ESERCIZIO 1

PREMESSA

La relazione che lega il costo totale conoscendo quello unitario e il numero di oggetti acquistati può essere rappresentata col termine regola(<sigla>,[costo unitario, quantità], <costo totale>). Più in generale, con il termine

regola(<Sigla>,<Lista antecedenti>,<Conseguente>,<Peso>)

si può descrivere ogni regola di deduzione che consente di dedurre <Conseguente> conoscendo tutti gli elementi contenuti nella <Lista antecedenti>; ogni regola è identificata in modo univoco da <Sigla> e da un <Peso> che misura la difficoltà di applicazione di quella regola (per esempio, basso per una somma, più alto per una divisione). Un procedimento di deduzione o di calcolo è rappresentato da un elenco di regole da applicare e quindi può essere descritto dalla lista delle sigle ad esse corrispondenti. Ad ogni procedimento può essere associato un peso complessivo dato dalla somma dei pesi delle singole regole che lo compongono.

PROBLEMA

È dato il seguente insieme di regole (in cui il nome del termine è "rs" invece di "regola"):

```
rs(1,[c1,c2],i,12).     rs(2,[c1,i],c2,7).      rs(3,[c2,i],c1,7).      rs(4,[i,h],a,7).
rs(5,[a,h],i,7).        rs(6,[i,a],h,7).        rs(7,[c1,c2],a,12).     rs(8,[c1,a],c2,12).
rs(9,[c2,a],c1,12).     rs(10,[c1,p1],h,7).     rs(11,[c1,h],p1,7).     rs(12,[p1,h],c1,7).
rs(13,[p1,p2],h,8).     rs(14,[h,p1],p2,7).     rs(15,[p2,h],p1,7).     rs(16,[c2,p2],h,7).
rs(17,[c2,h],p2,7).     rs(18,[p2,h],c2,7).     rs(19,[c1,c2,i],w,12).  rs(20,[c1,h,p1],x,15).
rs(21,[h,p2,c2],y,15).
```

Si osserva che, conoscendo [**c1**,**c2**], è possibile dedurre **i** con la regola 1 e **a** con la regola 7; ma è anche possibile dedurre **h** con la regola 6 dopo aver applicato prima la regola 1 (per dedurre **i**), poi la regola 7 (per dedurre **a**). Quindi, la lista [1,7,6] descrive il procedimento per dedurre **h** conoscendo [**c1**,**c2**]. N.B. Quando due regole possono essere applicate in sequenza e non importa l'ordine, nel procedimento si applichi prima quella con la sigla di valore più basso.

Utilizzando le regole sopra riportate, trovare la lista L1 del procedimento per derivare **x** a partire da [**p1**,**p2**] e calcolarne il peso complessivo K1 e la lista L2 del procedimento per derivare **y** a partire da [**p1**,**c1**] e calcolarne il peso complessivo K2.

L1	
L2	
K1	
K2	

SOLUZIONE

L1	[13,12,20]
L2	[10,14,18,21]
K1	30
K2	36

COMMENTI ALLA SOLUZIONE

Per dedurre **x** esiste solo la regola 20 che può essere applicata conoscendo [**c1**,**h**,**p1**]; **p1** è noto, ma **c1** e **h** non lo sono; per dedurre questi elementi (e costruire l'albero *backward*) si devono cercare le regole appropriate. Il procedimento è [13,12,20].

Analogamente per dedurre **y** esiste solo la regola 21 che può essere applicata conoscendo [**h**,**p2**,**c2**]; per dedurre questi elementi (e costruire l'albero *backward*) si devono cercare le regole appropriate. Il procedimento è [10,14,18,21].

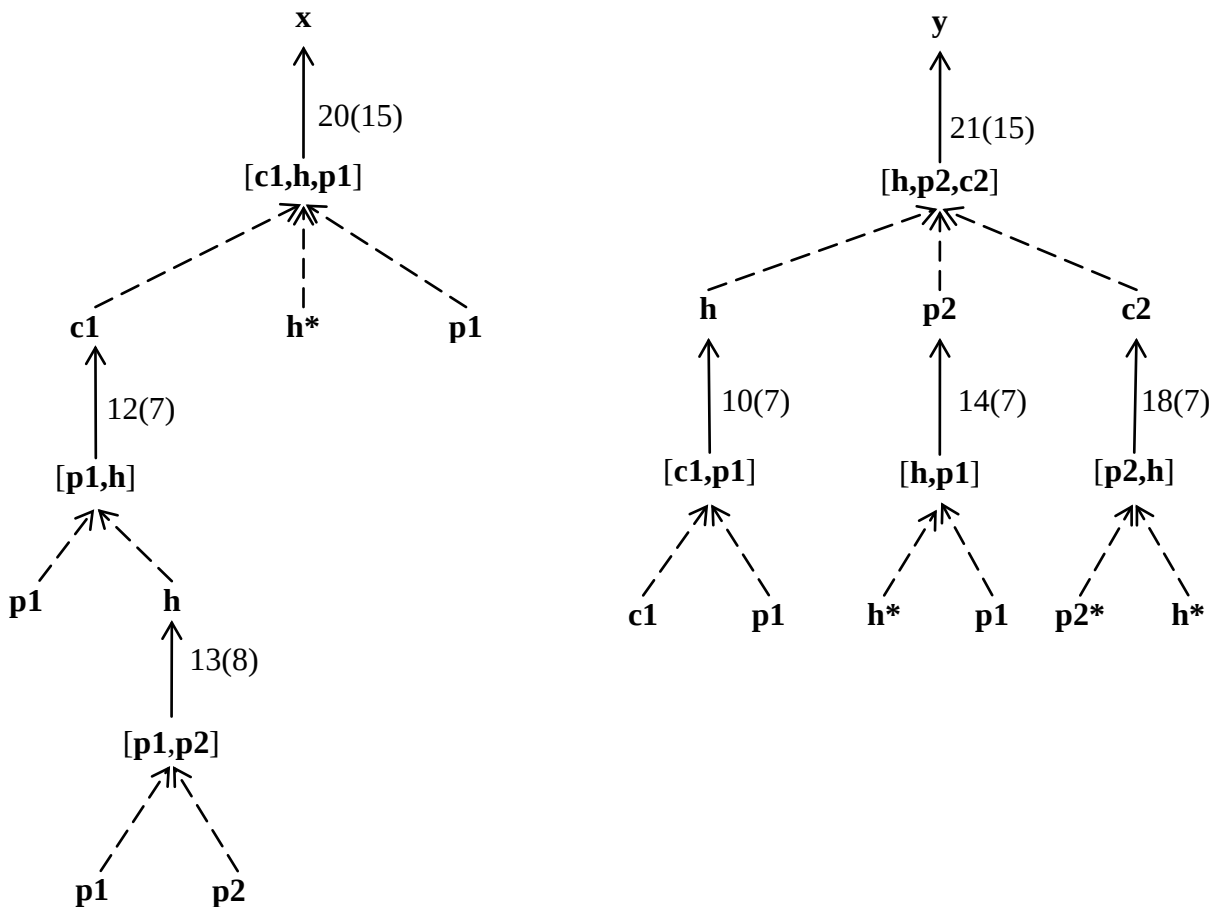
Si ricorda che, in generale, per risolvere il problema si può usare il metodo *backward* (o *top down*) che consiste nel partire dalla incognita e cercare di individuare una regola per derivarla. Se esiste una regola le cui premesse sono tutte note (i dati) la soluzione è trovata, altrimenti si continua a cercare regole per derivare i termini incogniti; il metodo è illustrato nella *prima* figura seguente, in cui le frecce non tratteggiate (di tipo OR) indicano le regole (la sigla è scritta a fianco) e le frecce tratteggiate (di tipo AND) indicano gli antecedenti della regola. In questo modo si trovano procedimenti per derivare l'incognita rappresentati graficamente da alberi, le cui foglie sono (tutte) dati.

I due alberi, mostrati in figura, descrivono i processi di derivazione richiesti da questo problema.

Un altro metodo è quello *forward* (o *bottom up*) che consiste nel partire dai dati e usare le regole applicabili per aumentare la conoscenza via via fino a comprendere l'incognita; il metodo è illustrato nella seconda figura seguente che mostra i due alberi relativi al problema.

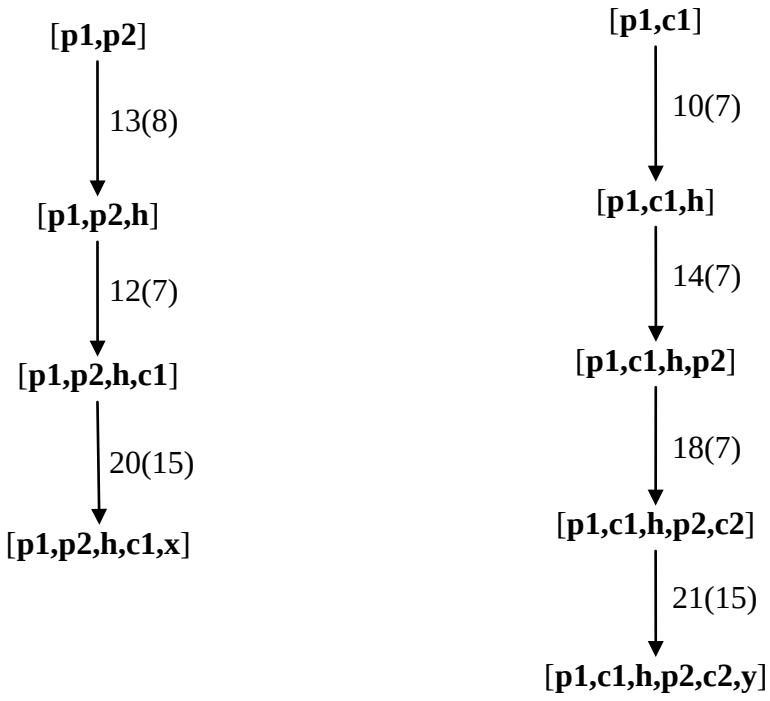
N.B. Nel primo caso la successione delle regole applicate è dal basso verso l'alto; nel secondo caso è dall'alto al basso.

***backward* (o *top down*)**



***** indica che quel nodo è già stato calcolato nello stesso albero e quindi può ritenersi noto

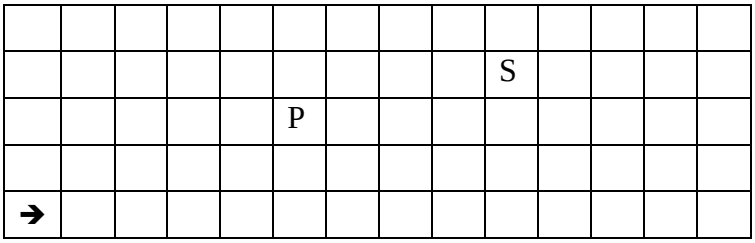
***forward* (o *bottom up*)**



ESERCIZIO 2

PREMESSA

In un foglio a quadretti è disegnato un campo di gara, per esempio di 14 quadretti in orizzontale e 5 in verticale (vedi figura).



Ogni casella può essere individuata da due numeri (interi); per esempio la casella contenente la lettera P è individuata spostandosi di sei colonne da sinistra e di tre righe dal basso: brevemente si dice che ha *coordinate* [6,3]; la prima coordinata (in questo caso 6) si dice *ascissa* e la seconda (in questo caso 3) si dice *ordinata*. Le coordinate della casella contenente la lettera S sono [10,4] e di quella contenente la freccia sono [1,1].

La freccia può essere pensata come un robot, in questo caso rivolto verso destra; il robot può eseguire tre tipi di comandi:

- girarsi di 90 gradi in senso *orario*: comando o;
- girarsi di 90 gradi in senso *antiorario*: comando a;
- avanzare di una casella (nel senso della freccia, mantenendo l'orientamento): comando f.

Questi comandi possono essere concatenati in sequenze in modo da permettere al robot di compiere vari percorsi; per esempio la sequenza di comandi descritta dalla lista [f,f,f,f,a,f,f] fa spostare il robot dalla posizione e orientamento iniziali mostrati in figura fino alla casella P; risultato analogo si ottiene con la lista [a,f,f,o,f,f,f,f,f]. Tuttavia, nel primo caso l'orientamento finale del robot è verso l'alto, mentre nel secondo caso l'orientamento finale è verso destra. Il robot ha sempre uno dei quattro orientamenti seguenti descritti con: n (nord, verso l'alto), s (sud, verso il basso), e (est, verso destra), o (ovest, verso sinistra).

N.B. Non confondere "o" come descrizione dell'orientamento e "o" come comando.

PROBLEMA

In un campo di gara, sufficientemente ampio, si trovano due robot (A e B) che devono compiere due percorsi così definiti:

robot A: coordinate della partenza [1,1], direzione **e**, lista dei comandi:

[f,a,f,f,f,f,o,f,f,f,f,a,f,f,f];

robot B: coordinate della partenza [9,9], direzione **s**, lista dei comandi:

[f,f,f,o,f,f,f,f,a,f,f,f,a,f,f,o,f,o,f,f].

Determinare la lista L delle caselle in cui i due *percorsi* si incrociano.

N.B. quanto segue:

1. una casella si indica con la lista delle sue due coordinate: per esempio [3,4] oppure [11,7];
2. L può essere la lista vuota ([]: vuol dire che i percorsi non si incrociano);
3. L può essere la lista di un solo elemento (per esempio [[4,5]]) o la lista di due elementi (per esempio [[3,4],[9,6]]) o la lista di più elementi;
4. se L ha due o più elementi questi devono comparire in ordine crescente di ascissa; a parità di ascissa, in ordine crescente di ordinata.

L	

SOLUZIONE

L	[[5,5],[6,6]]

COMMENTI ALLA SOLUZIONE

I percorsi dei robot si possono ottenere disegnandoli o, in maniera più sistematica, compilando tabelle come le seguenti, che mostrano lo stato del robot *dopo* ogni comando. (Si noti come l'azione del comando "f", che aumenta o diminuisce una della coordinate, dipende dall'orientamento.)

PRIMO ROBOT

SITUAZIONE	STATO
partenza	[1,1,e]
1 comando: f	[2,1,e]
2 comando: a	[2,1,n]
3 comando: f	[2,2,n]
4 comando: f	[2,3,n]
5 comando: f	[2,4,n]
6 comando: f	[2,5,n]
7 comando: o	[2,5,e]
8 comando: f	[3,5,e]
9 comando: f	[4,5,e]
10 comando:f	[5,5,e]
11 comando: f	[6,5,e]
12 comando: a	[6,5,n]
13 comando: f	[6,6,n]
14 comando: f	[6,7,n]
15 comando: f	[6,8,n]
arrivo	[6,8,n]

SECONDO ROBOT

SITUAZIONE	STATO
partenza	[9,9,s]
1 comando: f	[9,8,s]
2 comando: f	[9,7,s]
3 comando: f	[9,6,s]
4 comando: o	[9,6,o]
5 comando: f	[8,6,o]
6 comando: f	[7,6,o]
7 comando: f	[6,6,o]
8 comando: f	[5,6,o]
9 comando: a	[5,6,s]
10 comando:f	[5,5,s]
11 comando: f	[5,4,s]
12 comando: f	[5,3,s]
13 comando: a	[5,3,e]
14 comando: f	[6,3,e]
15 comando: f	[7,3,e]
16 comando: o	[7,3,s]
17 comando: f	[7,2,s]
18 comando: o	[7,2,o]
19 comando: f	[6,2,o]
20 comando: f	[5,2,o]

arrivo [5,2,o]

Ignorando l'orientamento, il percorso del primo robot è
[[1,1],[2,1],[2,2],[2,3],[2,4],[2,5],[3,5],[4,5],[5,5],[6,5],[6,6],[6,7],[6,8]]
il percorso del secondo robot è
[[9,9],[9,8],[9,7],[9,6],[8,6],[7,6],[6,6],[5,6],[5,5],[5,4],[5,3],[6,3],[7,3],[7,2],[6,2],[5,2]]

Le caselle in comune sono [5,5] e [6,6]

ESERCIZIO 3

PROBLEMA

Nel seguente testo sostituire a ciascun X1, X2, ecc. la parola più appropriata, scelta tra quelle proposte. (N.B. solo una scelta è *coerente* col significato generale del testo, anche se altre sono sintatticamente possibili; per svolgere l'esercizio non è necessario conoscere l'argomento trattato nel brano).

Fra i traduttori attivi in Spagna, il più grande fu forse Gerardo da Cremona (1114-X1). Si era recato in Spagna per imparare l'arabo allo scopo di capire Tolomeo, ma finì per dedicare il resto della sua vita a traduzioni dall'arabo; fra queste la X2 in latino di una versione riveduta della X2 araba degli *Elementi* di Euclide che era stata fatta da Thabit ibn-Qurra. La X2 di Gerardo era molto migliore di quella di Adelardo. Nel 1175 Gerardo tradusse l'*Almagesto*, e fu principalmente attraverso questa X2 che si diffuse in Occidente la X5 di Tolomeo. A Gerardo da Cremona viene attribuita la X2 di oltre ottantacinque opere, ma è possibile datare soltanto quella di X7; fra le sue opere vi è anche un adattamento in latino dell'*Algebra* di al-Kuwarizmi, sebbene una precedente e più popolare X2 dell'*Algebra* fosse stata fatta nel 1145 da Roberto da Chester. A questa X2, la X6 che sia mai stata fatta del trattato di al-Kuwarizmi (così come era stata la X6 del *Corano*, fatta pochi anni prima dallo stesso X4), può essere fatto risalire l'inizio dell'X3 europea.

Lista delle scelte:

A armonia	M arte
B 1167	N scienza
C Gerardo	O Euclide
D versione	P migliore
E nozione	Q terza
F seconda	R 1187
G Roberto	S al-Kuwarizmi
H traduzione	T algebra
I conoscenza	U prima
L Tolomeo	V classica
K voltura	Z 1174

Indicare le scelte con la lettera maiuscola corrispondente.

X1	
X2	
X3	
X4	
X5	
X6	
X7	

SOLUZIONE

X1	R
X2	H
X3	T
X4	G
X5	I
X6	U
X7	L

COMMENTI ALLA SOLUZIONE

Variabile	**Presumibili proprietà grammaticali o sintattiche**	**Scelte possibili**	**Scelta corretta**
X1	data (informa di anno e.c.)	1167, 1187, 1174	1187 (unica scelta compatibile col contesto)
X2	sostantivo femminile	armonia, versione, nozione, voltura, arte, scienza, algebra, conoscenza, traduzione	traduzione (unica possibile dato il numero di occorrenze)
X3	sostantivo femminile	armonia, versione, nozione, voltura, arte, scienza, algebra, conoscenza, traduzione	algebra (la più appropriata, dato il contesto)
X4	nome proprio	Gerardo, Tolomeo, Euclide, al-Kuwarizmi, Roberto	Roberto (l'unica coerente col contesto)
X5	sostantivo femminile	armonia, versione, nozione, voltura, arte, scienza, algebra, conoscenza, traduzione	conoscenza (la più appropriata, dato il contesto)
X6	aggettivo femminile	seconda, prima, terza, migliore, classica	prima (la più appropriata, dato il contesto)
X7	nome proprio	Gerardo, Tolomeo, Euclide, al-Kuwarizmi, Roberto	Tolomeo (l'unica coerente col contesto)

ESERCIZIO 4

PREMESSA

In un foglio a quadretti è disegnato un campo di gara di dimensioni 14x5 (14 quadretti in orizzontale e 5 in verticale, vedi figura).

		Q											
		5	■	■		■			S				
			7	P									
		1											
🚹													

Ogni casella può essere individuata da due numeri (interi); per esempio la casella contenente la lettera P è individuata spostandosi di sei colonne da sinistra e di tre righe dal basso: brevemente si dice che ha *coordinate* [5,3]; la prima coordinata (in questo caso 5) si dice *ascissa* e la seconda (in questo caso 3) si dice *ordinata*. Le coordinate della casella contenente la lettera S sono [10,4] e di quella contenente il robot 🚹 sono [1,1].

Il robot si muove a passi e ad ogni passo (o mossa) può spostarsi solo in una delle caselle contenenti ♘ come illustrato nella seguente figura (allo stesso modo del *cavallo* nel gioco degli scacchi).

	♘		♘	
♘				♘
		🚹		
♘				♘
	♘		♘	

Ciascuna mossa del cavallo può essere individuata dalle rispettive direzioni come indicate nella rosa dei venti (nord-nord-est, est-nord-est, est-sud-est, sud-sud-est, sud-sud-ovest, ovest-sud-ovest, ovest-nord-ovest, nord-nord-ovest) e riprodotte nel seguente schema

	nno		nne	
ono				ene
		🚹		
oso				ese
	sso		sse	

Il campo di gara contiene caselle interdette al robot (segnate da un quadrato nero) quindi, tenuto conto anche dei bordi del campo di gara, la mobilità del robot può essere limitata; ad esempio se il robot si trovasse nella casella in cui c'è Q si potrebbe spostare solo in 3 caselle; se fosse nella casella in cui c'è P avrebbe 7 mosse possibili; dalla casella [1,1] ha solo 2 mosse possibili. In alcune caselle sono posti dei premi che il robot può accumulare lungo un percorso. Ogni premio è descritto fornendo le coordinate della casella che lo contiene e il valore del premio: i premi sopra riportati sono descritti dalla seguente lista [[3,2,1],[4,3,7],[3,4,5]]. Un percorso del robot è descritto dalla li-

sta delle coordinate delle caselle attraversate. Un possibile percorso da P (coordinate [5,3]) a Q (coordinate [3,5]) è descritto dalla seguente lista: [[5,3],[3,2],[5,1],[4,3],[3,5]] e ha un totale di premi accumulati pari a 8. Se fosse vietata la direzione ese, il robot non potrebbe saltare da [3,2] a [5,1] e un percorso da P a Q (con valore dei premi pari a 1) potrebbe essere

[[5,3],[3,2],[1,1],[2,3],[3,5]].

PROBLEMA

In un campo di gara di dimensioni 5×5, il robot si trova in [1,1]; deve fare un percorso *chiuso*, cioè partire dalla casella iniziale e ritornarci, e *semplice*, cioè senza passare due volte in una stessa casella (quindi tutte le caselle del percorso sono diverse, tranne la partenza e l'arrivo); nel campo sono presenti le caselle interdette descritte dalla seguente lista

[[3,4],[2,1],[1,4],[4,1],[2,4],[2,5],[3,3],[3,5],[5,2]].

I premi distribuiti nel campo di gara sono descritti dalla seguente lista

[[2,3,5],[5,4,9],[3,2,6]].

Al robot è, inoltre, vietata la mossa corrispondente alla direzione est-nord-est (ene).

Trovare:

A. il numero N di possibili percorsi diversi chiusi e semplici;
B. la lista L dei valori dei premi accumulati in questi percorsi, elencati in ordine non decrescente.

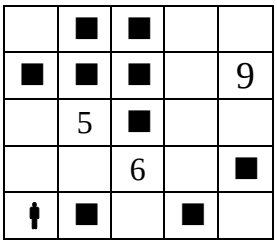
N	
L	

SOLUZIONE

N	2
L	[5,11]

COMMENTI ALLA SOLUZIONE

Il campo di gara è mostrato nello schema seguente:



I percorsi possibili sono:

1. [[1,1],[2,3],[1,1]], valore dei premi 5
2. [[1,1],[2,3],[3,1],[4,3],[5,1],[3,2],[1,1]]; valore dei premi 11

ESERCIZIO 5

PREMESSA

Alcuni ragazzi decidono di costruire un ipertesto multimediale sugli avvenimenti storici significativi della loro regione. Per organizzare il progetto, dividono il lavoro in singole attività e assegnano ogni attività a un gruppo di loro.

Le attività sono descritte col seguente termine

a(<sigla attività>,<durata in giorni>,<ragazzi impegnati>);

esempio, il termine a(A1,1,6) significa che l'attività A1 dura un giorno e impiega 6 ragazzi.

Le attività non possono svolgersi tutte contemporaneamente, ma devono essere rispettate delle priorità descritte con termini del tipo

p(<precedente>,<successiva>);

come per esempio p(A4,A8) e p(A6,A8); ogni termine esprime il fatto che l'attività associata alla sigla di destra (detta successiva) può iniziare solo quando l'attività associata alla sigla di sinistra (detta precedente) è terminata. Ovviamente se una attività ha più precedenti, può iniziare solo quando *tutte* le precedenti sono terminate; i due termini appena visti implicano che l'attività A8 può iniziare solo dopo che sono terminate le due attività A4 e A6.

N.B. Si dice *prima attività* del progetto quella che non ha precedenti; si dice *ultima attività* del progetto quella che non ha successive; in un progetto "ben fatto" sono uniche.

PROBLEMA.

Le attività di questo progetto sono descritte nella seguente lista:

[a(A1,1,3),a(A2,2,2),a(A3,2,3),a(A4,2,1),a(A5,1,1),a(A6,2,4),a(A7,2,2),a(A8,2,3),a(A9,2,6),a(A10,1,4),a(A11,1,3),a(A12,1,3) ,a(A13,2,7),a(A14,2,1),a(A15,1,2),a(A16,2,1),a(A17,1,1),a(A18,1,3)].

Le priorità sono descritte dalla seguente lista:

[p(A1,A2),p(A1,A3),p(A2,A4),p(A2,A5),p(A3,A6),p(A3,A7),p(A4,A8),p(A5,A8),p(A5,A15),p(A6,A12),p(A7,A11),p(A7,A10),p(A9,A12),p(A6,A13),p(A11,A14),p(A10,A14),p(A13,A18),p(A12,A18) ,p(A3,A5),p(A8,A9),p(A14,A18),p(A1,A17),p(A17,A7),p(A3,A16),p(A16,A11),p(A15,A12)].

Si supponga che siano disponibili a lavorare *contemporaneamente* al progetto *solamente* 10 ragazzi; posto che ogni attività inizi *prima possibile* (nel rispetto delle priorità) e col vincolo naturale di non impiegare contemporaneamente più risorse di quelle disponibili, determinare:

- il numero (minimo) N di giorni necessari per completare il progetto;
- l'unicità U: cioè se, nel numero N di giorni, esiste una sola maniera di organizzare il progetto.

N.B. Per l'unicità U rispondere SI oppure NO (in lettere maiuscole).

N	
U	

SOLUZIONE

N	12
U	NO

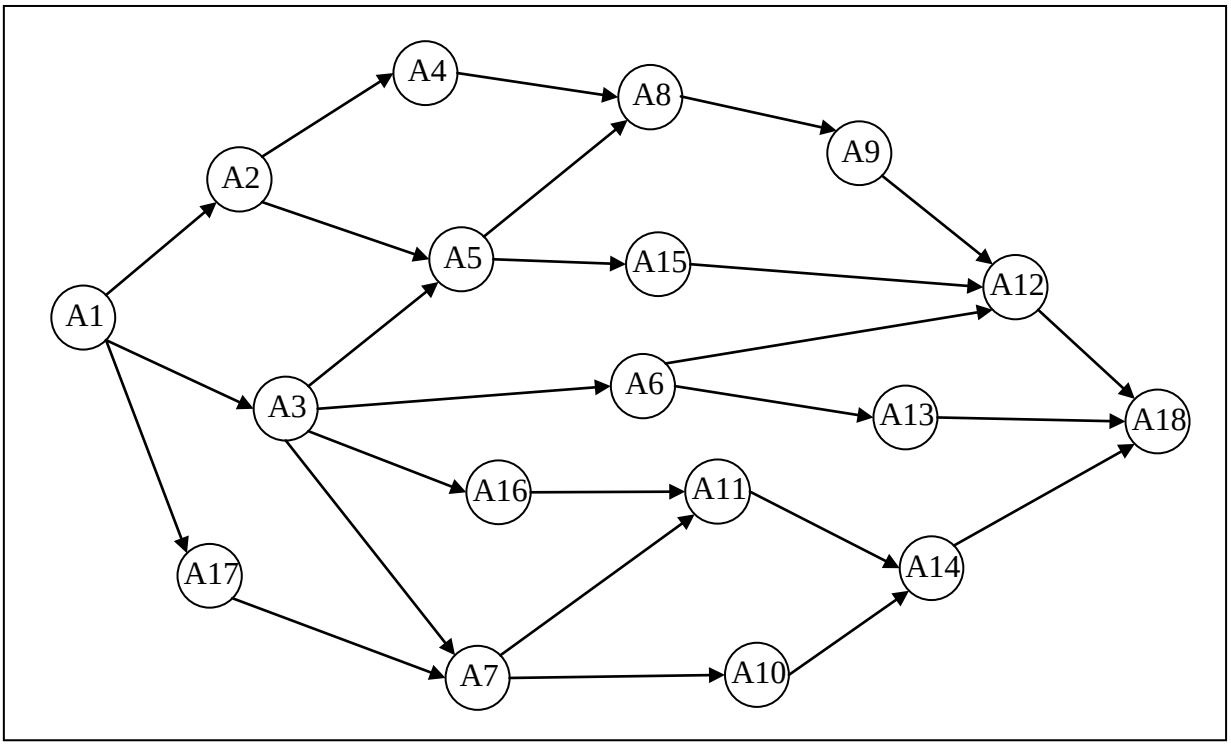
COMMENTI ALLA SOLUZIONE

Per facilitare la soluzione è utile trasformare in tabella la lista che descrive la durata e le persone relative ad ogni attività.

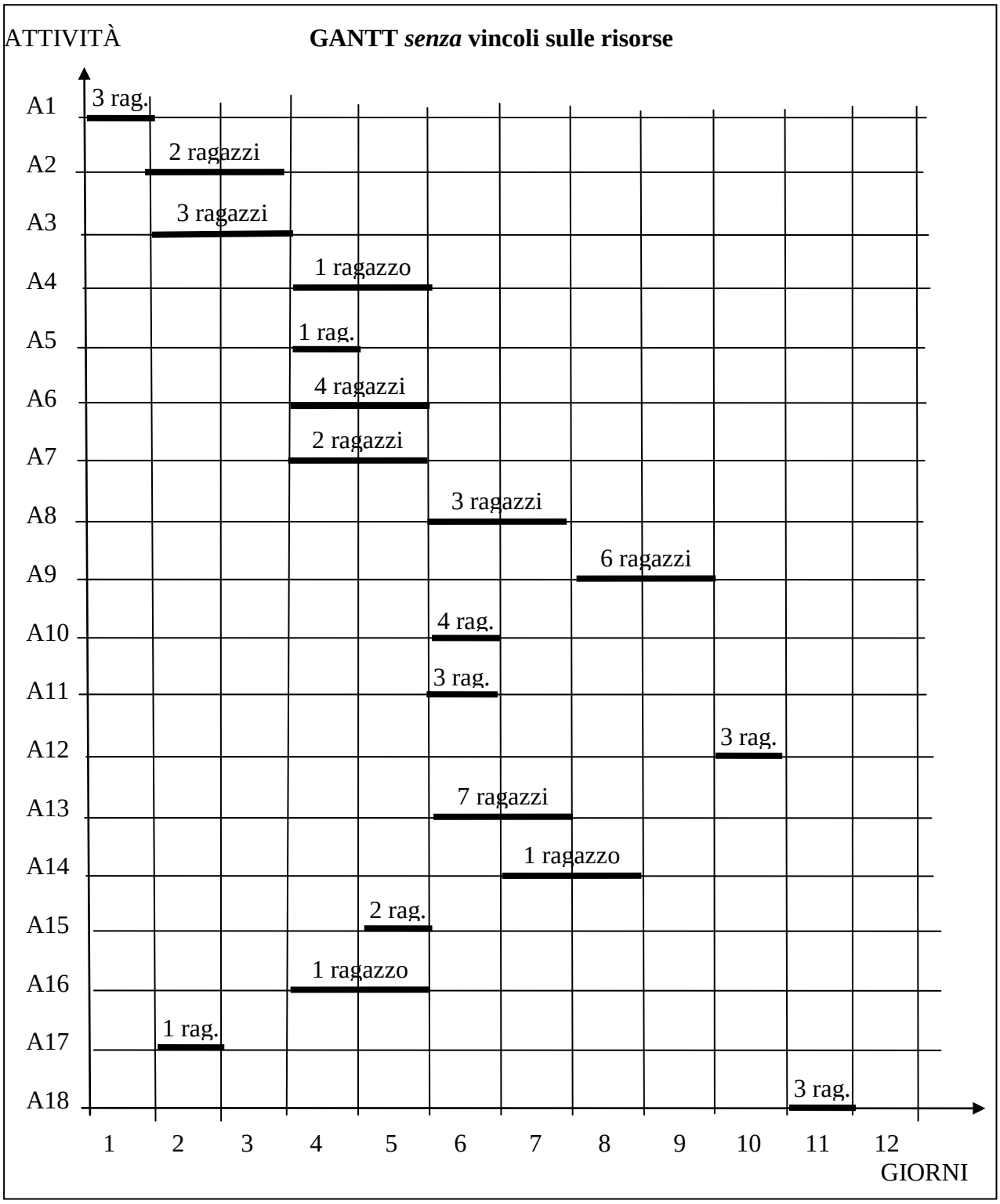
Attività	Durata	Ragazzi
A1	1	3

A2	2	2
A3	2	3
A4	2	1
A5	1	1
A6	2	4
A7	2	2
A8	2	3
A9	2	6
A10	1	4
A11	1	3
A12	1	3
A13	2	7
A14	2	1
A15	1	2
A16	2	1
A17	1	1
A18	1	3

Successivamente è bene disegnare il diagramma delle precedenze, cioè il grafo che ha come nodi le attività e come frecce le precedenze. Si procede per passi successivi: prima si disegnano in "ordine sparso" dei nodi con etichette che sono le attività (per esempio un cerchietto con la sigla della attività); poi si congiungono con una freccia i nodi che appartengono a un elemento (sottolista) della lista che rappresenta le priorità; successivamente si procede a ridisegnare, per tentativi, il grafo cercando di "disintrecciare" le frecce: di solito ci si riesce completamente, come nell'esempio in figura (ma non sempre è possibile). Si noti che esiste una "prima attività" del progetto (A1 in figura) e una "ultima attività" (A18 in figura).
N.B. È casuale che la prima attività abbia la sigla più piccola e l'ultima la sigla più grande.
Il diagramma delle precedenze esprime in maniera molto "leggibile" la precedenza tra le attività e consente di passare con facilità allo stadio successivo.



Dal grafo e dalla tabella si può compilare il Gantt; si elencano le attività sull'asse verticale, avendo cura di iniziare (dall'alto) con la prima attività e finire in basso con l'ultima; sull'asse orizzontale si elencano i giorni: *a priori* non si può dire quanti saranno necessari (certamente non più della somma di quelli che compaiono nella tabella che descrive le attività).
Di seguito è mostrato il Gantt "standard" (cioè che *non* tiene conto dei vincoli sulle risorse: la disponibilità di solo 10 ragazzi) che prevede il completamento del progetto in 11 giorni. Successivamente sono mostrati *due* Gantt che, non impiegando più di 10 ragazzi in un giorno, prevedono il completamento del progetto in 12 giorni.

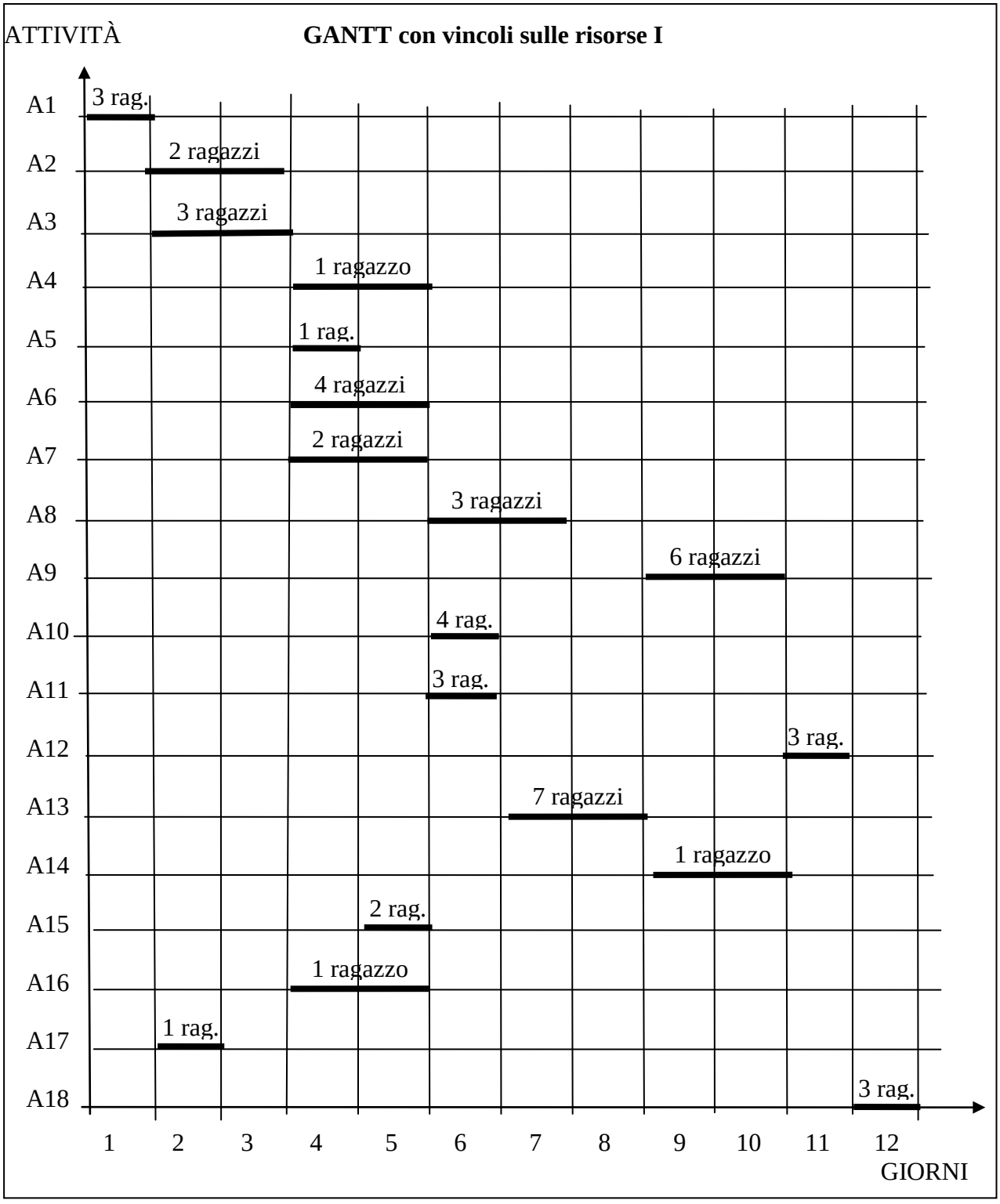


N.B. I giorni sono numerati a partire dal primo del progetto.
Dal Gantt si deduce facilmente la lista delle coppie [<giorno>, <ragazzi al lavoro>]:

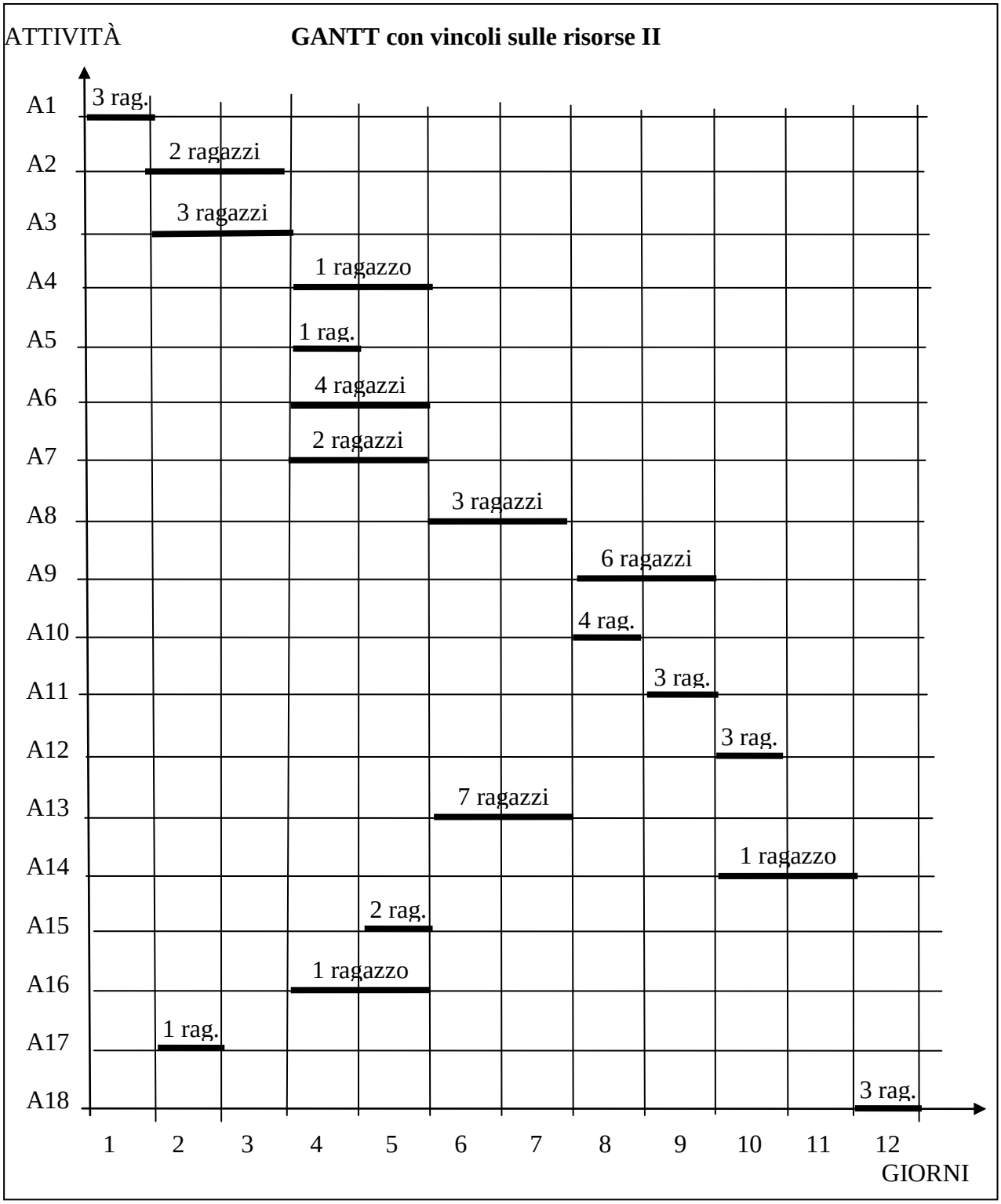
[[1,3],[2,6],[3,5],[4,9],[5,10],[6,17],[7,11],[8,7],[9,6],[10,3],[11,3]].

Si vede che i giorni 6 e 7 sono violati i vincoli sulle risorse, perché è previsto l'impiego rispettivamente di 17 e 11 ragazzi

Di seguito sono mostrati due Gantt che prevedono 12 giorni per essere completati e impiegano solo (al più) 10 ragazzi al giorno.



Dal Gantt si deduce facilmente la lista delle coppie [<giorno>, <ragazzi al lavoro>]:
[[1,3],[2,6],[3,5],[4,9],[5,10],[6,10],[7,10],[8,7],[9,7],[10,7],[11,3],[12,3]].



Dal Gantt si deduce facilmente la lista delle coppie [<giorno>, <ragazzi al lavoro>]:
[[1,3],[2,6],[3,5],[4,9],[5,10],[6,10],[7,10],[8,10],[9,9],[10,4],[11,1],[12,3]].

ESERCIZIO 6

PROBLEMA
Compresa la sequenza dei calcoli descritti nella procedura seguente, eseguire le operazioni indicate utilizzando i dati di input sotto riportati e trovare i valori di output.

```
procedure PROVA1;
variables A, B, K integer;
K ← 0;
input A, B;
while A ≥ B do
        A ← A - B;
        K ← K + 1;
endwhile;
output A, K;
endprocedure;
```

Completare la seguente tabella con i valori di output.

Lista dei valori in input per A e B	Lista dei valori in output per A e K
[30,12]	
[40,7]	
[60,11]	

SOLUZIONE

Lista dei valori in input per A e B	Lista dei valori in output per A e K
[30,12]	[6,2]
[40,7]	[5,5]
[60,11]	[5,5]

COMMENTI ALLA SOLUZIONE
I tre casi si possono riassumere con le seguenti tabelle:

Valori di input per A, B e K prima del “while”	[30,12,0]
valori di A, B e K dopo la prima esecuzione del ciclo “while”	[18,12,1]
valori di A, B e K dopo la seconda esecuzione del ciclo “while”	[6,12,2]

Valori di input per A, B e K prima del “while”	[40,7,0]
valori di A, B e K dopo la prima esecuzione del ciclo “while”	[33,7,1]
valori di A, B e K dopo la seconda esecuzione del ciclo “while”	[26,7,2]
valori di A, B e K dopo la terza esecuzione del ciclo “while”	[19,7,3]
valori di A, B e K dopo la quarta esecuzione del ciclo “while”	[12,7,4]
valori di A, B e K dopo la quinta esecuzione del ciclo “while”	[5,7,5]

Valori di input per A, B e K prima del “while”	[60,11,0]
valori di A, B e K dopo la prima esecuzione del ciclo “while”	[49,11,1]
valori di A, B e K dopo la seconda esecuzione del ciclo “while”	[38,11,2]
valori di A, B e K dopo la terza esecuzione del ciclo “while”	[27,11,3]
valori di A, B e K dopo la quarta esecuzione del ciclo “while”	[16,11,4]
valori di A, B e K dopo la quinta esecuzione del ciclo “while”	[5,11,5]

ESERCIZIO 7

PREMESSA
Nello pseudolinguaggio oltre al tipo “integer” che descrive variabili che hanno valore intero, esiste anche il tipo “real”, che descrive variabili che hanno valore razionale: un tale valore si può pensare come un numero (in rappresentazione decimale) “con il punto”. In alcuni linguaggi di programmazione è usato il termine “float” invece di “real”.
N.B. Si segue la convenzione anglosassone di scrivere i numeri decimali col “.” *e non con la* “,”.

PROBLEMA
Compresa la sequenza dei calcoli descritti nella procedura seguente, eseguire le operazioni indicate utilizzando i dati di input sotto riportati e trovare i valori di output.

```
procedure PROVA2;
variables N integer;
variables A, B, C, A1, B1, C1 real;
input N, A, B, C;
for I from 1 to N step 1 do
        A1 ← A – B + C;
        B1 ← A + B - C;
        C1 ← A - C + B;
        A ← A1;
        B ← B1;
        C ← C1;
endfor;
output A1, B1, C1;
endprocedure;
```

Con i valori di input per N, A, B e C riportati in tabella, calcolare i valori di output per A1, B1, C1.

N	A	B	C	A1	B1	C1
9	1.0	7.0	13.0			
20	15.0	2.0	19.0			

SOLUZIONE

N	A	B	C	A1	B1	C1
9	1.0	7.0	13.0	7.0	7.0	7.0
20	15.0	2.0	19.0	32.0	32.0	32.0

COMMENTI ALLA SOLUZIONE
I due casi sono riassunti nelle seguenti tabelle.

PRIMO CASO N vale 9	I	A	B	C	A1	B1	C1
prima del ciclo “for”	\	1.0	7.0	13.0	\	\	\
dopo la prima esecuzione del ciclo “for”	1	7.0	-5.0	-5.0	7.0	-5.0	-5.0
dopo la seconda esecuzione del ciclo “for”	2	7.0	7.0	7.0	7.0	7.0	7.0
. . .							
dopo la nona esecuzione del ciclo “for”	9	7.0	7.0	7.0	7.0	7.0	7.0

SECONDO CASO N vale 20	I	A	B	C	A1	B1	C1
prima del ciclo “for”	\	15.0	2.0	19.0	\	\	\

dopo la prima esecuzione del ciclo “for”	1	32.0	-2.0	-2.0	32.0	-2.0	-2.0
dopo la seconda esecuzione del ciclo “for”	2	32.0	32.0	32.0	32.0	32.0	32.0
. . .							
dopo la ventesima esecuzione del ciclo “for”	20	32.0	32.0	32.0	32.0	32.0	32.0

Esaminando il ciclo “for” si vede che i valori di B1 e C1 sono sempre eguali; quindi, dopo la prima esecuzione, il valore di A1 (e quindi di A) si propaga a B1 e C1, rimanendo costante nelle esecuzioni successive.

ESERCIZIO 8

PROBLEMA

Si consideri la funzione $f(x)$ (da numeri naturali a numeri naturali) definita come segue:

$$f(0) = 1$$
$$f(n+1) = f(n)(1+f(n))$$

Si costruisca la lista L contenente nell'ordine i valori di $f(1)$, $f(2)$, $f(3)$, $f(4)$, $f(5)$.

L	

SOLUZIONE

L	[2,6,42,1806,3263442]

COMMENTI ALLA SOLUZIONE

I valori sono facilmente ottenibili:

f(0) = 1;

$f(1) = f(0)\ (1+f(0)) = 1\times(1+1) = 2$

$f(2) = f(1)\ (1+f(1)) = 2\times(1+2) = 6$

$f(3) = f(2)\ (1+f(2)) = 6\times(1+6) = 42$

$f(4) = f(3)\ (1+f(3)) = 42\times(1+42) = 1806$

$f(5) = f(4)\ (1+f(4)) = 1806\times(1+1806) = 3263442$

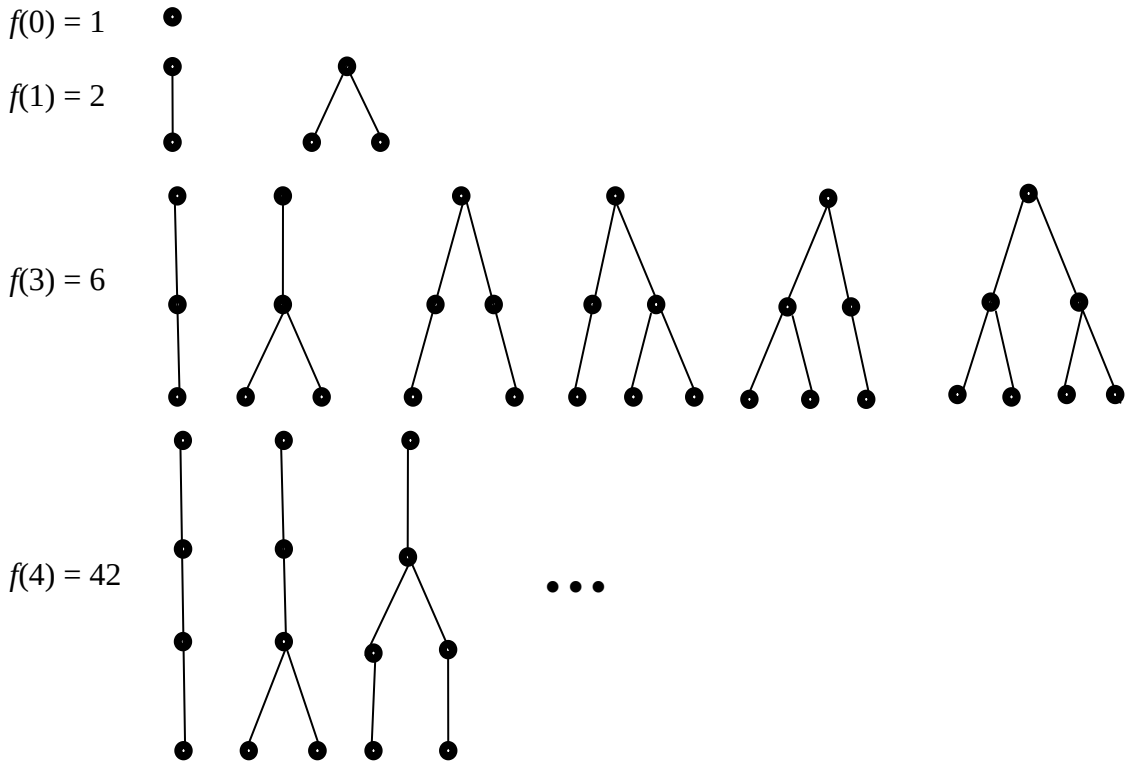
APPROFONDIMENTI

Un'altra (ovvia) maniera comune di definire la funzione è la seguente:

$$f(0) = 1$$
$$f(n) = (f(n-1))^2 + f(n-1)$$

La funzione, o meglio la successione dei suoi valori, ha molteplici impieghi in informatica teorica, in logica e in matematica discreta, ed ha una origine molto antica.

La figura seguente illustra come i valori di $f(n)$ contino, per esempio, il numero di alberi (ordinati)



con nodi che abbiano 0, 1 oppure 2 discendenti e tutte le foglie al livello *n*. Un utile esercizio è provare questa affermazione.
Hint: è piuttosto laborioso contare in quanti modi può essere completato con un altro livello di foglie un (*n*-1)-albero (per farlo diventare un *n*-albero): è molto più semplice contare in quanti modi si può "aggiungere" a una radice un (*n*-1)-albero per ottenere un *n*-albero: si veda l'illustrazione per il passaggio da *f*(0) a *f*(1).

La funzione $S(n) = f(n) + 1$ è detta funzione di Sylvester (matematico inglese che la studiò intorno al 1880) e i suoi valori sono chiamati successione di Sylvester: sono quindi 2, 3, 7, 43, 1807, …
Un fatto importante nella teoria dei numeri è che 1 si può esprimere come somma infinita di frazioni che hanno al numeratore 1 (dette *frazioni egizie*) e al denominatore i numeri di Sylvester:

$$1 = \frac{1}{2} + \frac{1}{3} + \frac{1}{7} + \frac{1}{43} + \frac{1}{1807} + \cdots$$

È notevole che la somma troncata a un numero finito di termini diventa esatta se si diminuisce di uno il denominatore dell'ultima frazione:

$$1 = \frac{1}{1};$$

$$1 = \frac{1}{2} + \frac{1}{2};$$

$$1 = \frac{1}{2} + \frac{1}{3} + \frac{1}{6};$$

$$1 = \frac{1}{2} + \frac{1}{3} + \frac{1}{7} + \frac{1}{42};$$

Le funzioni come $f(n)$ e $S(n)$, da numeri naturali a numeri naturali, sono state definite mediante un metodo detto "*ricorsivo*".
Tale metodo è di validità molto generale nel *problem solving* e (in maniera molto concisa) consiste nel descrivere la soluzione di un problema (per esempio il calcolo di una funzione per un certo valore *n*) in termini delle soluzioni di problemi dello stesso tipo ma più "semplici" (negli esempi visti i *valori della stessa funzione per valori di n più piccoli*). Naturalmente per evitare una *regressio ad infinitum* occorre dare esplicitamente la soluzione del problema in uno o più casi "semplici". Negli esempi visti è stato assegnato il valore delle funzioni nel caso "semplice" in cui $n = 0$.
Come altro esempio si consideri $F(n)$ la funzione di Fibonacci i cui valori sono appunto i numeri di Fibonacci: 0, 1, 1, 2, 3, 5, 8, 13, 21, ecc.
Una sua definizione in termini ricorsivi è la seguente:

$$F(0) = 0$$

$$F(1) = 1$$

$$F(n) = F(n-1) + F(n-2)$$

Si noti come la descrizione della soluzione nel caso generale $F(n)$ è assegnata in funzione di *due* casi più semplici (i due precedenti valori di *n*): occorre quindi dare esplicitamente la soluzione per i *due* casi più semplici in assoluto (0 ed 1).
Come ulteriore esempio di funzione ricorsiva, formalmente un po' più complicato, si consideri:

$$I(0) = 0$$

$$I(n) = I(I(n-1)) + 1$$

La complicazione è connessa al fatto che la funzione *I* è "annidata": cioè compare nell'argomento di un'altra funzione; i valori sono comunque facili da calcolare:

$I(0) = 0$

$I(1) = I(I(1-1)) + 1 = I(I(0)) + 1 = I(0) + 1 = 0 + 1 = 1$

$I(2) = I(I(2-1)) + 1 = I(I(1)) + 1 = I(1) + 1 = 1 + 1 = 2$

$I(3) = I(I(3-1)) + 1 = I(I(2)) + 1 = I(2) + 1 = 2 + 1 = 3$

$I(4) = I(I(4-1)) + 1 = I(I(3)) + 1 = I(3) + 1 = 3 + 1 = 4$

ecc.

N.B. Ovviamente quando ci sono funzioni annidate si comincia (almeno in questi casi) col valutare la funzione più "interna".

In linea di principio ci sono due modi per calcolare i valori di una funzione definita in termini ricorsivi: *bottom up* o *top down*. Nel primo caso si parte dai casi semplici e si "sale", utilizzando i valori precedentemente calcolati; esempi sono i calcoli delle funzioni *f* ed *I* appena visti. Un altro esempio è il seguente, riguardante la funzione di Fibonacci: supponiamo di voler calcolare *F*(8).

F(0) = 0
F(1) = 1
F(2) = *F*(1) + *F*(0) = 1 + 0 = 1
F(3) = *F*(2) + *F*(1) = 1 + 1 = 2
F(4) = *F*(3) + *F*(2) = 2 + 1 = 3
F(5) = *F*(4) + *F*(3) = 3 + 2 = 5
F(6) = *F*(5) + *F*(4) = 5 + 3 = 8
F(7) = *F*(6) + *F*(5) = 8 + 5 = 13
F(8) = *F*(7) + *F*(6) = 13 + 8 = 21

L'altro metodo, *top down*, viene illustrato di seguito, sempre per calcolare *F*(8).

F(8) = *F*(7) + *F*(6) =
F(6) + *F*(5) + *F*(6) = 2 *F*(6) + *F*(5) =
2 (*F*(5) + *F*(4)) + *F*(5) = 3 *F*(5) + 2 *F*(4) =
3 (*F*(4) + *F*(3)) + 2 *F*(4) = 5 *F*(4) + 3 *F*(3) =
5 (*F*(3) + *F*(2)) + 3 *F*(3) = 8 *F*(3) + 5 *F*(2) =
8 (*F*(2) + *F*(1)) + 5 *F*(2) = 13 *F*(2) + 8 *F*(1) =
13 (*F*(1) + *F*(0)) + 8 *F*(1) = 21 *F*(1) + *F*(0) = 21

In questo secondo caso occorre prestare attenzione a "raccogliere" espressioni simili in modo da "calcolarle" una volta sola. (Questa osservazione è particolarmente importante se il metodo di calcolo *top down* viene realizzato con un programma che consente "chiamate ricorsive".)

N.B. Nel caso di *F* i due metodi sono "essenzialmente equivalenti", nel senso che richiedono *circa* lo stesso numero di calcoli: non è sempre così in generale.

ESERCIZIO 9

PREMESSA

Un certo numero di pacchi, di peso complessivo *P*, deve essere suddiviso in due contenitori, ciascuno dei quali può contenere un peso massimo *P*/2, con la possibilità di scartare al più un *solo* pacco (che quindi non viene inserito in nessun contenitore).

N.B. Si prende in considerazione solo il *peso* dei pacchi, non altri parametri come volume o dimensioni; in particolare due pacchi dello stesso peso sono indistinguibili.

Per suddividere i pacchi tra i contenitori si usa il seguente metodo:

- disporre i pacchi "in fila" in un ordine iniziale *qualunque*;
- aprire il primo contenitore;
- inserire uno dopo l'altro nel primo contenitore i pacchi della fila, fino a quando si incontra un pacco che non entra più nel primo contenitore;
- lasciare il pacco in fila, chiudere il primo contenitore e aprire il secondo contenitore;
- inserire uno dopo l'altro nel secondo contenitore i pacchi rimasti nella fila: se si incontra un pacco che non entra nel secondo contenitore, scartare il pacco e continuare con i successivi.

Si verifica facilmente che il metodo funziona (sempre) per due pacchi: se i due pacchi hanno peso eguale, ciascuno viene sistemato in un contenitore, altrimenti uno solo (quello di peso più grande) viene scartato.

PROBLEMA

Sono dati 5 pacchi, di peso rispettivamente 10,11,30,40,11 chilogrammi.

Il metodo esposto precedentemente nella PREMESSA permette che si riesca sempre a riempire correttamente i due contenitori con tutti i pacchi, scartandone al più uno? Oppure il metodo *fallisce*, cioè c'è qualche ordine iniziale con il quale avanza più di un pacco?

Determinare la lista L delle liste dei pacchi dati per cui il metodo fallisce; naturalmente se il metodo ha (sempre) successo L è la lista vuota (da indicare con []), altrimenti essa contiene come elementi una o più liste con 5 elementi. Se ci sono più liste queste devono comparire in ordine crescente del primo elemento, quelle che hanno lo stesso primo elemento devono comparire in ordine crescente del secondo, e così via.

L	

SOLUZIONE

L	[[30,40,10,11,11],[30,40,11,10,11],[30,40,11,11,10]]

COMMENTI ALLA SOLUZIONE

Il peso complessivo dei pacchi è 102 chilogrammi, quindi ogni contenitore è da 51 chilogrammi; si noti la dissimmetria tra il riempimento del primo contenitore (che si interrompe appena un pacco non entra) e quello del secondo (che continua fino ad esaurimento dei pacchi, *scartando* quelli che non entrano).

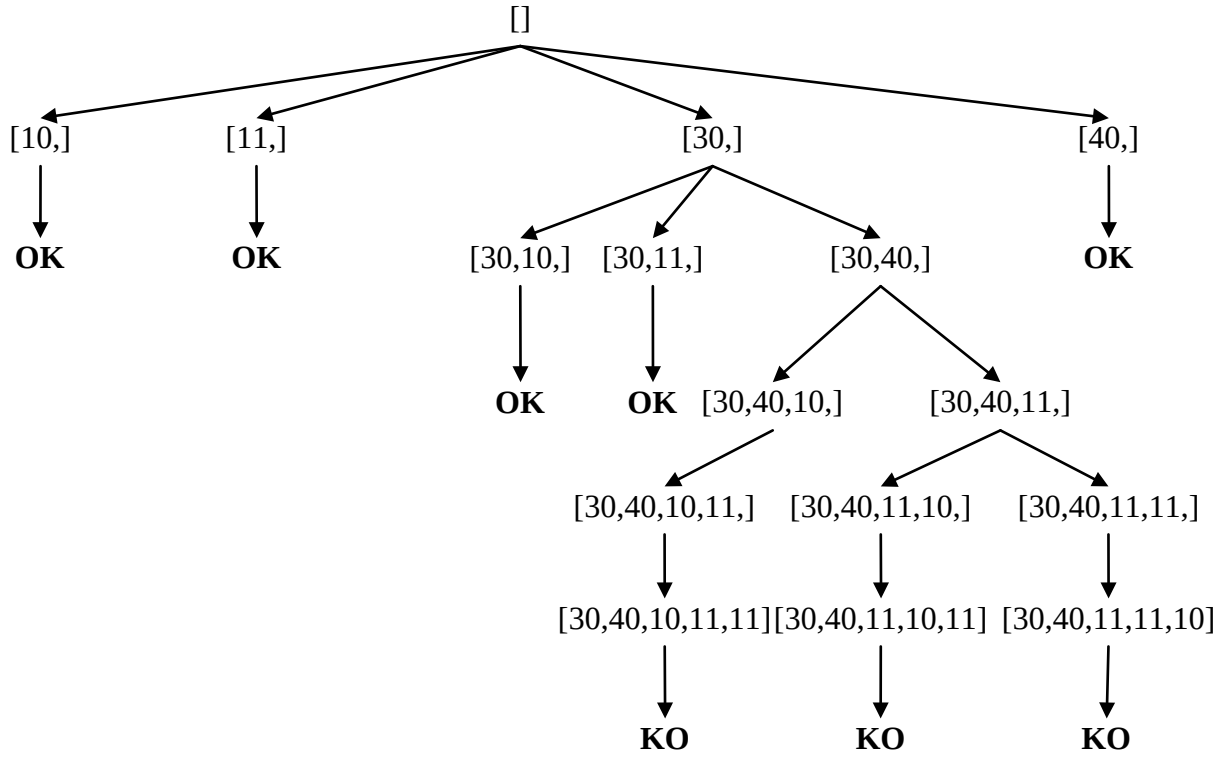
Sembrerebbe che per risolvere il problema occorra esaminare *tutte* le 60 *disposizioni* (di cinque oggetti di cui due eguali): in realtà si può abbreviare molto il ragionamento.

Se il primo elemento della disposizione è 11 o 10, allora sicuramente i primi due (almeno) entrano nel primo contenitore: dei tre rimanenti sicuramente due (almeno) entrano nel secondo contenitore (e rimane al più uno escluso).

Se il primo elemento è 40 questo va nel primo contenitore e, degli altri quattro, tre vanno nel secondo: anche in questo caso c'è al più un escluso.

Se il primo elemento è 30 allora, se il secondo *non* è 40, è facile verificare che almeno due elementi vanno nel primo contenitore e sicuramente almeno due nel secondo.

Le disposizioni che "mettono in crisi" il metodo sono quelle che cominciano con 30 seguito da 40: per esse avanzano sempre *due* pacchi.
Se si immagina di costruire le disposizioni per passi successivi, a partire dalla "radice" vuota si ottiene la seguente figura che riassume il precedente ragionamento.



APPROFONDIMENTO
Come utile esercizio si consideri il seguente metodo di riempimento dei contenitori (in sostituzione di quello riportato in premessa:

- disporre i pacchi "in fila" in un ordine iniziale *qualunque*;
- aprire il primo contenitore;
- inserire uno dopo l'altro nel primo contenitore i pacchi della fila, fino a quando si incontra un pacco che non entra più nel primo contenitore;
- scartare il pacco, chiudere il primo contenitore e aprire il secondo contenitore;
- inserire uno dopo l'altro nel secondo contenitore i pacchi rimasti nella fila, scartando quelli che eventualmente non entrano.

È facile vedere che il metodo funziona *sempre*; infatti quando si scarta il pacco di peso x, detti k_i quelli già inseriti nel primo contenitore, si ha:

$$\sum_i k_i + x > \frac{P}{2}.$$

Dunque i restanti pacchi hanno la somma dei pesi strettamente minore di P/2 e quindi entrano tutti nel secondo contenitore.
N.B. Si osservi come una "piccola" differenza nel metodo di riempimento cambi radicalmente il risultato.

Questi due esempi fanno parte di una vasta classe di problemi detta "*Bin packing*"; tutti i problemi di questa classe sono accomunati dal fatto che un certo numero di oggetti deve essere suddiviso tra contenitori (*bin*) di eguale capienza; può variare (per esempio essere minimo) il numero di contenitori (negli esempi visti sono esattamente due), la possibilità (o meno) di riordinare gli oggetti, il

numero dei parametri che descrivono gli oggetti (e i contenitori: negli esempi visti un solo parametro, il peso), ecc.
Esempi di applicazioni sono numerosi: posizionare dei *file* su dischi di *backup*, distribuire le pubblicità nelle interruzioni di trasmissioni televisive, registrare una biblioteca di pezzi musicali su CD, ritagliare delle figure piane da un certo numero di lastre di plexiglass (o pezze di stoffa), caricare pacchi su una flotta di veicoli da trasporto per la distribuzione, impaginare gli articoli di un giornale, ecc.

ESERCIZIO 10

PROBLEMA

A man went into a bank with exactly £1000 in pound coins. He gave them to a cashier and asked the cashier to put the money into 10 bags in such a way that if he later needed any amount of coins up to £1000, he could lay his hands on that amount without needing to open any of the bags. How did the cashier achieve this?

Put the number of coins in each bag, in descending order, in the list L.

L	

SOLUZIONE

L	[489,256,128,64,32,16,8,4,2,1]

COMMENTI ALLA SOLUZIONE

Il punto fondamentale è capire che il problema è un semplice esercizio di aritmetica binaria; dato che non si possono aprire i sacchetti, si tratta di poter esprimere un numero (di sterline) da 1 a 1000 con cifre *binarie*, perché ogni cifra (1 o 0) rappresenta rispettivamente la presenza o l'assenza del sacchetto che contiene (un numero di monete pari a) la potenza di due relativa alla posizione della cifra binaria. Per esempio se si desiderano 428 (in base 10) sterline si ha

$$428_{10} = 110101100_2$$

Cioè occorrono, partendo da destra: no il sacchetto da 1
no il sacchetto da 2
il sacchetto da 4,
il sacchetto da 8,
no il sacchetto da 16
il sacchetto da 32,
no il sacchetto da 64
il sacchetto da 128
il sacchetto da 256.

Ciò naturalmente fino a 511 (= 512 – 1 = 2^9 – 1 = 111111111_2). Un (ulteriore) sacchetto deve quindi contenere 1000 – 511 = 489 sterline.

Si noti come i numeri da 1 a 488 e quelli da 512 a 1000 hanno una unica scomposizione in sacchetti.