che appartengono alla partizione stessa) può essere realizzata con le strutture dati più varie. Ad es. essa stessa potrebbe essere una matrice dove ogni elemento assume il valore 0 o il valore 1, ogni riga rappresenta una partizione (aggregato di uno o più insiemi) e l'intestazione di ogni colonna identifica un insieme delle collezione. Questa rappresentazione occupa però spazio, pertanto è interessante optare per alternative quali una lista di stringhe binarie o una lista di insiemi di interi, oppure, ogni volta che viene scoperta una nuova copertura esatta, essa potrebbe essere salvata direttamente nel file di uscita

Esempio di esecuzione di EC

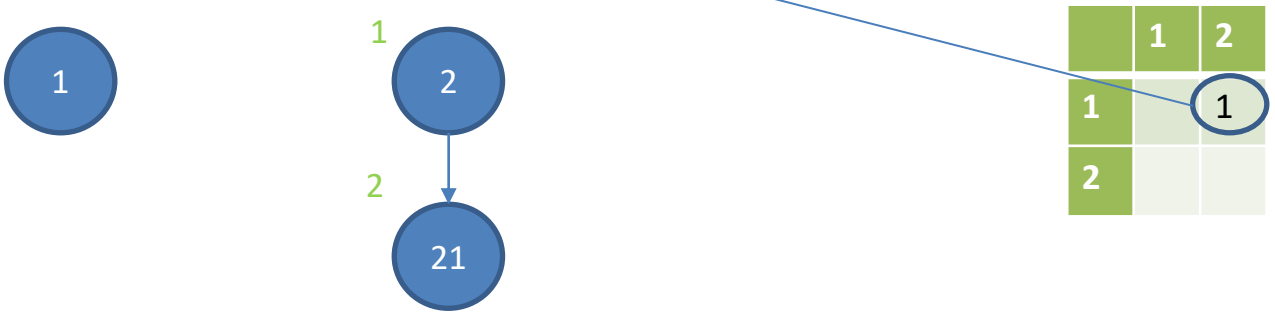
- Matrice A (le intestazioni delle colonne riportano gli elementi del dominio)

	a	b	c	d	e	f	g	h	i	j
1	1		1							1
2				1			1	1	1	
3		1		1				1		
4		1			1	1				
5					1	1	1		1	
6		1	1							1
7	1				1	1				
8	1			1				1		

Alberi radicati in 1 e 2

- 1
- 2
- 21

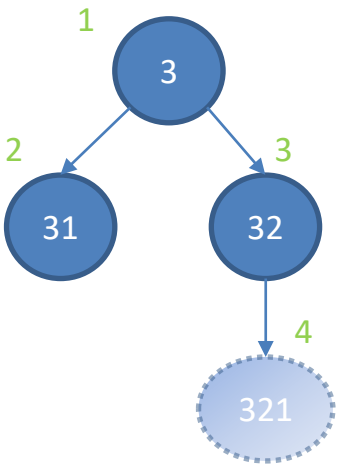
	a	b	c	d	e	f	g	h	i	j
1	1		1							1
2				1			1	1	1	



Albero radicato in 3

- 3
- 31

	a	b	c	d	e	f	g	h	i	j
1	1		1							1
3		1		1				1		



	1	2	3
1		1	1
2			
3			

Albero radicato in 3 (cont.)

- 32 KO

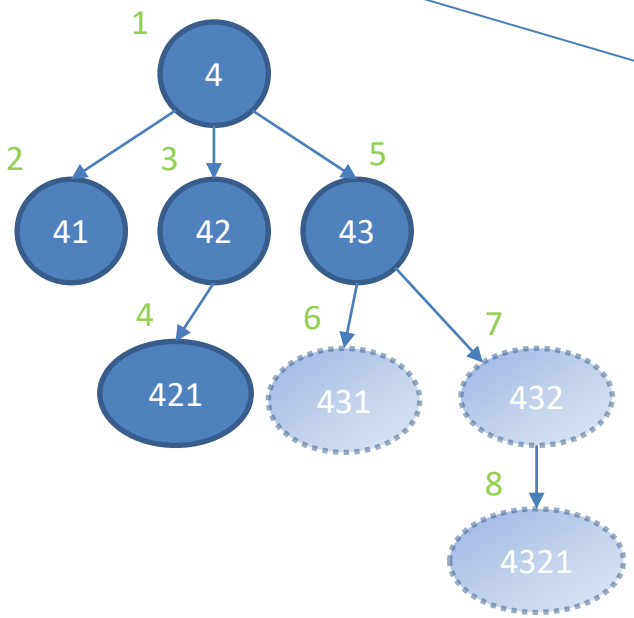
  KO indica il fatto che i due insiemi considerati (A[3] e A[2]) hanno una intersezione non vuota, pertanto sono incompatibili, cioè a B[2,3] viene assegnato il valore 0 (riga 11 dello pseudocodice di pag. 28) e i discendenti del nodo 32 non vengono visitati

	a	b	c	d	e	f	g	h	i	j
2				1			1	1	1	
3		1		1				1		

	1	2	3
1		1	1
2			
3			

Albero radicato in 4

- 4
- 41



	a	b	c	d	e	f	g	h	i	j
1	1		1							1
4		1			1	1				

	1	2	3	4
1		1	1	1
2				
3				
4				

Albero radicato in 4 (cont.)

- 42

	a	b	c	d	e	f	g	h	i	j
2				1			1	1	1	
4		1			1	1				

	1	2	3	4
1		1	1	1
2				1
3				
4				

Albero radicato in 4 (cont.)

Lista COV

421

- 421 COV

  COV indica che l'aggregato considerato è una copertura esatta, pertanto deve essere inserito in COV

	a	b	c	d	e	f	g	h	i	j
1	1		1							1
42		1		1	1	1	1	1	1	

	1	2	3	4
1		1	1	1
2				1
3				
4				

Albero radicato in 4 (cont.)

Lista COV
421

- 43 KO

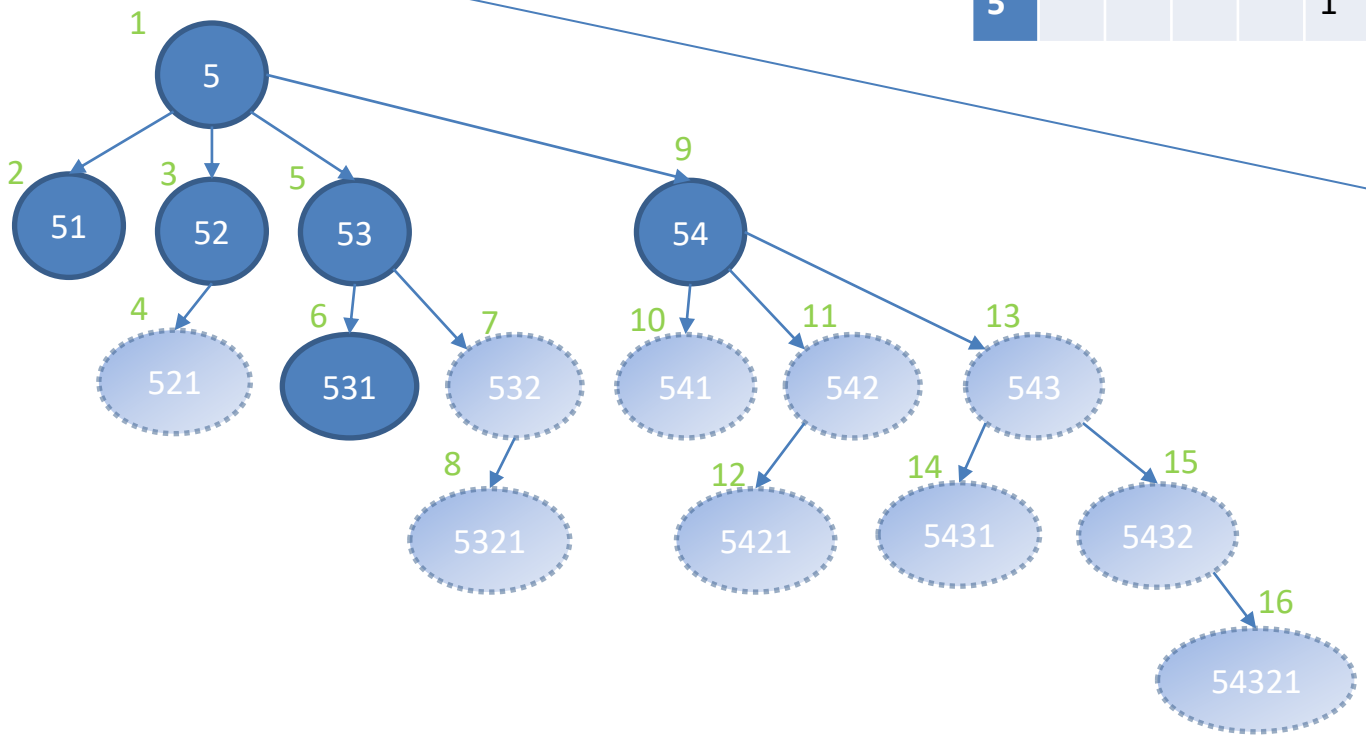
	a	b	c	d	e	f	g	h	i	j
3		1		1				1		
4		1			1	1				

	1	2	3	4
1		1	1	1
2				1
3				
4				

Albero radicato in 5

Lista COV
421

- 5
- 51



	a	b	c	d	e	f	g	h	i	j
1	1		1							1
5					1	1	1		1	

	1	2	3	4	5
1		1	1	1	1
2				1	
3					
4					
5					

Albero radicato in 5 (cont.)

Lista COV
421

- 52 KO

	a	b	c	d	e	f	g	h	i	j
2				1			1	1	1	
5					1	1	1		1	

	1	2	3	4	5
1		1	1	1	1
2				1	
3					
4					
5					

Albero radicato in 5 (cont.)

Lista COV
421

- 53

	a	b	c	d	e	f	g	h	i	j
3		1		1				1		
5					1	1	1		1	

	1	2	3	4	5
1		1	1	1	1
2				1	
3					1
4					
5					

Albero radicato in 5 (cont.)

Lista COV
421
531

- 531 COV

	a	b	c	d	e	f	g	h	i	j
1	1		1							1
53		1		1	1	1	1	1	1	

	1	2	3	4	5
1		1	1	1	1
2				1	
3					1
4					
5					

Albero radicato in 5 (cont.)

Lista COV
421
531

- 54 KO

	a	b	c	d	e	f	g	h	i	j
4		1			1	1				
5					1	1	1		1	

	1	2	3	4	5
1		1	1	1	1
2				1	
3					1
4					
5					

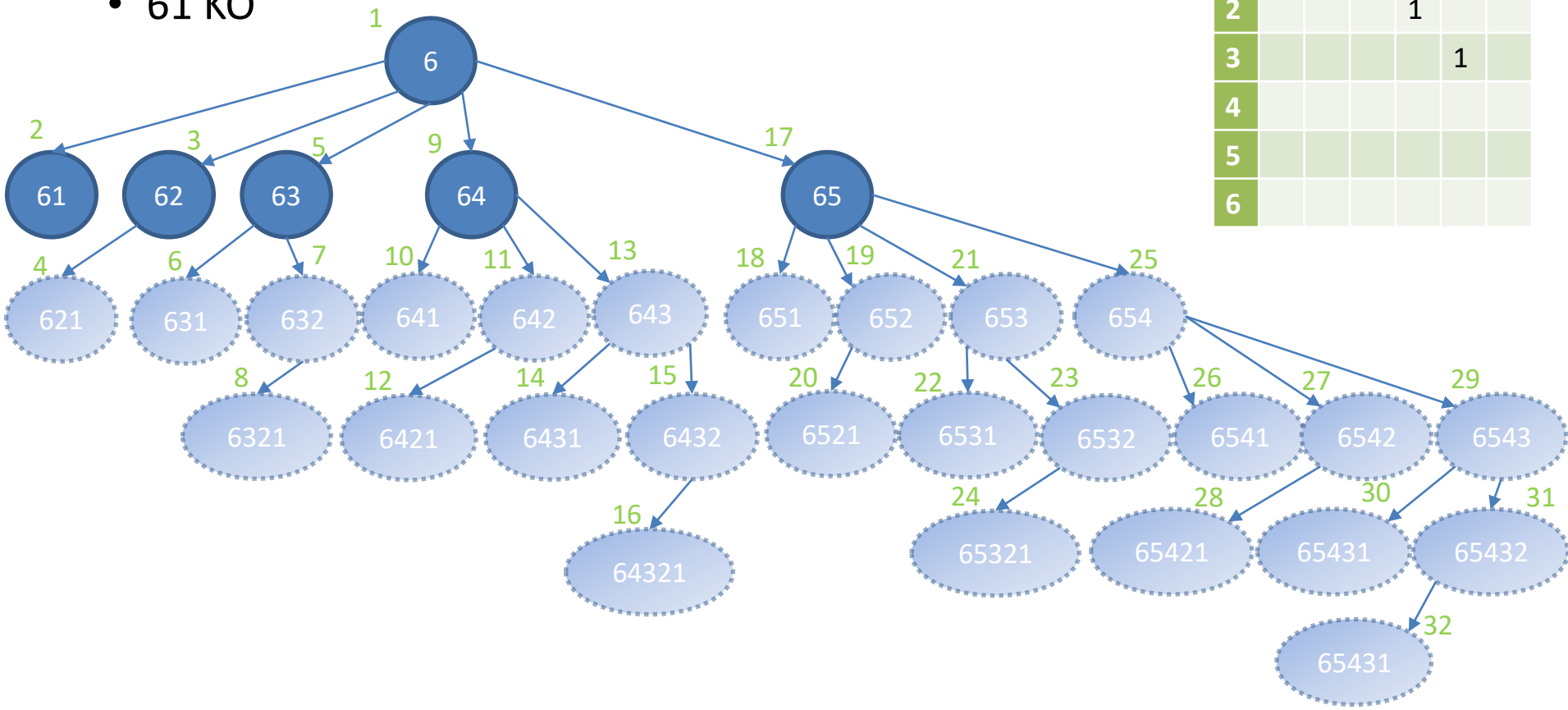
Albero radicato in 6

- 6
- 61 KO

	a	b	c	d	e	f	g	h	i	j
1	1		1							1
6		1	1							1

Lista COV
421
531

	1	2	3	4	5	6
1		1	1	1	1	
2				1		
3					1	
4						
5						
6						



Albero radicato in 6

Lista COV
421
531

- 62

	a	b	c	d	e	f	g	h	i	j
2				1			1	1	1	
6		1	1							1

	1	2	3	4	5	6
1		1	1	1	1	
2				1		1
3					1	
4						
5						
6						

Albero radicato in 6 (cont.)

Lista COV
421
531

- 63 KO

	a	b	c	d	e	f	g	h	i	j
3		1		1				1		
6		1	1							1

	1	2	3	4	5	6
1		1	1	1	1	
2				1		1
3					1	
4						
5						
6						

Albero radicato in 6 (cont.)

Lista COV
421
531

- 64 KO

	a	b	c	d	e	f	g	h	i	j
4		1			1	1				
6		1	1							1

	1	2	3	4	5	6
1		1	1	1	1	
2				1		1
3					1	
4						
5						
6						

Albero radicato in 6 (cont.)

Lista COV
421
531

- 65

	a	b	c	d	e	f	g	h	i	j
5					1	1	1		1	
6		1	1							1

	1	2	3	4	5	6
1		1	1	1	1	
2				1		1
3					1	
4						
5						1
6						

Albero radicato in 7

Lista COV
421
531

- 7
- 71 KO

	a	b	c	d	e	f	g	h	i	j
1	1		1							1
7	1				1	1				

	1	2	3	4	5	6	7
1		1	1	1	1		
2				1		1	
3					1		
4							
5						1	
6							
7							

Albero radicato in 7 (cont.)

Lista COV
421
531

- 72

	a	b	c	d	e	f	g	h	i	j
2				1			1	1	1	
7	1				1	1				

	1	2	3	4	5	6	7
1		1	1	1	1		
2				1		1	1
3					1		
4							
5						1	
6							
7							

Albero radicato in 7

Lista COV
421
531

- 73

	a	b	c	d	e	f	g	h	i	j
3		1		1				1		
7	1				1	1				

	1	2	3	4	5	6	7
1		1	1	1	1		
2				1		1	1
3					1		1
4							
5						1	
6							
7							

Albero radicato in 7 (cont.)

Lista COV
421
531

- 74 KO

	a	b	c	d	e	f	g	h	i	j
4		1			1	1				
7	1				1	1				

	1	2	3	4	5	6	7
1		1	1	1	1		
2				1		1	1
3					1		1
4							
5						1	
6							
7							

Albero radicato in 7 (cont.)

Lista COV
421
531

- 75 KO

	a	b	c	d	e	f	g	h	i	j
5					1	1	1		1	
7	1				1	1				

	1	2	3	4	5	6	7
1		1	1	1	1		
2				1		1	1
3					1		1
4							
5						1	
6							
7							

Albero radicato in 7 (cont.)

Lista COV
421
531

- 76

	a	b	c	d	e	f	g	h	i	j
6		1	1							1
7	1				1	1				

	1	2	3	4	5	6	7
1		1	1	1	1		
2				1		1	1
3					1		1
4							
5						1	
6							1
7							

Albero radicato in 7 (cont.)

Lista COV
421
531
762

- 762 COV

	a	b	c	d	e	f	g	h	i	j
2				1			1	1	1	
76	1	1	1		1	1				1

	1	2	3	4	5	6	7
1		1	1	1	1		
2				1		1	1
3					1		1
4							
5						1	
6							1
7							

Albero radicato in 8

Lista COV
421
531
762

- 8
- 81 KO

	a	b	c	d	e	f	g	h	i	j
1	1				1	1				
8	1			1				1		

	1	2	3	4	5	6	7	8
1		1	1	1	1			
2				1		1	1	
3					1		1	
4								
5						1		
6							1	
7								
8								

Albero radicato in 8 (cont.)

Lista COV
421
531
762

- 82 KO

	a	b	c	d	e	f	g	h	i	j
2				1			1	1	1	
8	1			1				1		

	1	2	3	4	5	6	7	8
1		1	1	1	1			
2				1		1	1	
3					1		1	
4								
5						1		
6							1	
7								
8								

Albero radicato in 8 (cont.)

Lista COV
421
531
762

- 83 KO

	a	b	c	d	e	f	g	h	i	j
3		1		1				1		
8	1			1				1		

	1	2	3	4	5	6	7	8
1		1	1	1	1			
2				1		1	1	
3					1		1	
4								
5						1		
6							1	
7								
8								

Albero radicato in 8 (cont.)

Lista COV
421
531
762

- 84

	a	b	c	d	e	f	g	h	i	j
4		1			1	1				
8	1			1				1		

	1	2	3	4	5	6	7	8
1		1	1	1	1			
2				1		1	1	
3					1		1	
4								1
5						1		
6							1	
7								
8								

Albero radicato in 8 (cont.)

Lista COV
421
531
762

- 85

	a	b	c	d	e	f	g	h	i	j
5					1	1	1		1	
8	1			1				1		

	1	2	3	4	5	6	7	8
1		1	1	1	1			
2				1		1	1	
3					1		1	
4								1
5						1		1
6							1	
7								
8								

Albero radicato in 8 (cont.)

Lista COV
421
531
762

- 86

	a	b	c	d	e	f	g	h	i	j
6		1	1							1
8	1			1				1		

	1	2	3	4	5	6	7	8
1		1	1	1	1			
2				1		1	1	
3					1		1	
4								1
5						1		1
6							1	1
7								
8								

Albero radicato in 8 (cont.)

Lista COV
421
531
762
865

- 865 COV

	a	b	c	d	e	f	g	h	i	j
5					1	1	1		1	
86	1	1	1	1				1		1

	1	2	3	4	5	6	7	8
1		1	1	1	1			
2				1		1	1	
3					1		1	
4								1
5						1		1
6							1	1
7								
8								

Albero radicato in 8 (cont.)

Lista COV
421
531
762
865

- 87 KO
- FINE

	a	b	c	d	e	f	g	h	i	j
7	1				1	1				
8	1			1				1		

- Per ottenere tutte le partizioni, è stato sufficiente visitare 40 nodi (sui 255 che costituiscono il numero totale di aggregati non vuoti dati gli 8 insiemi della collezione in ingresso)

	1	2	3	4	5	6	7	8
1		1	1	1	1			
2				1		1	1	
3					1		1	
4								1
5						1		1
6							1	1
7								
8								

Compito

- Si realizzi una applicazione software modulare che incarni l'algoritmo EC
- Per quanto riguarda i formati di I/O dell'applicazione si faccia riferimento alla sezione «Sperimentazione» di questo documento

ALGORITMO PLUS

Differenze rispetto alla versione base

- La nuova versione dell'algoritmo, proposta di seguito e denominata EC^+, è basata sulla medesima logica di esplorazione degli alberi della precedente (visita esattamente gli stessi nodi) e ne ricalca la struttura di controllo
- La differenza consiste nel fatto che, per gli aggregati di insiemi, non si controlla più se l'unione di tali insiemi coincida con M bensì se la somma delle cardinalità di tali insiemi coincida con $|M|$. I due controlli, quello effettuato dalla prima e quello effettuato dalla seconda versione, si equivalgono per gli aggregati di 3 o più insiemi, dal momento che - per costruzione - essi coinvolgono solo insiemi tutti reciprocamente disgiunti. Nel caso di aggregati di 2 insiemi, il controllo viene effettuato solo dopo avere appurato che essi sono disgiunti

Differenze rispetto alla versione base (cont.)

- Questa variante comporta il calcolo e la memorizzazione della cardinalità di ciascun insieme della collezione (linea 8 dello pseudocodice di cui alla pagina successiva, dove tutte le differenze rispetto alla versione precedente sono sottolineate)
- Tale memorizzazione produce l'effetto di evitare, durante l'intera esecuzione, il computo dell'unione di insiemi
- Resta valido il richiamo all'attenzione di cui a pag. 29. Si noti che l'indice i viene ora usato anche per l'array card: in card[i] viene memorizzata la cardinalità dell'insieme A[i], indicata come |A[i]|

Pseudocodice

```
1:  procedure EC+(A)                                   ▷ A è la matrice d'ingresso
2:      for i ← 1 to rows[A] do
3:          if A[i] == ∅ then
4:              break                                  ▷ break termina l'iterazione i-ma
5:          if A[i] == M then
6:              inserire {i} nell'insieme delle partizioni COV, inizialmente vuoto
                                                       ▷ COV è una variabile globale
7:              break
8:          card[i] ← |A[i]|
9:          in B aggiungere la colonna relative a i
10:         for j ← 1 to i − 1 do
11:             if A[j] ∩ A[i] ≠ ∅ then
12:                 B[j, i] ← 0
13:             else
14:                 I ← {i, j}, cardU ← card[i] + card[j]
15:                 if cardU == |M| then
16:                     inserire I in COV
17:                     B[j, i] ← 0
18:                 else          ▷ se si esegue questa sezione è sicuramente cardU < |M|
19:                     B[j, i] ← 1
20:                     Inter ← B[1..j − 1, i] ∩ B[1..j − 1, j]
21:                     if Inter ≠ ∅ then
22:                         ESPLORA(I, cardU, Inter)
```

si rammenti pag. 20

Pseudocodice (cont.)

```
1: procedure ESPLORA+(I, cardU, Inter)
2:     for all k ∈ Inter, in ordine lessicografico del valore di k do
3:         Itemp ← I ∪ {k}, cardtemp ← cardU + |A[k]|
4:         if cardtemp == |M| then
5:             inserire Itemp in COV
6:         else
7:             Intertemp ← Inter ∩ B[1..k − 1, k]
8:             if Intertemp ≠ ∅ then
9:                 ESPLORA(Itemp, cardtemp, Intertemp)
```

si rammenti pag. 20

Si sottolinea che la versione base e quella plus dell'algoritmo visitano esattamente gli stessi nodi e producono esattamente le stesse uscite, nello stesso ordine

Esempio di esecuzione di EC^+

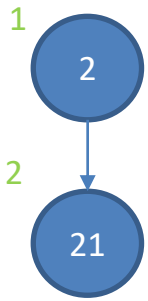
- Matrice A (è la stessa dell'esempio precedente, in cui |M| = 10)

	a	b	c	d	e	f	g	h	i	j
1	1		1							1
2				1			1	1	1	
3		1		1				1		
4		1			1	1				
5					1	1	1		1	
6		1	1							1
7	1				1	1				
8	1			1				1		

Alberi radicati in 1 e 2

- 1
- 2
- 21

	a	b	c	d	e	f	g	h	i	j
1	1		1							1
2				1			1	1	1	



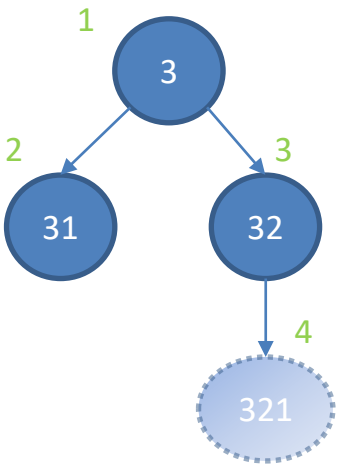
	3	4
	1	2
1		1
2		

card

Albero radicato in 3

- 3
- 31

	a	b	c	d	e	f	g	h	i	j
1	1		1							1
3		1		1				1		



	3	4	3
	1	2	3
1		1	1
2			
3			

Albero radicato in 3 (cont.)

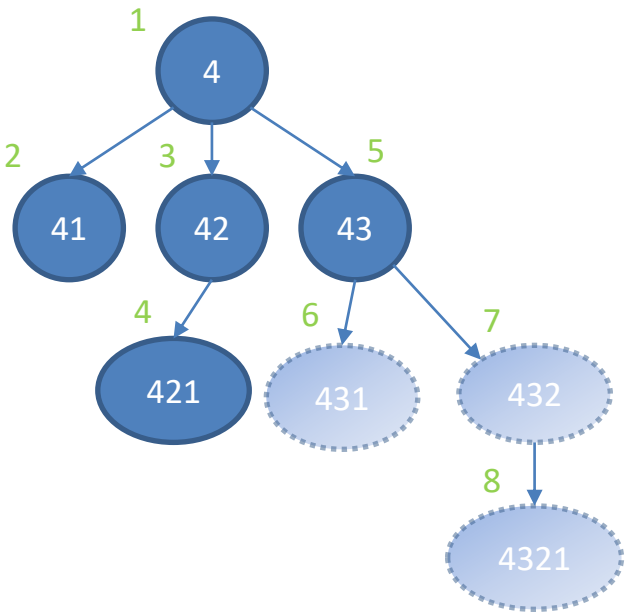
- 32 KO

	a	b	c	d	e	f	g	h	i	j
2				1			1	1	1	
3		1		1				1		

	3	4	3
	1	2	3
1		1	1
2			
3			

Albero radicato in 4

- 4
- 41



	a	b	c	d	e	f	g	h	i	j
1	1		1							1
4		1			1	1				

	3	4	3	3
	1	2	3	4
1		1	1	1
2				
3				
4				

Albero radicato in 4 (cont.)

- 42

	a	b	c	d	e	f	g	h	i	j
2				1			1	1	1	
4		1			1	1				

	3	4	3	3
	1	2	3	4
1		1	1	1
2				1
3				
4				

Albero radicato in 4 (cont.)

Lista COV
421

- 421 COV perché (card[4] + card[2]) + card[1] = (3+4) + 3 = 10

	3	4	3	3
	1	2	3	4
1		1	1	1
2				1
3				
4				

Albero radicato in 4 (cont.)

Lista COV
421

- 43 KO

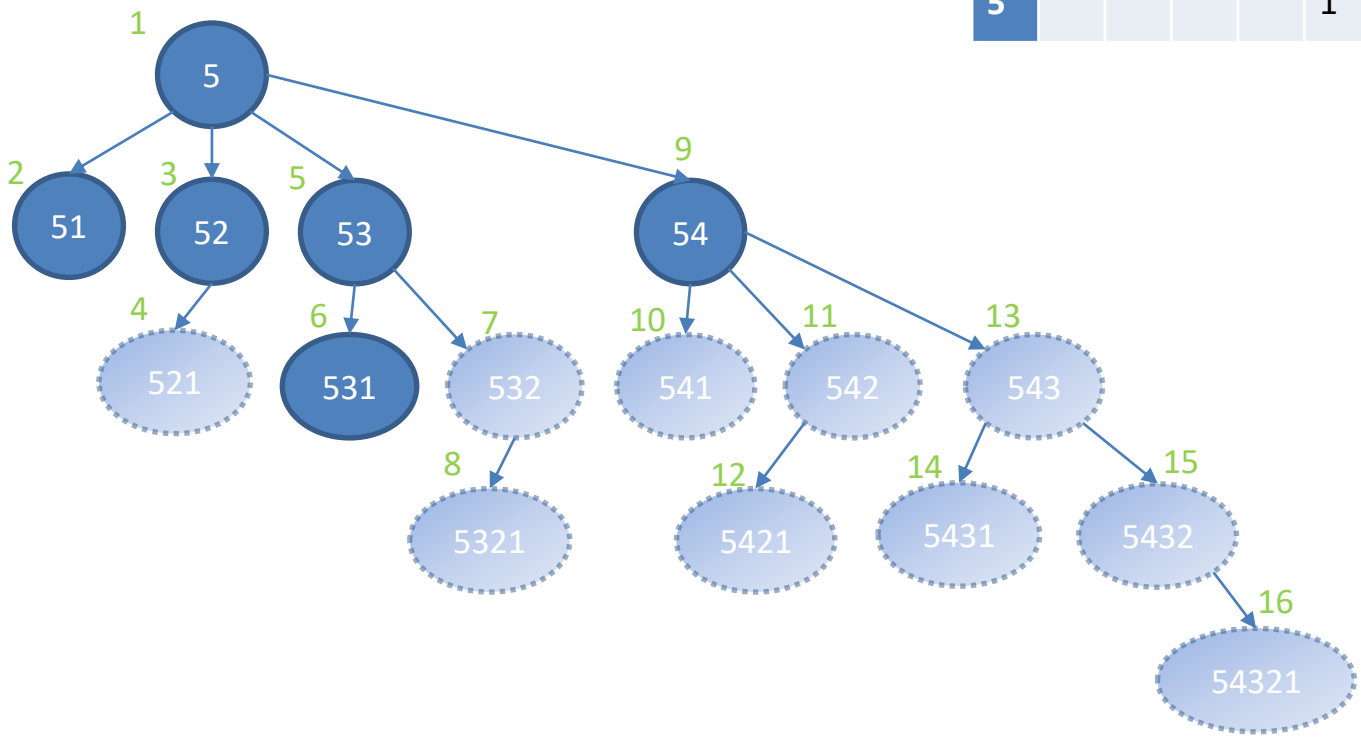
	a	b	c	d	e	f	g	h	i	j
3		1		1				1		
4		1			1	1				

	3	4	3	3
	1	2	3	4
1		1	1	1
2				1
3				
4				

Albero radicato in 5

Lista COV
421

- 5
- 51



	a	b	c	d	e	f	g	h	i	j
1	1		1							1
5					1	1	1		1	

	3	4	3	3	4
	1	2	3	4	5
1		1	1	1	1
2				1	
3					
4					
5					

Albero radicato in 5 (cont.)

Lista COV
421

- 52 KO

	a	b	c	d	e	f	g	h	i	j
2				1			1	1	1	
5					1	1	1		1	

	3	4	3	3	4
	1	2	3	4	5
1		1	1	1	1
2				1	
3					
4					
5					

Albero radicato in 5 (cont.)

Lista COV
421

- 53

	a	b	c	d	e	f	g	h	i	j
3		1		1				1		
5					1	1	1		1	

	3	4	3	3	4
	1	2	3	4	5
1		1	1	1	1
2				1	
3					1
4					
5					

Albero radicato in 5 (cont.)

Lista COV
421
531

- 531 COV perché (card[5] + card[3]) + card[1] = (4+3) + 3 = 10

	3	4	3	3	4
	1	2	3	4	5
1		1	1	1	1
2				1	
3					1
4					
5					

Albero radicato in 5 (cont.)

Lista COV
421
531

- 54 KO

	a	b	c	d	e	f	g	h	i	j
4		1			1	1				
5					1	1	1		1	

	3	4	3	3	4
	1	2	3	4	5
1		1	1	1	1
2				1	
3					1
4					
5					

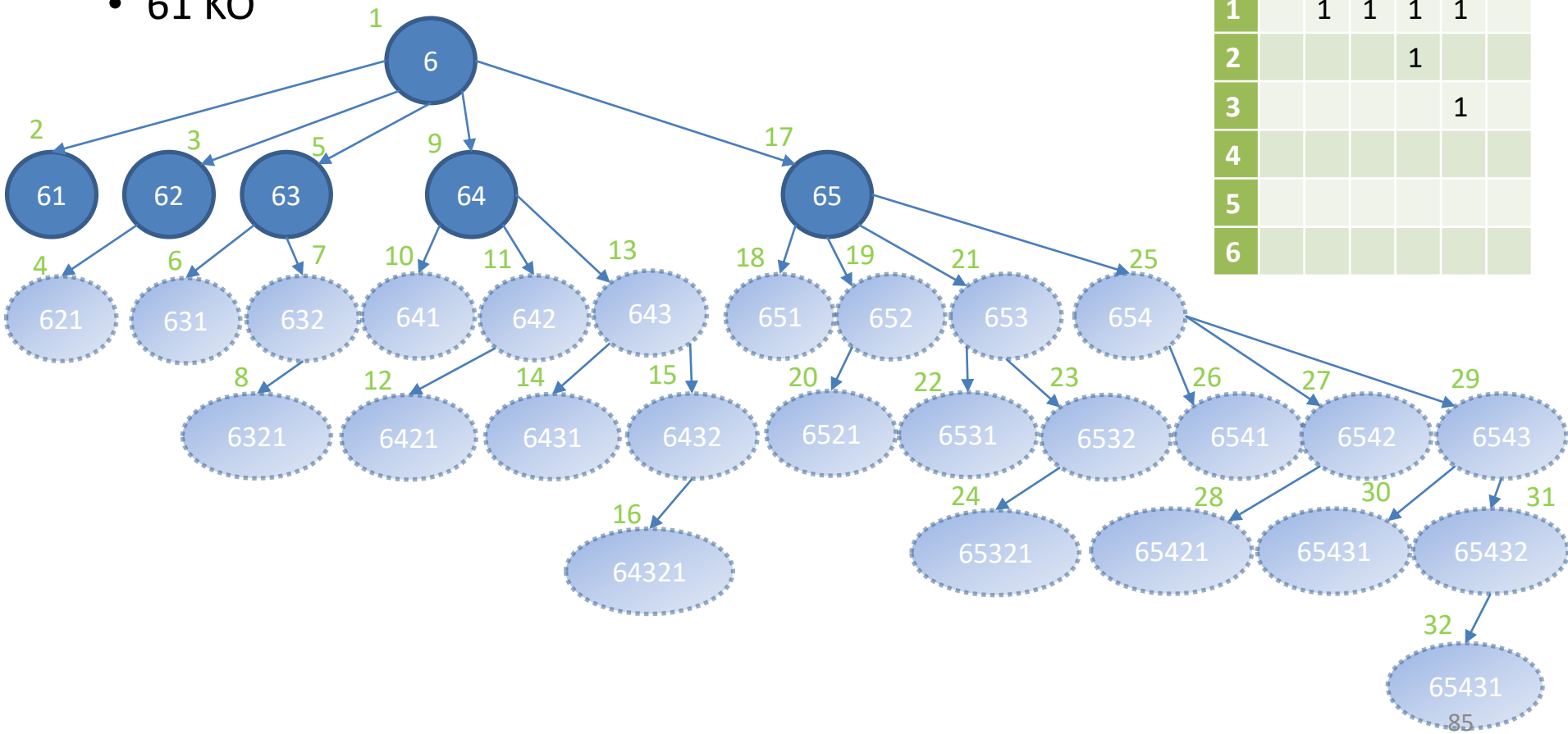
Albero radicato in 6

Lista COV
421
531

	a	b	c	d	e	f	g	h	i	j
1	1		1							1
6		1	1							1

	3	4	3	3	4	3
	1	2	3	4	5	6
1		1	1	1	1	
2				1		
3					1	
4						
5						
6						

- 6
- 61 KO



Albero radicato in 6

Lista COV
421
531

- 62

	a	b	c	d	e	f	g	h	i	j
2				1			1	1	1	
6		1	1							1

	3	4	3	3	4	3
	1	2	3	4	5	6
1		1	1	1	1	
2				1		1
3					1	
4						
5						
6						

Albero radicato in 6 (cont.)

Lista COV
421
531

- 63 KO

	a	b	c	d	e	f	g	h	i	j
3		1		1				1		
6		1	1							1

	3	4	3	3	4	3
	1	2	3	4	5	6
1		1	1	1	1	
2				1		1
3					1	
4						
5						
6						

Albero radicato in 6 (cont.)

Lista COV
421
531

- 64 KO

	a	b	c	d	e	f	g	h	i	j
4		1			1	1				
6		1	1							1

	3	4	3	3	4	3
	1	2	3	4	5	6
1		1	1	1	1	
2				1		
3					1	
4						
5						
6						

Albero radicato in 6 (cont.)

Lista COV
421
531

- 65

	a	b	c	d	e	f	g	h	i	j
5					1	1	1		1	
6		1	1							1

	3	4	3	3	4	3
	1	2	3	4	5	6
1		1	1	1	1	
2				1		
3					1	
4						
5						1
6						

Albero radicato in 7

Lista COV
421
531

- 7
- 71 KO

	a	b	c	d	e	f	g	h	i	j
1	1		1							1
7	1				1	1				

	3	4	3	3	4	3	3
	1	2	3	4	5	6	7
1		1	1	1	1		
2				1		1	
3					1		
4							
5						1	
6							
7							

Albero radicato in 7 (cont.)

Lista COV
421
531

- 72

	a	b	c	d	e	f	g	h	i	j
2				1			1	1	1	
7	1				1	1				

	3	4	3	3	4	3	3
	1	2	3	4	5	6	7
1		1	1	1	1		
2				1		1	1
3					1		
4							
5						1	
6							
7							

Albero radicato in 7

Lista COV
421
531

- 73

	a	b	c	d	e	f	g	h	i	j
3		1		1				1		
7	1				1	1				

	3	4	3	3	4	3	3
	1	2	3	4	5	6	7
1		1	1	1	1		
2				1		1	1
3					1		1
4							
5						1	
6							
7							

Albero radicato in 7 (cont.)

Lista COV
421
531

- 74 KO

	a	b	c	d	e	f	g	h	i	j
4		1			1	1				
7	1				1	1				

	3	4	3	3	4	3	3
	1	2	3	4	5	6	7
1		1	1	1	1		
2				1		1	1
3					1		1
4							
5						1	
6							
7							

Albero radicato in 7 (cont.)

Lista COV
421
531

- 75 KO

	a	b	c	d	e	f	g	h	i	j
5					1	1	1		1	
7	1				1	1				

	3	4	3	3	4	3	3
	1	2	3	4	5	6	7
1		1	1	1	1		
2				1		1	1
3					1		1
4							
5						1	
6							
7							

Albero radicato in 7 (cont.)

Lista COV
421
531

- 76

	a	b	c	d	e	f	g	h	i	j
6		1	1							1
7	1				1	1				

	3	4	3	3	4	3	3
	1	2	3	4	5	6	7
1		1	1	1	1		
2				1		1	1
3					1		1
4							
5						1	
6							1
7							

Albero radicato in 7 (cont.)

Lista COV
421
531
762

- 762 COV perché (card[7] + card[6]) + card[2] = (3+3)+4 = 10

	3	4	3	3	4	3	3
	1	2	3	4	5	6	7
1		1	1	1	1		
2				1		1	1
3					1		1
4							
5						1	
6							1
7							

Albero radicato in 8

Lista COV
421
531
762

- 8
- 81 KO

	a	b	c	d	e	f	g	h	i	j
1	1				1	1				
8	1			1				1		

	3	4	3	3	4	3	3	3
	1	2	3	4	5	6	7	8
1		1	1	1	1			
2				1		1	1	
3					1		1	
4								
5						1		
6							1	
7								
8								

Albero radicato in 8 (cont.)

Lista COV
421
531
762

- 82 KO

	a	b	c	d	e	f	g	h	i	j
2				1			1	1	1	
8	1			1				1		

	3	4	3	3	4	3	3	3
	1	2	3	4	5	6	7	8
1		1	1	1	1			
2				1		1	1	
3					1		1	
4								
5						1		
6							1	
7								
8								

Albero radicato in 8 (cont.)

Lista COV
421
531
762

- 83 KO

	a	b	c	d	e	f	g	h	i	j
3		1		1				1		
8	1			1				1		

	3	4	3	3	4	3	3	3
	1	2	3	4	5	6	7	8
1		1	1	1	1			
2				1		1	1	
3					1		1	
4								
5						1		
6							1	
7								
8								

Albero radicato in 8 (cont.)

Lista COV
421
531
762

- 84

	a	b	c	d	e	f	g	h	i	j
4		1			1	1				
8	1			1				1		

	3	4	3	3	4	3	3	3
	1	2	3	4	5	6	7	8
1		1	1	1	1			
2				1		1	1	
3					1		1	
4								1
5						1		
6							1	
7								
8								

Albero radicato in 8 (cont.)

Lista COV
421
531
762

- 85

	a	b	c	d	e	f	g	h	i	j
5					1	1	1		1	
8	1			1				1		

	3	4	3	3	4	3	3	3
	1	2	3	4	5	6	7	8
1		1	1	1	1			
2				1		1	1	
3					1		1	
4								1
5						1		1
6							1	
7								
8								

Albero radicato in 8 (cont.)

Lista COV
421
531
762

- 86

	a	b	c	d	e	f	g	h	i	j
6		1	1							1
8	1			1				1		

	3	4	3	3	4	3	3	3
	1	2	3	4	5	6	7	8
1		1	1	1	1			
2				1		1	1	
3					1		1	
4								1
5						1		1
6							1	1
7								
8								

Albero radicato in 8 (cont.)

Lista COV
421
531
762
865

- 865 COV perché (card[8] + card[6]) + card[5] = (3+3)+4 = 10

	3	4	3	3	4	3	3	3
	1	2	3	4	5	6	7	8
1		1	1	1	1			
2				1		1	1	
3					1		1	
4								1
5						1		1
6							1	1
7								
8								

Albero radicato in 8 (cont.)

Lista COV
421
531
762
865

- 87 KO
- FINE

	a	b	c	d	e	f	g	h	i	j
7	1				1	1				
8	1			1				1		

	3	4	3	3	4	3	3	3
	1	2	3	4	5	6	7	8
1		1	1	1	1			
2				1		1	1	
3					1		1	
4								1
5						1		1
6							1	1
7								
8								

Compito

- Si estenda l'applicazione software che incarna l'algoritmo EC cosicché sia invocabile anche l'algoritmo EC^{+}
- Si eviti la presenza di cloni nel codice dell'applicazione

SPERIMENTAZIONE

File di ingresso

- Al fine di effettuare un'attività sperimentale relativa alle due versioni dell'algoritmo create, è necessario condurre delle prove di esecuzione. In ciascuna di tali prove si deve passare in ingresso all'algoritmo una collezione di insiemi (idealmente una matrice in cui ciascuna riga rappresenta un insieme distinto). Ciò deve avvenire usando un file di testo (.txt) avente la forma di cui alla pagina successiva

Formato interno dei file d'ingresso

commenti

- ;;; Host = zelda6, Version = 26.1, date = 2010-01-13-17-08-09
  ;;; Source = /tilde/dekleer/projects/GDE/DXC/dxc-09-syn-benchmark-1.1/74l85/74L85.000.scn
  ;;; Error status nil
  ;;; Injected fault:32(o2)
  ;;; Map 1(z1) 2(z2) 3(z3) 4(z4) 5(z5) 6(z6) 7(z7) 8(z8) 9(z9) 10(z10) 11(z11) 12(z12) 13(z13) 14(z14) 15(z15) 16(z16) 17(z17) 18(z18) 19(z19) 20(z20) 21(z21) 22(z22) 23(z23) 24(z24) 25(z25) 26(z26) 27(z27) 28(z28) 29(z29) 30(z30) 31(o1) 32(o2) 33(o3)
  0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 0 -
  0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 -

righe della matrice A; qui ne sono illustrate solo 2

Il numero di elementi per riga coincide con |M|; qui |M| = 33

separatori delle righe della matrice

File di uscita

- Il file di uscita è un file di testo (.txt) su ciascuna riga del quale è rappresentata una partizione sotto forma di insieme degli indici degli insiemi contenuti nella stessa
- Affinché chi legge il contenuto del file di uscita possa comprenderlo e verificarlo, è necessario che l'applicazione sviluppata identifichi ogni insieme della collezione d'ingresso mediante un indice e che tale indice sia noto all'utente; pertanto l'applicazione deve scrivere tale indice (ad es. come commento) entro il file d'ingresso o in un copia dello stesso

Compiti

- Si progetti e sviluppi un programma software che generi i file d'ingresso - per le due versioni dell'algoritmo di cui alle sezioni precedenti del documento - in modo (quasi) casuale; tali file devono contenere la rappresentazione di collezioni aventi dimensioni e caratteristiche diverse in termini di |N|, |M|, distribuzione delle cardinalità degli insiemi, ecc.
- In alternativa, si progetti e sviluppi un programma software che riduca le istanze di un altro problema (ad es. una griglia di Sudoku o un grafo di cui si vuole stabilire se è hamiltoniano) in istanze del problema exact cover e generi i corrispondenti file d'ingresso per le due versioni dell'algoritmo di cui alle sezioni precedenti del documento. Si considerino istanze aventi caratteristiche diverse (ad es. griglie di Sudoku di più livelli difficoltà)

Compiti

- L'uso di file che rappresentano collezioni aventi caratteristiche diverse è teso a rilevare gli andamenti delle prestazioni dei due risolutori al variare di tali caratteristiche
- Si conduca una sperimentazione con entrambi i risolutori realizzati, sottoponendo ogni volta a tutti e due lo stesso file di input e confrontando le uscite prodotte nonché le prestazioni (spaziali e temporali) riscontrate
- Il confronto delle uscite prodotte dai due risolutori deve avvenire automaticamente. Pertanto deve essere progettato e sviluppato (oppure riusato) un programma software che effettui il confronto delle uscite calcolate dalle due versioni. Eventuali discrepanze nei valori delle uscite denunciano la presenza di errori (logici o di programmazione) in uno dei due risolutori o in entrambi e/o nello pseudocodice fornito in queste specifiche

Sui compiti precedenti

- Come già accennato, un fine della sperimentazione è quello di registrare e valutare criticamente le prestazioni delle prove condotte, dal punto di vista sia temporale sia spaziale
- L'applicazione sviluppata potrebbe produrre anche informazioni intermedie (ad esempio, i nodi visitati e l'«esito» - KO o meno - relativo all'analisi di ciascun nodo) reputate di interesse per la valutazione sperimentale
- È consentita ogni forma di riuso del software

RICHIESTE

Gruppi di lavoro

- Ogni gruppo, costituito da due studenti, deve portare a termine i compiti assegnati già descritti nelle sezioni precedenti, redigendo una relazione scritta che illustri il lavoro svolto, le scelte implementative compiute, le prove sperimentali eseguite e una valutazione critica delle stesse

Lavoro e relazione

- Particolare attenzione deve essere dedicata alla scelta di strutture dati volte a estendere le dimensioni (in termini di valori |N| e |M|) delle collezioni di insiemi che possono essere elaborate nonché di altri accorgimenti aventi lo stesso fine. 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 file d'ingresso generato automaticamente, scrivere nel file stesso, sotto forma di commento, le dimensioni della collezione rappresentata (numero di insiemi e cardinalità del dominio) e ogni altra informazione che abbia guidato la generazione della collezione stessa (ad es. distribuzione delle cardinalità degli insiemi) o ritenuta utile ai fini della valutazione sperimentale

Requisiti funzionali (cont.)

- L'applicazione software sviluppata deve:
  - per ogni collezione acquisita in ingresso dai due algoritmi (sia essa stata generata automaticamente o meno), produrre automaticamente un riassunto dell'elaborazione compiuta in termini di dimensioni della collezione (numero di insiemi e cardinalità del dominio), numero di nodi visitati (magari valutando anche la percentuale di nodi visitati rispetto al numero totale dei nodi degli alberi), numero di partizioni prodotte in uscita e distribuzione delle cardinalità delle stesse. Il riassunto relativo all'esecuzione dell'algoritmo EC^+ dovrebbe anche fornire la distribuzione delle cardinalità degli insiemi della collezione in ingresso, ad es. elencando le coppie (valore cardinalità, numero di insiemi dotati di tale cardinalità). Si noti che il file contenente il riassunto di cui sopra potrebbe anche non essere distinto dal file contenente le partizioni in uscita

Requisiti funzionali (cont.)

- L'applicazione software sviluppata deve:
  - 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 da un riassunto analogo a quello del punto precedente, esplicitando però per quali risultati 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 contenente relazione (in formato sia “sorgente”, sia pfd), codice sorgente, eventuale codice eseguibile, tutti i file di ingresso su cui è stata condotta la sperimentazione unitamente ai corrispondenti file di uscita prodotti da ciascuno dei due risolutori e ai relativi riassunti nonché eventuali file (ad es. Excel o Matlab) creati al fine di valutare gli andamenti delle prestazioni
- La consegna della cartella deve avvenire inviando via email il link alla stessa, creato usando un sistema di condivisione (ad es. dropbox, GDrive), oppure attraverso la piattaforma Moodle
- Il progetto descritto in questo documento deve essere presentato entro la sessione d’esame autunnale di Novembre (inclusa) dell’a.a. 2022-2023