

GUIDA ALLA SOLUZIONE DEI PROBLEMI - OPS 2020

INDICE DEL DOCUMENTO

0. INTRODUZIONE	pag. 2
1. ALCUNE REGOLE DI SCRITTURA	pag. 4
2. LA COMPILAZIONE DELLE RISPOSTE	pag. 8
3. PROBLEMI RICORRENTI	pag. 9
4. ELEMENTI DI PSEUDOLINGUAGGIO	pag. 35

0. INTRODUZIONE

Il programma OPS per il 2019-2020 si rivolge, come di consueto, a tre livelli di partecipazione:

- scuola primaria,
- scuola secondaria di primo grado,
- scuola secondaria di secondo grado (primo biennio).

Sono previste *gare a squadre* per tutti i livelli e *gare individuali* per gli ultimi due livelli.

Ogni *gara a squadre* consisterà di norma in 13 problemi ; l'articolazione dei problemi sarà, *usualmente*, la seguente:

1. cinque problemi formulati in italiano e scelti, di volta in volta, tra l'insieme dei “Problemi ricorrenti” (si veda il successivo elenco);
2. sei problemi formulati in italiano e relativi a uno pseudo-linguaggio di programmazione;
3. un problema di comprensione di un testo in lingua italiana;
4. un problema formulato in inglese, di argomento ogni volta diverso (almeno in linea di principio).

Ogni *gara individuale* consisterà di 8 problemi ; l'articolazione dei problemi sarà, *usualmente*, la seguente:

1. quattro problemi formulati in italiano e scelti, di volta in volta, tra l'insieme dei “Problemi ricorrenti” (si veda il successivo elenco);
2. tre problemi formulati in italiano e relativi a uno pseudo-linguaggio di programmazione;
3. un problema formulato in inglese, di argomento ogni volta diverso (almeno in linea di principio).

La difficoltà e la complessità dei problemi saranno commisurate al livello cui tali problemi saranno proposti.

I *Problemi ricorrenti* nelle gare OPS 2019-2020 sono tratti del seguente insieme:

- a) Regole e deduzioni.
- b) Fatti e conclusioni.
- c) Grafi.
- d) *Knapsack*.
- e) Pianificazione.
- f) Crittografia.
- g) Movimenti di un robot.
- h) Sottosequenze.

Il seguito di questo documento si articola in quattro parti.

Il paragrafo 1 illustra brevemente come scrivere i numeri, i *termini* e le *liste* (questi ultimi sono “scritture formali” che possono essere utilizzate nei testi dei problemi proposti e nelle soluzioni richieste).

Il paragrafo 2 elenca le regole generali per compilare le risposte ai problemi: tali regole sono *stringenti*, perché la correzione dei problemi viene fatta (normalmente) confrontando la stringa, proposta come risposta ad un quesito, con la stringa che rappresenta la risposta attesa (o “esatta”).

Il paragrafo 3 riporta esempi dei problemi ricorrenti; tali esempi sono di norma due: uno orientato alla scuola primaria e uno orientato alla scuola secondaria (se è fornito un solo esempio il problema ricorrente è orientato a un solo ordine di scuola). I problemi sono di carattere abbastanza semplice; servono principalmente di riferimento per:

- fissare gli argomenti e la terminologia usata,
- tratteggiare i metodi di soluzione,
- mantenere “sintetico” l'enunciato dei problemi assegnati effettivamente in gara.

L'ultimo paragrafo illustra brevemente gli *elementi di pseudolinguaggio* che saranno utilizzati nei problemi assegnati in gara. La trattazione è divisa in due parti: la prima parte riguarda i tre livelli scolastici, la seconda riguarda *prevalentemente* la scuola secondaria (anche se, nel volgere delle gare, potrà interessare anche la scuola primaria). Seguono esempi di problemi riguardanti lo pseudolinguaggio.

N.B. Si suppone che i partecipanti alle gare abbiano letto il presente materiale e, comunque, l'abbiano a disposizione durante lo svolgimento della gara.

1. ALCUNE REGOLE DI SCRITTURA

Nei testi dei problemi possono comparire: *numeri*, *termini* e *liste*; le seguenti osservazioni sono utili per comprenderne il significato e l'uso corretto.

a.0 I NUMERI

I numeri sono scritti sempre, salvo diverso avviso in notazione decimale.

a.1 I NUMERI INTERI

I numeri interi sono stringhe di cifre che, a sinistra, non iniziano con la cifra "0". Talvolta, *nel testo dei problemi*, se i numeri sono molto lunghi (cioè con almeno cinque cifre), per facilitarne la lettura possono avere un *separatore periodico* ogni tre cifre a partire da destra. Se l'ambito è in *lingua italiana* il separatore è un punto nella parte superiore della riga, ad esempio:

123
1234
12'345
1'234'556'789

Se l'ambito è in *lingua inglese* il separatore è la virgola, ad esempio:

123
1234
12,345
1,234,556,789

N.B. Nella compilazione delle risposte i numeri interi sono *sempre* scritti *senza* separatore periodico e *senza* zeri a sinistra.

a.2 I NUMERI RAZIONALI

I numeri razionali sono scritti con una parte intera e una parte decimale, separate da un marcatore; se l'ambito è in *lingua italiana* il marcatore è la *virgola* nella parte bassa del rigo:

0,23
2,343
45,0

se l'ambito è in *lingua inglese* il marcatore è il *punto* nella parte bassa del rigo:

0.23
2.343
45.0

I numeri razionali non devono iniziare (a sinistra) con uno zero.

A destra lo zero è ammesso se è l'unica cifra a destra della virgola, oppure quando il testo del problema specifica il numero di decimali dopo la virgola. Per esempio $9/5$ è uguale a 1,8; se però il problema richiede esplicitamente due decimali allora si scriverà 1,80.

N.B. Nella compilazione delle risposte il numero *intero* 45 è "diverso" dal numero *decimale* 45,0 (o 45.0 se l'ambito è in lingua inglese): il testo del problema specificherà sempre la natura della risposta richiesta.

a.3 TRONCAMENTO E ARROTONDAMENTO DEI NUMERI RAZIONALI

Le due nozioni sono bene illustrate dai seguenti esempi.

Esempio 1. Supponiamo che, nella soluzione di un problema, il risultato (ottenuto con opportuni calcoli) sia 9,3472; se nella risposta sono richieste, *due cifre decimali arrotondate*: occorre scrivere 9,35; se invece sono richieste *due cifre decimali troncate*, occorre scrivere 9,34.

Es. 2. Supponiamo che, nella soluzione di un problema, il risultato (ottenuto con opportuni calcoli) sia 9,3442; se nella risposta sono richieste, *due cifre decimali arrotondate*: occorre scrivere 9,34; se invece sono richieste *due cifre decimali troncate*, occorre scrivere 9,34.

Altri esempi, sempre riferiti a due cifre decimali, sono raccolti nella seguente tabella.

Numero calcolato	Numero arrotondato	Numero troncato
9,3435	9,34	9,34
9,3402	9,34	9,34
8,9951	9,00	8,99
0,898	0,90	0,89

Come avrete capito nell'operazione di arrotondamento (a due cifre decimali) si considera la terza cifra decimale e :

se tale cifra è 0,1,2,3,4 il numero arrotondato è uguale al numero troncato
 se tale cifra è 5,6,7,8,9 il numero arrotondato è uguale a numero troncato + 0,01

a.4 DATE ED ORE

Il testo del problema e/o le diciture che accompagnano i campi da riempire, specificheranno in maniera puntuale come indicare date e ore.

b.0 I TERMINI

Un *termine* è una scrittura del tipo:

minerale(m1,90,300)
 attività(alfa,4,6)

cioè un *nome* seguito da *uno* o più *argomenti* racchiusi tra parentesi e separati da virgole (se sono più di uno).

N.B. Nella scrittura di un termine non intervengono spazi.

Il *nome* è una stringa di caratteri che inizia con una lettera; gli *argomenti* sono parole, sigle o numeri interi.

b.1 I TERMINI: DEFINIZIONE ED ESEMPI D'USO

In generale, prima dell'uso, ogni termine è definito. La definizione indica il *nome* del termine, gli *argomenti* e il loro *significato*.

Si consideri, ad esempio, la tabella seguente che si riferisce ad un magazzino di minerali: riporta la sigla il peso e il valore di ogni esemplare presente nel magazzino.

minerale		
sigla	valore in euro	peso in kg
m1	213	30
m2	200	35
m3	230	38

Il contenuto della tabella può essere descritto con dei termini; ogni termine è così *definito*:
 minerale(<sigla>,<valore in euro>,<peso in kg>)

La *definizione* mostra il *nome* del termine, il *numero* degli *argomenti* e il loro *significato*. Nel caso delle tabelle il nome del termine è il nome della tabella, il numero degli argomenti è il numero delle colonne e il significato di ogni termine è l'intestazione della colonna. Il contenuto della tabella può essere così descritto:

```
minerale(m1,213,30)
minerale(m2,200,35)
minerale(m3,230,38)
```

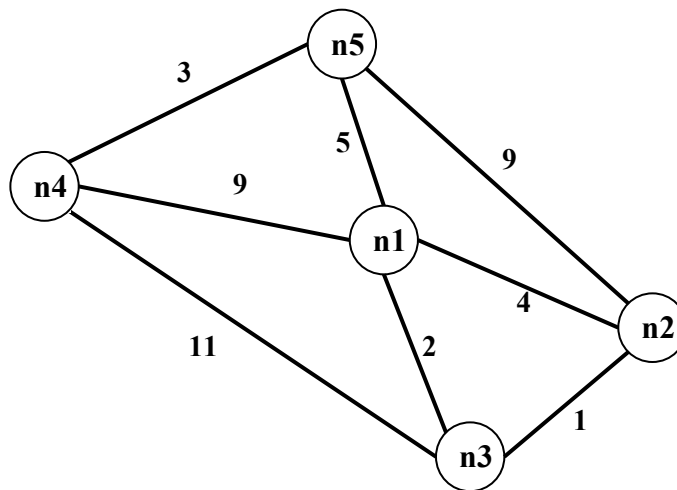
Come ulteriore esempio si consideri la seguente figura che rappresenta un *grafo*; questo è composto da *nodi*, distinti da una sigla, e da *archi* che congiungono i nodi.

Un grafo si può pensare come la rappresentazione schematica di città congiunte da strade.

Un grafo può essere descritto con dei termini; ogni termine è associato a un arco ed è così *definito*:

arco(<nodo1>,<nodo2>,<lunghezza>)

La definizione mostra il nome del termine e il numero degli argomenti (tre): i primi due si riferiscono ai nodi posti agli estremi dell'arco, il terzo descrive la lunghezza dell'arco.



I termini che descrivono il grafo in figura sono:

arco(n1,n2,4)	arco(n1,n3,2)	arco(n1,n4,9)	arco(n1,n5,5)
arco(n4,n5,3)	arco(n4,n3,11)	arco(n2,n3,1)	arco(n2,n5,9)

Grafi più generali sono descritti nel capitolo 3 al punto c).

c.0 LE LISTE

Una *lista* è una scrittura del tipo:

[alfa,beta,gamma]

[]

[a1,2,56,b2,3,67]

cioè una coppia di parentesi quadre che racchiudono un certo numero di *elementi* separati da virgole (se sono più di uno). Una lista può non contenere elementi: in tal caso si dice *lista vuota*.

N.B. Nella scrittura di una lista non intervengono spazi.

Gli *elementi* sono parole, sigle, numeri interi o *altre liste*.

c.1 LE LISTE: ESEMPI D'USO

Si consideri la seguente figura che rappresenta una scacchiera 8×8. Facendo riferimento allo spigolo in basso a sinistra, ogni casella può essere individuata da una coppia di coordinate, scritte come una lista, il cui primo elemento è la “X”, cioè lo spostamento orizzontale e il secondo elemento è la “Y”, cioè lo spostamento verticale.

Per esempio il numero 5 è nella casella [3,2]; la bandiera è nella casella [1,8] e il robot nella casella [8,1].

♞			■				
	■	■			11		
						■	
					12		
		5		■		13	
		■					♚

Per indicare la posizione sulla scacchiera dei quadrati neri si può usare una lista i cui elementi sono le coordinate delle caselle che li contengono; si ottiene, quindi, una lista di liste:

`[[2,5],[3,1],[3,5],[4,8],[5,2],[7,4]]`

Volendo indicare la posizione sulla scacchiera e il valore dei quattro numeri si può usare una lista di quattro elementi che, a loro volta, sono liste; ciascuna lista “interna” è formata dalle coordinate della casella che contiene il numero seguite dal valore:

`[[3,2,5],[6,3,12],[6,5,11],[7,2,13]]`

Per indicare la posizione della freccia ➡ sulla scacchiera si può usare la lista vuota:

`[]`

per significare, appunto, che non compare.

2. LA COMPILAZIONE DELLE RISPOSTE

Si tenga presente che le risposte ai problemi vengono valutate da un sistema automatico che, essenzialmente, confronta stringhe: quindi tali risposte vanno compilate rispettando *esattamente* le indicazioni riportate nel testo dei problemi; inoltre, si devono sempre osservare le seguenti regole generali.

1. In ciascun campo disponibile per la risposta è possibile, *di norma*, digitare, allineato a sinistra, un solo elemento: parola, numero, lista o termine; le parole devono essere scritte *senza accenti*; per i numeri si veda il paragrafo precedente.
2. Talvolta il testo del problema può specificare che la risposta è una frase: in tal caso occorre scriverla con *un solo spazio* tra due parole consecutive.
3. Gli elementi che compongono una lista vanno riportati fra parentesi quadre separati da virgole *senza spazi*: per esempio la lista delle prime tre lettere (minuscole) dell'alfabeto va scritta nel modo seguente: [a,b,c] e non [a, b, c]; i campi, per le risposte che richiedono liste, riportano già le parentesi quadre “esterne”; se gli elementi delle lista sono, a loro volta, liste occorre scrivere le opportune parentesi che li racchiudono.
4. L'elevazione a potenza è indicata da x^n ; per esempio: x^2 deve essere scritto x^2 .
5. Una frazione deve essere scritta su una linea; per esempio: $2/3$.
6. Se nel testo del problema è indicata una particolare modalità di risposta, per esempio:

digitare V per vero o F per falso

 è necessario attenersi strettamente alle istruzioni: in caso contrario il sistema registrerà una risposta errata.
7. Gli allenamenti disponibili sul sito consentono di familiarizzarsi con le regole di compilazione delle risposte.

3. PROBLEMI RICORRENTI

a) REGOLE E DEDUZIONI

PREMESSA

Per risolvere problemi spesso esistono delle regole che, dai dati del problema, permettono di calcolare o *dedurre* la soluzione.

Questa situazione si può descrivere col termine

regola(<sigla>,<lista degli antecedenti>,<conseguente>)

che indica una regola di nome <sigla> che consente di dedurre <conseguente> conoscendo tutti gli elementi contenuti nella <lista degli antecedenti>, detta anche *premessa*. Problemi “facili” possono essere risolti con una sola regola; per problemi “difficili” una sola regola non basta a risolverli, ma occorre applicarne diverse in successione.

Si considerino le seguenti regole (a rigore le regole associate ai seguenti termini):

regola(1,[e,f],b)	regola(2,[m,f],e)	regola(3,[m],f)
regola(4,[b,f],g)	regola(5,[b,g],c)	regola(6,[g,f],c)

Per esempio la regola 1 dice che si può calcolare (o dedurre) **b** conoscendo **e** ed **f** (cioè gli elementi della lista [e,f]); conoscendo **b** ed **f** (cioè gli elementi della lista [b,f]) è possibile dedurre **g** con la regola 4. Quindi, a partire da **e** ed **f** è possibile dedurre prima **b** (con la regola 1) e poi **g** (con la regola 4).

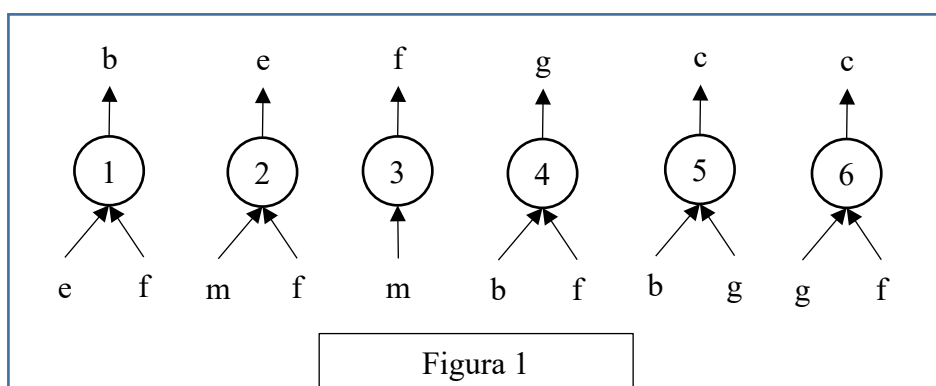
N.B. I due seguenti termini:

regola(1,[e,f],b)
regola(1,[f,e],b)

individuano la stessa regola, che permette di dedurre **b** da **e** ed **f** (o da **f** e da **e**).

Un *procedimento di deduzione* (o deduttivo, o di calcolo) è rappresentato da un *insieme di regole da applicare in sequenza opportuna* per dedurre un certo elemento (incognito) a partire da certi dati: quindi può essere descritto dalla lista delle sigle di queste regole. Il procedimento [1,4] descrive la soluzione del problema: “dedurre **g** a partire da **e** ed **f**”.

Una maniera grafica per rappresentare le regole è quella mostrata nella seguente figura 1: consiste nell’associare un albero (rovesciato) ad ogni regola: la radice (in alto) è il conseguente, le foglie (in basso) sono gli antecedenti.



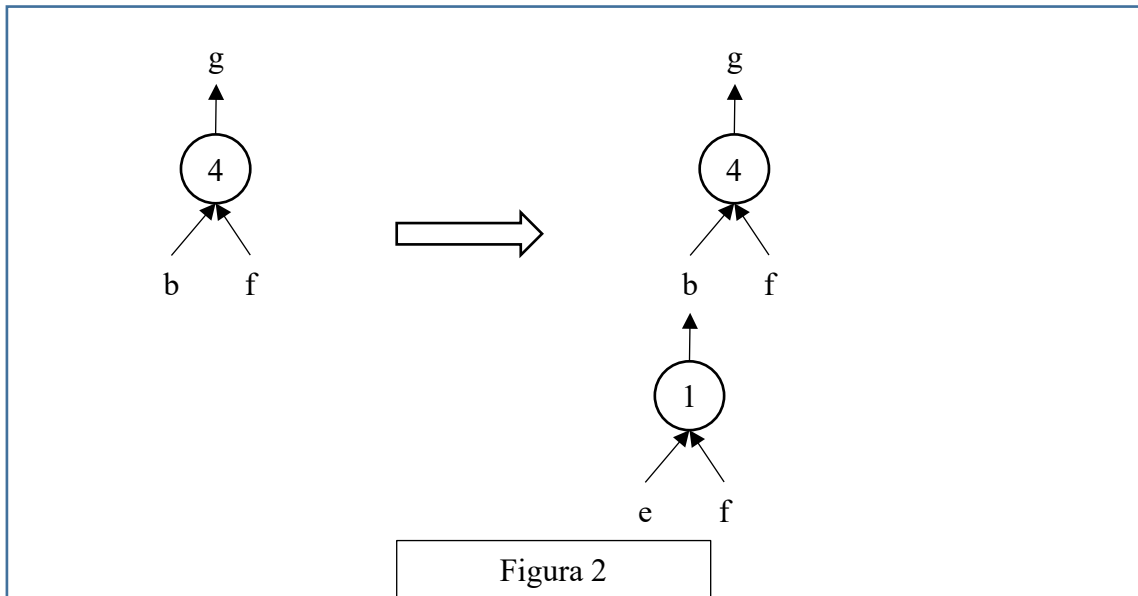
Con questa rappresentazione grafica, risolvere il problema “dedurre **g** a partire da **e** ed **f**” è particolarmente facile; si cerca un “albero” (cioè una regola) che ha come radice l’incognita (cioè **g**): in questo caso ne esiste solo uno che è la regola 4: si veda la figura 2 a sinistra.

Le foglie di questo albero (**b** ed **f**) *non* sono tutte note: quelle note (**f** in questo caso) sono vere e proprie foglie, quelle incognite (**b** in questo caso) vanno considerati come “anelli” a cui “appende-

re” un altro albero; quindi bisogna continuare *sviluppando* la foglia incognita **b**, cioè “*appendendo*” a **b** l’albero rappresentato dalla regola 1, come illustrato nella figura 2 a destra.

Adesso tutte le foglie dell’albero così ottenuto (**e** ed **f**) sono note e il problema è risolto.

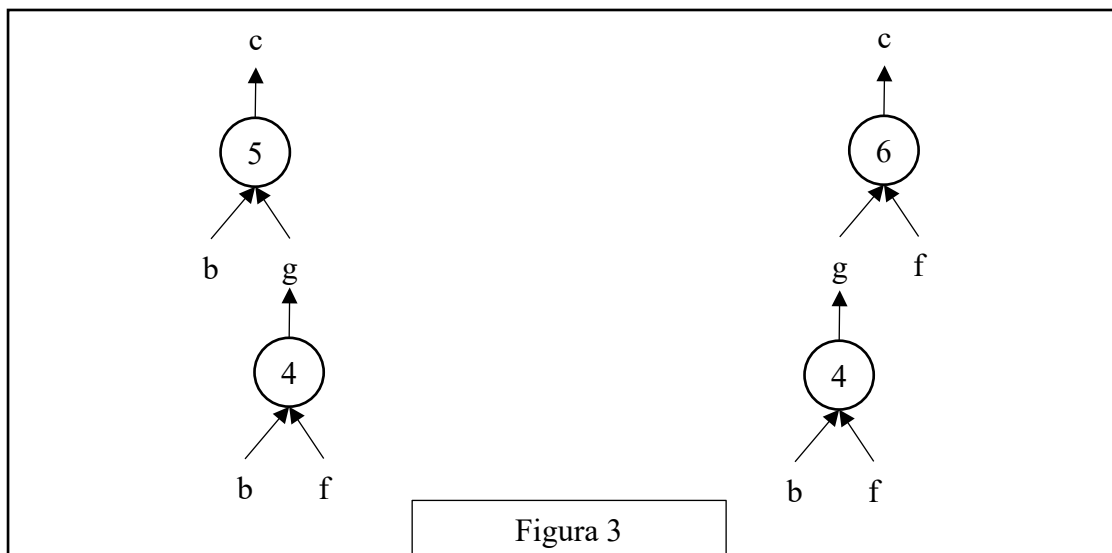
Si può anche dire che un albero le cui foglie sono tutte note rappresenta un procedimento per dedurre la “radice” a partire dalle “foglie”. Per costruire la lista corrispondente occorre *partire dal basso*: prima si applica la regola 1, che utilizza solo i dati; poi si può applicare la regola 4. Il procedimento è quindi (individuato dalla lista) [1,4].



N.B. Nelle liste richieste occorre elencare le sigle delle regole nell’ordine che corrisponde alla sequenza di applicazione: la prima (a sinistra) della lista deve essere la sigla che corrisponde alla prima regola da applicare (che ha come antecedenti solo dati); l’ultima (a destra) deve essere la sigla della regola che ha come conseguente l’elemento incognito da dedurre.

Nella lista non ci sono regole *ripetute* (infatti un procedimento di deduzione è un *insieme* di regole da applicare in opportuna sequenza). L’applicazione di una regola rende disponibile il conseguente da utilizzare (come antecedente) nell’applicazione di regole successive.

La lista associata a un (ben preciso) procedimento si costruisce quindi per passi successivi a partire dal primo elemento che è la sigla della prima regola da applicare; ad ogni passo, se ci fossero più regole applicabili, occorre dare la precedenza (nella lista) a quella con sigla *inferiore* (questo per rendere *unica* la lista associata al procedimento).



N.B. In alcuni casi esistono più procedimenti deduttivi possibili che permettono di ricavare un certo elemento dagli stessi dati, in maniere diverse (cioè con alberi diversi e quindi con insiemi diversi di regole). Per esempio il problema “dedurre **c** a partire da **b** ed **f**” (dalle regole viste sopra) ha due distinti procedimenti risolutivi; gli alberi relativi ai due procedimenti sono mostrati nella seguente figura 3.

Le liste associate sono, rispettivamente, [4,5] e [4,6].

In un procedimento deduttivo, il numero di regole *differenti* coinvolte (e, quindi, anche il numero di elementi della lista corrispondente al procedimento) si dice *lunghezza* del procedimento.

PROBLEMA 1

Siano date le seguenti regole:

regola(1,[b,c],a)	regola(2,[c,d],a)
regola(3,[b,c,d],a)	regola(4,[b,a],f)

Trovare:

1. la sigla N della regola che consente di dedurre **a** da **d** e **c**;
2. la lista L che rappresenta il procedimento per dedurre **f** da **b** e **c**.

Scrivere le soluzioni nella seguente tabella.

N	
L	

SOLUZIONE

N	2
L	[1,4]

COMMENTI ALLA SOLUZIONE

Per rispondere alle due semplici domande è opportuno applicare il metodo *backward*, cioè è opportuno partire dalla incognita (l'elemento che occorre dedurre) e cercarlo nel conseguente delle varie regole.

Per la prima domanda (che chiede di dedurre **a**) si osservi che le regole 1, 2 e 3 hanno tutte come conseguente **a**, e quindi permettono di dedurlo; ma solo la regola 2 ha come premessa $[c,d]$ cioè ha come antecedenti i dati (**d** e **c**) della prima domanda: quindi è quella la regola cercata.

Per rispondere alla seconda domanda, si osservi come una sola regola, la 4, ha come conseguente **f**; ma ha come antecedenti **b** e **a**; però, di questi, solo **b** è noto: quindi, per poter applicare tale regola, occorre dedurre **a**; come già osservato, le regole 1, 2 e 3 hanno tutte come conseguente **a**, e quindi permettono di dedurlo; ma adesso (a differenza della prima domanda) sono noti **b** e **c**: quindi stavolta è la regola 1 quella che deve essere applicata. Nel costruire la lista richiesta si ricordi che il primo elemento di tale lista è la prima regola che deve essere applicata, quindi (tutti) i suoi antecedenti devono essere dati.

PROBLEMA 2

Siano date le seguenti regole:

regola(1,[u,d],c)	regola(2,[q,n],g)	regola(3,[p,q],n)
regola(4,[c,d],z)	regola(5,[u],d)	regola(6,[n,u],m)

Trovare:

- la lista L1 che descrive il procedimento per dedurre **g** a partire da **p** e **q**;
- la lista L2 che descrive il procedimento per dedurre **z** a partire da **u**.

L1	
L2	

SOLUZIONE

L1	[3,2]
L2	[5,1,4]

COMMENTI ALLA SOLUZIONE

Per risolvere questo tipo di problemi 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 i cui antecedenti sono tutti noti (i dati) la soluzione è trovata; altrimenti si cerca una regola i cui antecedenti non sono tutti noti e si continua a cercare regole per derivare gli antecedenti incogniti (che compaiono nella premessa).

Per la prima domanda si verifica immediatamente che **g** compare come conseguente nella regola 2 che ha come antecedenti **q** (dato) e **n** (incognito). Occorre quindi dedurre **n**: questo è conseguente solo della regola 3, che ha come antecedenti **p** e **q** entrambi dati. Quindi la lista L1 è [3,2]; si noti l'ordine delle regole: la prima che compare (a sinistra) nella lista è la prima da applicare e l'ultima trovata col metodo *backward*.

Per la seconda domanda, di nuovo si può osservare che **z** compare come conseguente nella regola 4 che ha come antecedenti **c** e **d**: entrambi incogniti. È facile vedere che **d** può essere dedotto con la regola 5 da **u** (noto); poi noto anche **d**, con la regola 1 si deduce **c**. Quindi la lista L2 è [5,1,4].

N.B. La prima regola che compare nella lista (che rappresenta il procedimento) ha come antecedenti solo dati; la seconda e le successive hanno antecedenti presi dai dati o dagli elementi dedotti mediante le regole che compaiono precedentemente nella lista. L'ultima regola ha come conseguente l'elemento cercato.

b) FATTI E CONCLUSIONI

PREMESSA

Questi problemi trattano di *entità* correlate da fatti; ciascuna entità ha *valori* discreti. Consideriamo, per esempio, le entità “nome”, “cognome”, “età”; se si parla di tre persone, allora il nome può avere (3) valori: Aldo, Giacomo, Giovanni; il cognome può avere i (3) valori Storti, Poretti, Baglio e l’età i (3) valori: 58, 59, 60. Nei problemi vengono enunciati dei fatti e da questi occorre *ragionare* e trarre *conclusioni* per associare opportunamente i valori di nome, cognome ed età.

Per risolvere questi problemi è utile tracciare una tabella.

La tabella si completa esaminando ognuno dei fatti e traendone le conseguenze utilizzando essenzialmente il principio del sillogismo ternario: se $x \in A \cap B$ e $x \in B \cap C$ allora $x \in A \cap C$.

PROBLEMA 1

Alice, Bastiano e Carla abitano nella stessa strada. I loro cognomi sono Rossi, Verdi e Bianchi, e le loro età sono 17, 19 e 20. I nomi, i cognomi e le età sono elencati in ordine casuale (e quindi non si corrispondono ordinatamente).

Dai due fatti elencati di seguito, determinare il nome completo e l’età di ogni persona, riempiendo la successiva tabella.

1. La signorina Rossi è tre anni più vecchia di Carla.
2. La persona di cognome Bianchi ha 19 anni.

NOMI	COGNOMI	ETÀ
Alice		
Bastiano		
Carla		

SOLUZIONE

NOMI	COGNOMI	ETÀ
Alice	Rossi	20
Bastiano	Bianchi	19
Carla	Verdi	17

COMMENTI ALLA SOLUZIONE

Fatto 1 : Rossi è una signorina e ha 20 anni mentre Carla ne ha 17.

Rossi è Alice perché la terza persona (Bastiano) è maschio.

NOMI	COGNOMI	ETÀ
Alice	Rossi	20
Bastiano		
Carla		17

Fatto 2 : Bianchi ha 19 anni. Allora Bianchi è Bastiano e Carla ha cognome Verdi.

Questo permette di completare la tabella.

NOMI	COGNOMI	ETÀ
Alice	Rossi	20
Bastiano	Bianchi	19
Carla	Verdi	17

PROBLEMA 2

Alice, Bastiano e Carla sono pronti per i Grandi Giochi Estivi. Vivono in tre grandi città: Roma, Napoli e Milano; si sono allenati duramente: chi per 3 mesi, chi per 5 mesi chi, addirittura per 7 mesi; praticano il golf, il canottaggio e il ciclismo.

Dai fatti elencati di seguito, determinare dove vive ogni atleta, per quanto tempo si è allenato e che sport pratica, riempiendo la successiva tabella.

1. Chi pratica golf si è allenato per 3 mesi.
2. Bastiano, che non si è allenato per 5 mesi, pratica il ciclismo.
3. Alice si è allenata per due mesi più dell'atleta che sta a Roma.
4. Chi pratica canottaggio non sta a Milano.

NOMI	CITTÀ	SPORT	TEMPO
Alice			
Bastiano			
Carla			

SOLUZIONE

NOMI	CITTÀ	SPORT	TEMPO
Alice	Napoli	canottaggio	5
Bastiano	Milano	ciclismo	7
Carla	Roma	golf	3

COMMENTI ALLA SOLUZIONE

Fatto 1 : Sport : golf -----> Tempo 3 mesi

Fatto 2 : Bastiano pratica il ciclismo e si è allenato per 7 mesi

NOMI	CITTÀ	SPORT	TEMPO
Alice			
Bastiano		ciclismo	7
Carla			

Fatto 3 : Alice non abita a Roma. Alice si è allenata per 5 mesi

Di conseguenza Carla si è allenata per 3 mesi , pratica golf e abita a Roma.

NOMI	CITTÀ	SPORT	TEMPO
Alice			5
Bastiano		ciclismo	7
Carla	Roma	golf	3

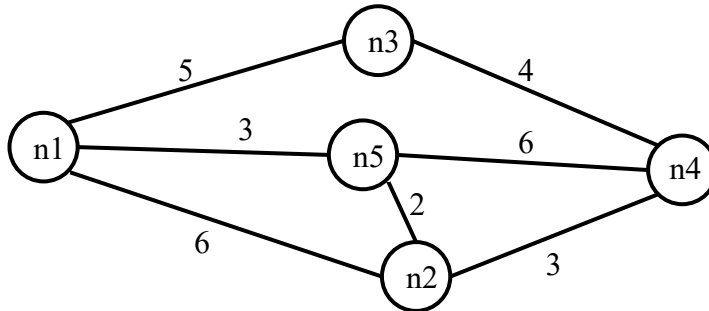
Fatto 4 : Sport : canottaggio -----> Città : Napoli . Sono fatti che riguardano Alice.

Di conseguenza Bastiano abita a Milano e questo completa la tabella.

NOMI	CITTÀ	SPORT	TEMPO
Alice	Napoli	golf	5
Bastiano	Milano	ciclismo	7
Carla	Roma	golf	3

c) GRAFI

Un *grafo* si può pensare come l'astrazione di una carta geografica: per esempio il grafo rappresentato nella figura seguente, descrive i collegamenti esistenti fra alcune (5) città: queste sono rappresentate da *nodi* di nome $n1, n2, \dots, n5$ e i collegamenti sono rappresentati da segmenti tra i nodi, detti *archi*.



A seconda del problema, gli archi possono essere percorsi in entrambe le direzioni (e in questo caso si parla di archi non-diretti) oppure solo in una (archi diretti). Gli archi diretti si rappresentano mediante una freccia, che va dal nodo di partenza a quello di destinazione.

In alcuni problemi, a ogni arco è associata una lunghezza, ovvero un numero, detta anche *peso* dell'arco. Quando gli archi di un grafo hanno un peso, si dice che sono *pesati* e i pesi degli archi vengono rappresentati come numeri scritti vicino alle frecce.

Dunque il grafo rappresentato in figura ha archi non-diretti e pesati.

Un grafo può essere descritto, invece che da una figura, mediante un elenco di termini, ciascuno dei quali definisce un arco tra due nodi del grafo. Nel caso di grafi con archi non pesati, si usano termini con due argomenti. I due argomenti sono i nomi dei nodi connessi dall'arco. Spesso (ma non in tutti i problemi!) si userà il termine “arco” per archi non diretti e “freccia” per archi diretti. Quindi un arco non diretto e non pesato, che connette i nodi x ed y , sarà descritto dal termine **arco(x,y)**, mentre un arco diretto e non pesato, che connette i nodi **Bologna** e **Roma** sarà descritto dal termine **freccia(Bologna,Roma)**.

Nel caso di grafi con archi pesati, è necessario descrivere il peso, oltre che i due nodi collegati. Per questo motivo si useranno termini con 3 argomenti: i primi due sono i nomi dei nodi collegati e il terzo è un numero che rappresenta il valore del peso.

Il grafo rappresentato dalla precedente figura, che ha archi non diretti e pesati, viene quindi descritto dal seguente insieme di termini:

arco($n1,n2,6$)	arco($n1,n3,5$)	arco($n3,n4,4$)
arco($n1,n5,3$)	arco($n2,n4,3$)	arco($n2,n5,2$)
arco($n5,n4,6$)		

Due nodi si dicono *adiacenti* se sono collegati da un arco.

Il numero di archi che “escono” da un nodo si dice *grado* del nodo; per esempio nel grafo in figura, il nodo $n3$ ha grado 2, gli altri hanno grado 3.

Un *percorso* (o *cammino*) tra due nodi del grafo consiste in una sequenza di nodi ciascuno dei quali (tranne l'ultimo) è adiacente con il successivo; un percorso può, quindi essere descritto con una lista di nodi (quelli toccati dal percorso, ordinata dal nodo di partenza al nodo di arrivo). Per esempio, la lista $[n5,n2,n4,n3]$ descrive un percorso dal nodo $n5$ al nodo $n3$; tale percorso ha lunghezza $K = 2 + 3 + 4 = 9$.

Un *ciclo* è un percorso che inizia e termina nello stesso nodo, per esempio $[n5,n2,n1,n5]$.

Un percorso si dice *semplice* se *non* ha nodi ripetuti: un percorso semplice, quindi, non contiene cicli; per esempio $[n5, n2, n4, n3]$ è semplice, mentre $[n5, n2, n1, n5, n2, n4, n3]$ non è semplice perché ha nodi ripetuti.

N.B. Dato un grafo, come quello della precedente figura, è facile scrivere l'insieme di termini che lo descrivono; viceversa, disegnare il grafo da un insieme dei termini è meno ovvio (si vedano i problemi seguenti).

PROBLEMA 1

È dato un grafo descritto dal seguente elenco di archi:

arco($n1, n2, 2$)	arco($n2, n3, 2$)	arco($n3, n1, 5$)
arco($n4, n1, 1$)	arco($n4, n2, 4$)	arco($n4, n5, 3$)

Disegnare il grafo e trovare:

1. la lista L1 del percorso più breve tra $n5$ e $n3$ e calcolarne la lunghezza K1;
2. la lista L2 del percorso più lungo (senza passare più volte per uno stesso nodo) tra $n5$ e $n3$ e calcolarne la lunghezza K2.

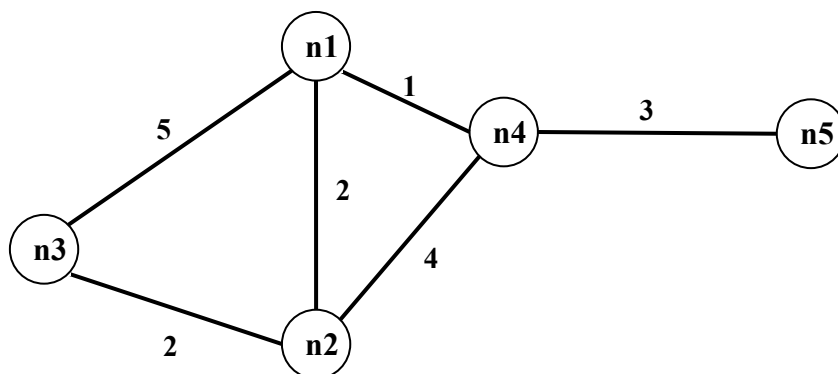
L1	
K1	
L2	
K2	

SOLUZIONE

L1	$[n5, n4, n1, n2, n3]$
K1	8
L2	$[n5, n4, n2, n1, n3]$
K2	14

COMMENTI ALLA SOLUZIONE

Per disegnare il grafo si osservi innanzitutto che vengono menzionati 5 nodi ($n1, n2, n3, n4, n5$); si procede per tentativi: si disegnano i 5 punti nel piano e li si collega con archi costituiti da segmenti: probabilmente al primo tentativo gli archi si incrociano; si cerca poi di risistemare i punti in modo da evitare gli incroci degli archi: spesso questo si può fare in più modi. Da ultimo si riportano le distanze sugli archi, come mostrato dalla figura seguente.



Si noti che le lunghezze degli archi che compaiono nei termini (che rappresentano delle strade) *non* sono (necessariamente) proporzionali a quelle degli archi del grafo (che sono, segmenti di retta).

Per rispondere alle due domande occorre elencare sistematicamente *tutti* i percorsi, che non passino più volte per uno stesso punto, tra n_5 e n_3 :

PERCORSO da n_5 a n_3	LUNGHEZZA
$[n_5, n_4, n_1, n_2, n_3]$	$3+1+2+2=8$
$[n_5, n_4, n_1, n_3]$	$3+1+5=9$
$[n_5, n_4, n_2, n_3]$	$3+4+2=9$
$[n_5, n_4, n_2, n_1, n_3]$	$3+4+2+5=14$

L_1 , K_1 , L_2 , K_2 seguono immediatamente.

PROBLEMA 2

Un grafo (che corrisponde alla rete di strade che collegano delle città) è descritto dal seguente elenco di archi:

$a(n_1, n_2, 13)$	$a(n_2, n_3, 3)$	$a(n_3, n_4, 13)$	$a(n_1, n_4, 3)$
$a(n_4, n_5, 3)$	$a(n_5, n_1, 5)$	$a(n_2, n_5, 7)$	$a(n_3, n_5, 11)$

Disegnare il grafo e trovare:

- la lista L_1 del percorso semplice più breve tra n_1 e n_3 ;
- la lista L_2 del percorso semplice più lungo tra n_1 e n_3 .

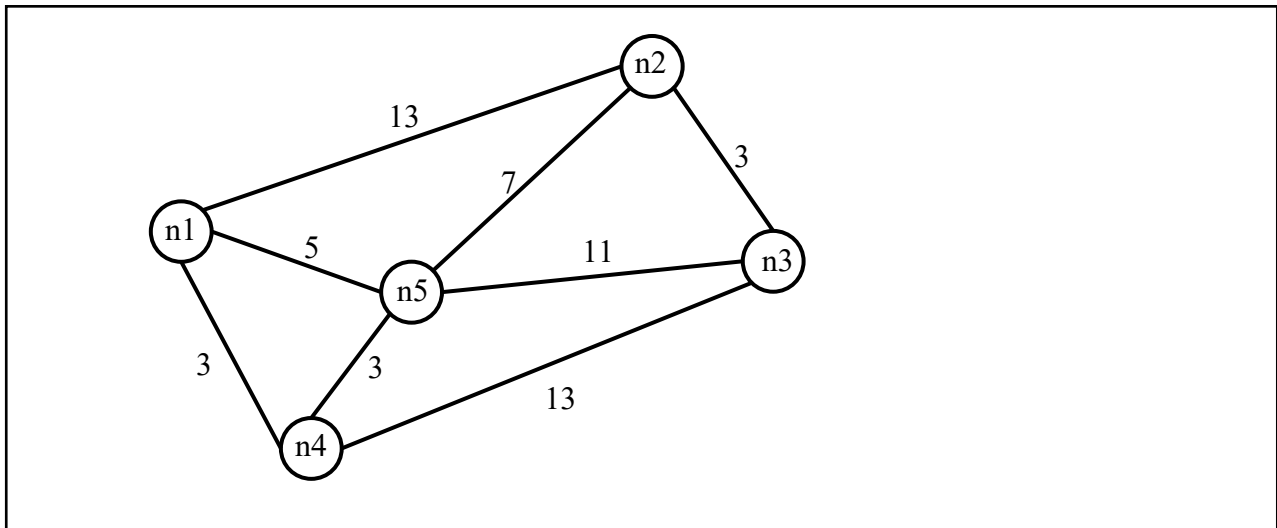
L_1	
L_2	

SOLUZIONE

L_1	$[n_1, n_5, n_2, n_3]$
L_2	$[n_1, n_2, n_5, n_4, n_3]$

COMMENTI ALLA SOLUZIONE

Per disegnare il grafo si osservi innanzitutto che vengono menzionati 5 nodi (n_1, n_2, n_3, n_4, n_5); si procede per tentativi: si disegnano i 5 punti nel piano e li si collega con archi rettilinei: probabilmente al primo tentativo gli archi si incrociano; si cerca poi di risistemare i punti in modo da evitare gli incroci degli archi: spesso questo si può fare in più modi. Da ultimo si riportano le distanze sugli archi, come mostrato dalla figura seguente.



Si noti che le lunghezze degli archi che compaiono nei termini (che rappresentano delle strade) *non* sono necessariamente proporzionali a quelle degli archi del grafo (che sono segmenti di retta).

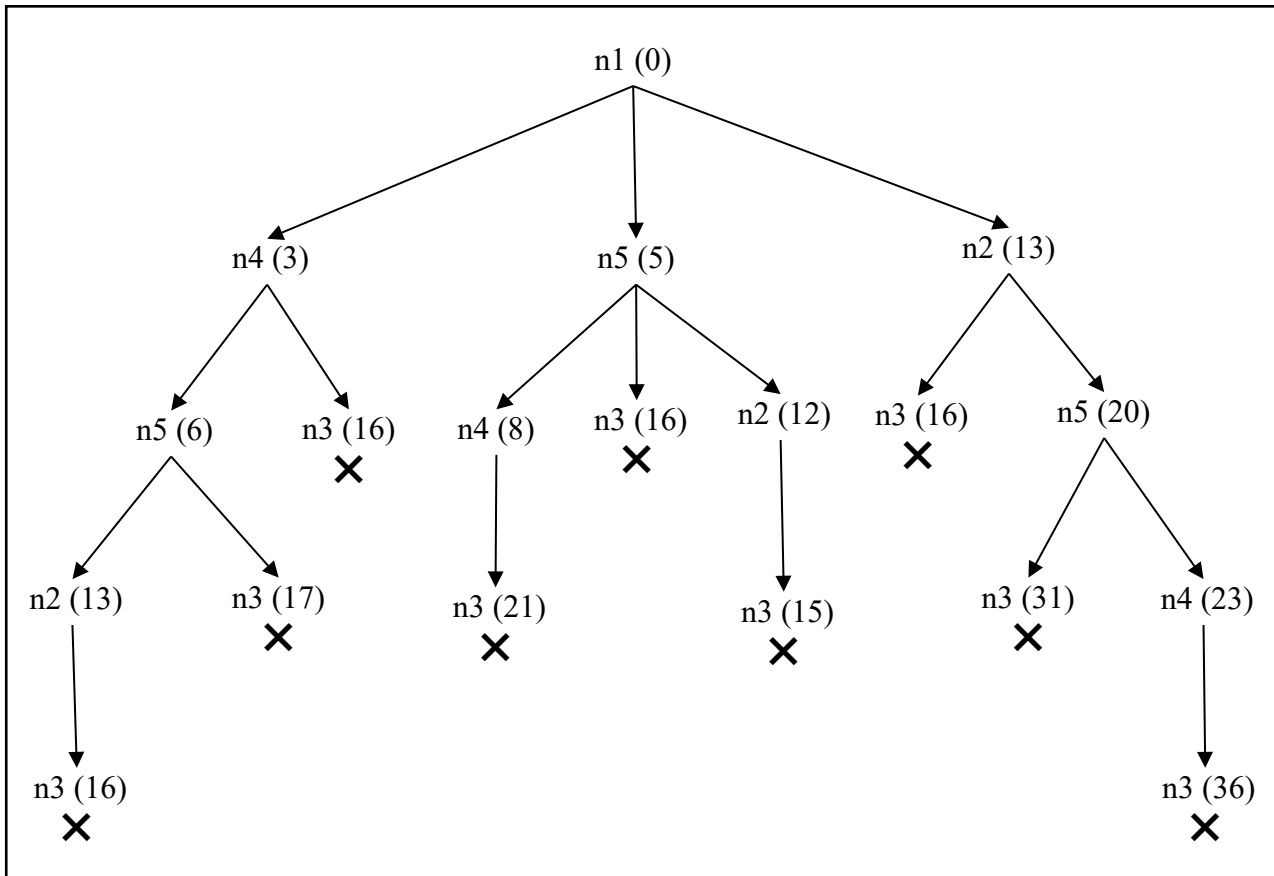
Per risolvere il problema occorre elencare i cammini semplici tra $n1$ e $n3$ (con la loro lunghezza) in maniera *sistematica*, in modo da essere certi di averli presi in esame *tutti*. Da tale elenco la soluzione segue immediatamente.

$[n1, n2, n3]$	16
$[n1, n2, n5, n3]$	31
$[n1, n2, n5, n4, n3]$	36
$[n1, n5, n3]$	16
$[n1, n5, n2, n3]$	15
$[n1, n5, n4, n3]$	21
$[n1, n4, n3]$	16
$[n1, n4, n5, n3]$	17
$[n1, n4, n5, n2, n3]$	16

Si noti che, a partire dal nodo $n1$, si “visitano” i tre nodi adiacenti ($n2, n5, n4$); a partire da ciascuno di questi si costruiscono tutti i cammini che arrivano a $n3$ senza passare per un nodo già visitato (cammini semplici).

Una maniera grafica di chiara evidenza (ma anche concettualmente profonda) è illustrata dalla seguente figura che mostra un albero in cui la radice è il nodo di partenza ($n1$) del grafo, e ogni nodo dell'albero ha tanti figli quanti sono i nodi del grafo a lui collegati purché non compaiono come antenati (nell'albero). Le foglie dell'albero sono il nodo di arrivo ($n3$) (o un nodo da cui non ci si può più muovere perché il nodo successivo sarebbe un antenato). Ad ogni nodo (dell'albero) è stata aggiunta tra parentesi la distanza dalla radice.

Le foglie che individuano i cammini richiesti sono segnate da una **X** (in questo caso tutte).



PROBLEMA 3

L'ufficio tecnico di un piccolo comune deve scegliere dove piazzare dei nuovi lampioni.

Il paese di cui si parla può essere pensato come un insieme di piazzette collegate da strade, descritte dal seguente grafo (dove i nodi sono le piazze e gli archi sono le strade):

arco(n1,n2) arco(n2,n3) arco(n3,n1)
arco(n4,n1) arco(n4,n2) arco(n4,n5)

Ogni lampione illumina la piazza in cui è collocato, le strade da essa uscenti, e le piazze direttamente collegate alla piazza in cui si trova il lampione.

Trovare:

- il numero minimo di lampioni che consente di illuminare tutte le piazze del paese;
- la lista delle piazze (cioè dei nodi del grafo) su cui collocare tali lampioni, in modo che nessuna piazza sia illuminata da due lampioni.

N.B. la lista deve avere gli elementi in ordine crescente ($n1 < n2 < \dots < n5$).

Scrivere la risposta nella seguente tabella.

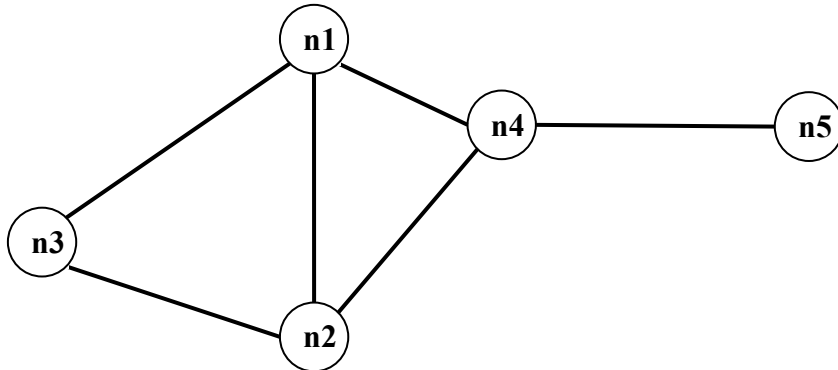
numero minimo di lampioni	
lista delle piazze	

SOLUZIONE

numero minimo di lampioni	2
lista delle piazze	[n3,n4]

COMMENTI ALLA SOLUZIONE

Il grafo è il seguente.



Un metodo risolutivo generale è: generare tutti i sottoinsiemi di vertici, e per ciascuno verificare se permette di “illuminare” tutto il grafo; tra tutti quelli che soddisfano tale requisito, prendere quello/quelli con minor numero di elementi.

Questo metodo, può rapidamente diventare impraticabile: bastano già 5 nodi a renderlo difficoltoso. Tuttavia, poiché occorre individuare un sottoinsieme con minor numero di elementi che soddisfa il requisito, è ovvio che conviene generare ed esaminare i sottoinsiemi per cardinalità crescente. Il metodo quindi diventa:

1. verificare se esiste un vertice che da solo “illumina” tutto il grafo;
2. generare tutte le coppie e verificare se ne esiste una che “illumina” il grafo;
3. generare tutte le triple e verificare se ne esiste una che “illumina” il grafo;
- etc

Inoltre, in molti casi, semplici osservazioni sulla *topologia* del grafo possono portare immediatamente alla soluzione. Nel caso in esame, è chiaro che per coprire n5, la soluzione deve contenere o n4 oppure n5. Converrà senz’altro provare a costruire la soluzione partendo da n4, e una volta verificato che n4 copre tutti i vertici tranne n3, si arriva subito ad una soluzione minimale.

In generale conviene considerare con particolare attenzione i nodi con valenza maggiore.

PROBLEMA4

Un commesso viaggiatore deve visitare un insieme di città, tornando nel punto di partenza e senza passare due volte per la stessa città (ovvero facendo un *tour*).

Le distanze tra le coppie di città, in chilometri, sono date dai seguenti termini (che hanno la struttura arco(<nome di città>, <nome di città>, <distanza>):

arco(C1,C2,10)	arco(C1,C3,4)	arco(C1,C4,9)
arco(C2,C3,6)	arco(C2,C4,5)	arco(C3,C4,4)

Il commesso parte dalla città C1; (disegnare il grafo delle città e) trovare la lunghezza, in chilometri del *tour* più corto.

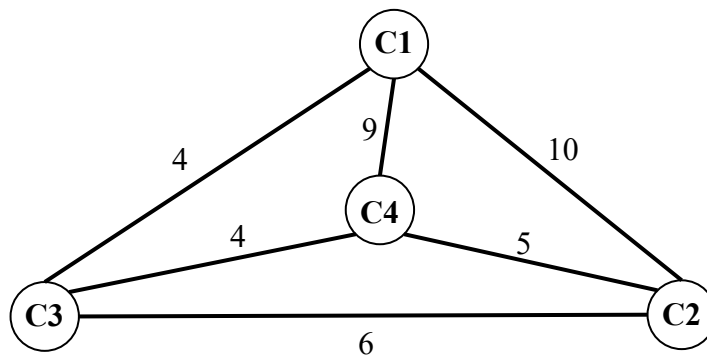
Lunghezza del <i>tour</i> più corto	
-------------------------------------	--

SOLUZIONE

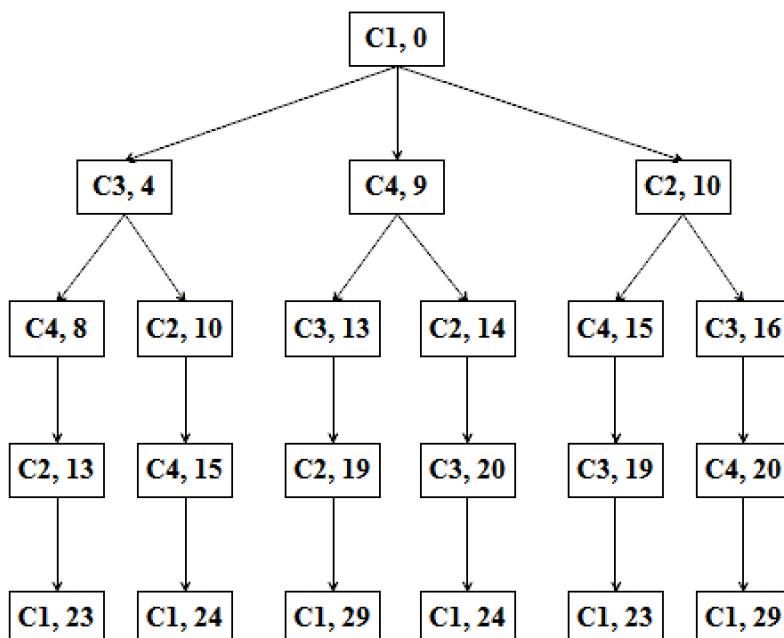
Lunghezza del <i>tour</i> più corto	23
-------------------------------------	----

COMMENTI ALLA SOLUZIONE

Il grafo è il seguente.



Un metodo risolutivo generale è costruire l'albero dei percorsi, da C1 a C1, che toccano una sola volta tutte le città, come nella seguente figura in cui, in ogni nodo è stata riportata la distanza dalla radice.



Si vede che la lunghezza minima di un *tour* è 23 km: tale lunghezza (come tutte le altre) compare due volte perché ogni *tour* ha un suo “simile” percorso in senso inverso.

d) KNAPSACK**PROBLEMA 1**

In un deposito di minerali esistono esemplari di vario peso e valore individuati da sigle di riconoscimento. Ciascun minerale è descritto da una sigla che contiene le seguenti informazioni:

$\text{tab}(\langle \text{sigla del minerale} \rangle, \langle \text{valore in euro} \rangle, \langle \text{peso in Kg} \rangle)$.

Il deposito contiene i seguenti minerali:

$\text{tab}(m1, 15, 35) \quad \text{tab}(m2, 19, 46) \quad \text{tab}(m3, 14, 25) \quad \text{tab}(m4, 10, 12)$

Disponendo di un piccolo motocarro con portata massima di 59 Kg trovare la lista L delle sigle di due minerali diversi che siano trasportabili contemporaneamente con questo mezzo e che abbiano il massimo valore complessivo; calcolare inoltre questo valore V.

N.B. Nella lista, elencare le sigle in ordine (lessicale) crescente; per le sigle usate si ha il seguente ordine: $m1 < m2 < m3 < \dots$.

L	
V	

SOLUZIONE

L	[m2,m4]
V	29

COMMENTI ALLA SOLUZIONE

Per risolvere il problema occorre considerare *tutte* le possibili *combinazioni* di due minerali diversi, il loro valore e il loro peso.

N.B. Le *combinazioni* corrispondono ai sottoinsiemi: cioè sono indipendenti dall'ordine; per esempio la combinazione "m1, m4" è uguale alla combinazione "m4, m1". Quindi per elencarle tutte (una sola volta) conviene costruirle sotto forma di liste i cui elementi sono ordinati, come richiesto dal problema: si veda di seguito.

Costruite le combinazioni occorre individuare quelle trasportabili (cioè con peso complessivo minore o eguale a 59) e tra queste scegliere quella di maggior valore.

COMBINAZIONI	VALORE	PESO	TRASPORTABILI
[m1,m2]	$15+19=34$	$35+46=81$	no
[m1,m3]	$15+14=29$	$35+25=60$	no
[m1,m4]	$15+10=25$	$35+12=47$	si
[m2,m3]	$19+14=33$	$46+25=71$	no
[m2,m4]	$19+10=29$	$46+12=58$	si
[m3,m4]	$14+10=24$	$25+12=37$	si

Dal precedente prospetto la soluzione si deduce facilmente.

N.B. Conviene elencare (costruire) prima tutte le combinazioni che iniziano col "primo" minerale, poi tutte quelle che iniziano col "secondo" minerale, e così via, in modo da essere sicuri di averle considerate tutte.

PROBLEMA 2

In un deposito di minerali esistono esemplari di vario peso e valore individuati da sigle di riconoscimento. Ciascun minerale è descritto da un termine che contiene le seguenti informazioni:

minerale(<sigla minerale>,<valore>,<peso>).

Il deposito contiene i seguenti minerali:

minerale(m1,39,58)	minerale(m2,42,64)	minerale(m3,40,65)
minerale(m4,38,59)	minerale(m5,37,61)	minerale(m6,42,62)

I minerali possono essere spostati con carrelli di diversa portata su cui si possono mettere tre esemplari (diversi).

- Disponendo di un carrello con portata massima di 180 Kg, trovare la lista L1 delle sigle di tre minerali diversi che siano trasportabili contemporaneamente e che abbiano il massimo valore complessivo.
- Disponendo di un carrello con portata massima di 185 Kg, trovare la lista L2 delle sigle di tre minerali diversi che siano trasportabili contemporaneamente e che abbiano il massimo valore complessivo.
- Disponendo di un carrello con portata massima di 200 Kg, trovare la lista L3 delle sigle di tre minerali diversi che siano trasportabili contemporaneamente e che abbiano il massimo valore complessivo.

N.B. Nella lista, elencare le sigle in ordine (lessicale) crescente; per le sigle usate si ha il seguente ordine: m1<m2<m3< ...

L1	
L2	
L3	

SOLUZIONE

L1	[m1,m4,m6]
L2	[m1,m2,m6]
L3	[m2,m3,m6]

COMMENTI ALLA SOLUZIONE

In generale, in problemi di questo tipo, occorre considerare *tutte* le possibili *combinazioni* di tre minerali diversi; in questo caso occorre inoltre, per ognuna, determinare il valore, il peso e il carrello più “piccolo” che la può trasportare.

N.B. Le *combinazioni* corrispondono ai sottoinsiemi: cioè sono indipendenti dall’ordine; per esempio la combinazione “m1, m2, m3” è uguale alla combinazione “m3, m2, m1”. Quindi per elencarle tutte (una sola volta) conviene costruirle sotto forma di liste i cui elementi sono ordinati come richiesto dal problema.

COMBINAZIONE	VALORE	PESO	CARRELLO MIN.
[m1,m2,m3]	121	187	200
[m1,m2,m4]	119	181	185
[m1,m2,m5]	118	183	185
[m1,m2,m6]	123	184	185
[m1,m3,m4]	117	182	185
[m1,m3,m5]	116	184	185
[m1,m3,m6]	121	185	185
[m1,m4,m5]	114	178	180

L2

[m1,m4,m6]	119	179	180	L1
[m1,m5,m6]	118	181	185	
[m2,m3,m4]	120	188	200	
[m2,m3,m5]	119	190	200	
[m2,m3,m6]	124	191	200	L3
[m2,m4,m5]	117	184	185	
[m2,m4,m6]	122	185	185	
[m2,m5,m6]	121	187	200	
[m3,m4,m5]	115	185	185	
[m3,m4,m6]	120	186	200	
[m3,m5,m6]	119	188	200	
[m4,m5,m6]	117	182	185	

Costruite le combinazioni, occorre individuare, per ogni carrello, quella di maggior valore.

e) PIANIFICAZIONE

PROBLEMA 1

Alcuni ragazzi decidono di costruire un ipertesto multimediale sugli avvenimenti significativi della loro regione per la prossima stagione turistica. Per organizzare il progetto, dividono il lavoro in singole attività e, per ciascuna di queste stabiliscono quanti di loro devono partecipare e stimano il tempo per portarla a conclusione. La tabella che segue descrive le attività (indicate rispettivamente con le sigle A1, A2, A3, ...), riportando per ciascuna di esse il numero di ragazzi assegnato e il numero di giorni necessari per completarla.

ATTIVITÀ	RAGAZZI	GIORNI
A1	6	2
A2	4	2
A3	3	3
A4	6	2
A5	4	2
A6	5	1

N.B. Ai fini del problema non è importante conoscere la descrizione delle singole attività.

Le attività devono *succedersi opportunamente* nel tempo perché, per esempio, una attività utilizza il prodotto di altre: quindi esistono delle *priorità* descritte con coppie di sigle; ogni coppia 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.

In questo caso le priorità sono:

[A1,A2], [A1,A3], [A2,A4], [A3,A4], [A4,A5], [A5,A6].

Trovare il numero N di giorni necessari per completare il progetto, tenuto presente che alcune attività possono essere svolte in parallelo e che ogni attività *deve* iniziare prima possibile (nel rispetto delle priorità). Inoltre, trovare il numero massimo RM di ragazzi che lavora contemporaneamente al progetto.

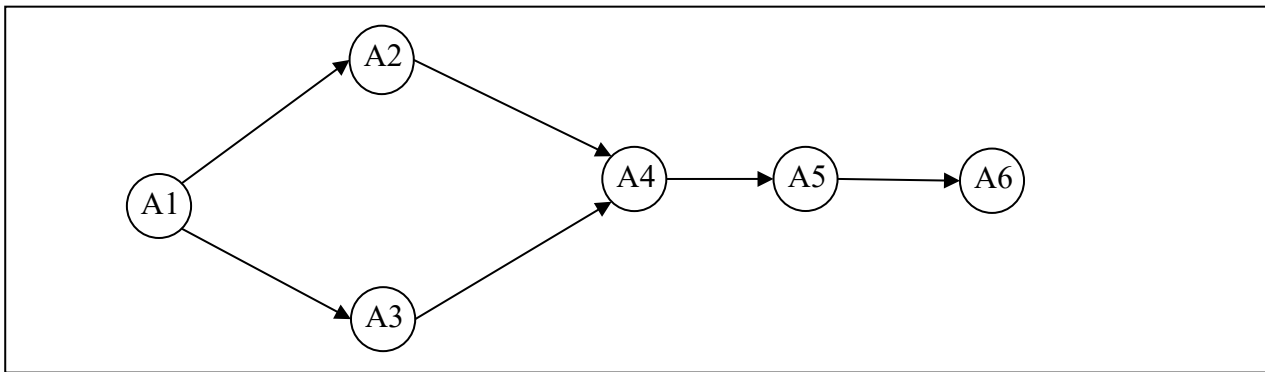
N	
RM	

SOLUZIONE

N	10
RM	7

COMMENTI ALLA SOLUZIONE

Per prima cosa, dai dati sulle priorità occorre disegnare il *diagramma delle precedenze* (diagramma di Pert), cioè il grafo che ha come nodi le attività e come frecce le precedenze: indica visivamente la dipendenza “logica” tra le attività, quindi come si devono susseguire nel tempo.



Per costruire tale grafo (mostrato in figura) si disegnano tanti nodi quante sono le attività (ciascun nodo porta il nome della corrispondente attività).

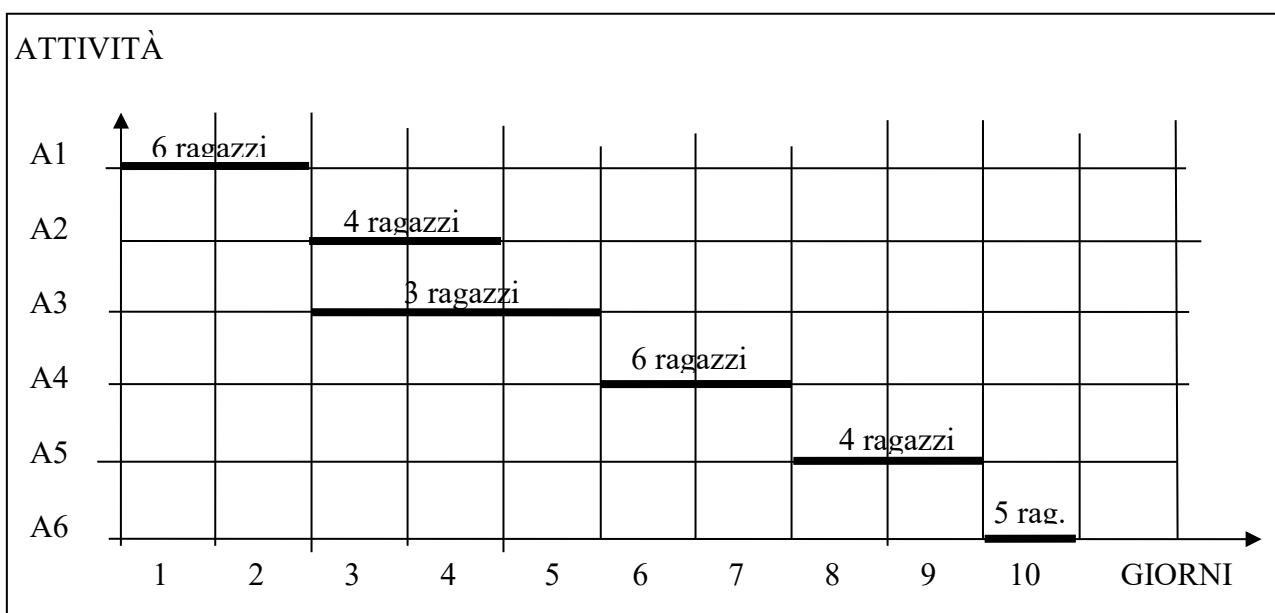
Esiste una attività che compare solo a sinistra nelle coppie che descrivono le priorità: questa è l'attività *iniziale* (in questo caso A1); il nodo corrispondente deve essere disegnato alla sinistra di tutti gli altri.

Esiste una attività che compare solo a destra nelle coppie che descrivono le priorità: questa è l'attività *finale* (in questo caso A6); il nodo corrispondente deve essere disegnato alla destra di tutti gli altri.

Poi per ogni coppia che descrive le priorità si disegna una freccia che connette i nodi coinvolti in quella coppia. Alla fine, in generale, si otterrà un grafo con frecce che si incrociano: tenendo fissi il nodo iniziale e il nodo finale si spostano gli altri nodi per cercare di ottenere (se possibile) un grafo con frecce che non si incrociano (come, appunto, è mostrato in figura).

Poi dal grafo e dalla tabella che descrive le attività, si può compilare il diagramma di Gantt; questo riporta sull'asse verticale le attività (dall'alto verso il basso), sugli assi orizzontali il tempo, in questo caso misurato in giorni. Su ogni asse orizzontale (parallelo a quello dei tempi e in corrispondenza a una attività) è sistemato un segmento che indica l'inizio e la durata della corrispondente attività (e il numero di ragazzi che devono svolgerla).

Così, per esempio, l'attività A1 inizia il giorno 1 e dura due giorni; quando è terminata, il giorno 3 possono iniziare le attività A2 e A3 (che quindi si svolgono parzialmente in parallelo). L'attività A4 può iniziare solamente quando è terminata sia A3 sia A2.



Dal Gantt si vede che il progetto dura 10 giorni e che il numero massimo di ragazzi al lavoro contemporaneamente è 7 (i giorni 3 e 4).

PROBLEMA 2

La tabella che segue descrive le attività di un progetto (indicate rispettivamente con le sigle A1, A2, ...), riportando per ciascuna di esse il numero di persone assegnato e il numero di giorni necessari per completarla.

ATTIVITÀ	PERSONE	GIORNI
A1	6	2
A2	3	3
A3	2	4
A4	6	1
A5	2	3
A6	2	4
A7	3	2
A8	2	4
A9	5	1

Le priorità tra le attività sono:

[A1,A2], [A1,A3], [A1,A4], [A2,A5], [A3,A8], [A7,A9],
[A4,A6], [A6,A7], [A5,A7], [A6,A8], [A8,A9].

Trovare il numero N di giorni necessari per completare il progetto, tenuto presente che alcune attività possono essere svolte in parallelo e che ogni attività *deve* iniziare prima possibile (nel rispetto delle priorità). Inoltre, determinare PM: il *numero massimo* di persone che lavorano contemporaneamente al progetto.

(N.B. PM è anche il *numero minimo* di persone contemporaneamente disponibili necessarie per attuare il progetto così pianificato).

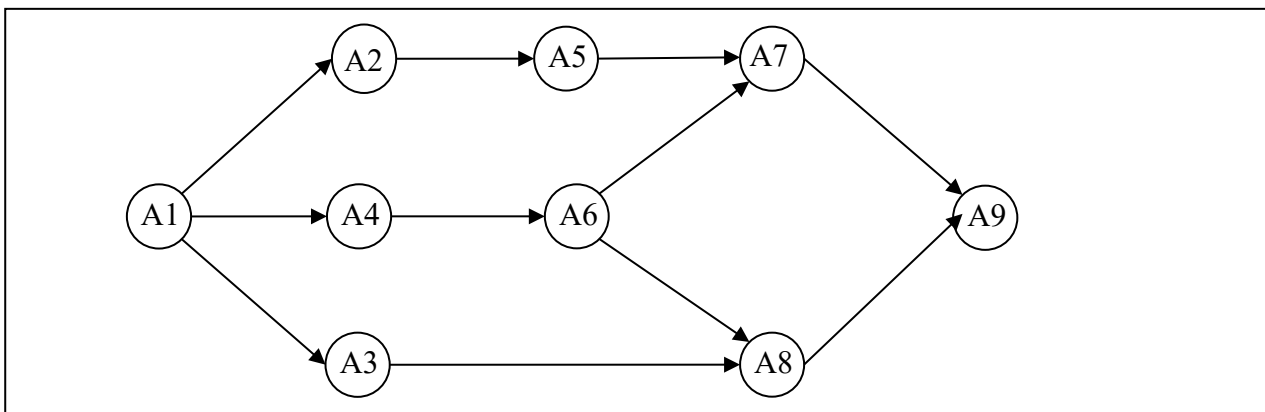
N	
PM	

SOLUZIONE

N	12
PM	11

COMMENTI ALLA SOLUZIONE

Per prima cosa, dai dati sulle priorità occorre disegnare il *diagramma delle precedenze*, cioè il grafo che ha come nodi le attività e come frecce le precedenze.



Tale grafo indica visivamente la dipendenza “logica” tra le attività, quindi come si devono susseguire nel tempo; per costruirlo (come mostrato in figura) si disegnano tanti nodi quante sono le attività (ciascun nodo porta il nome della corrispondente attività).

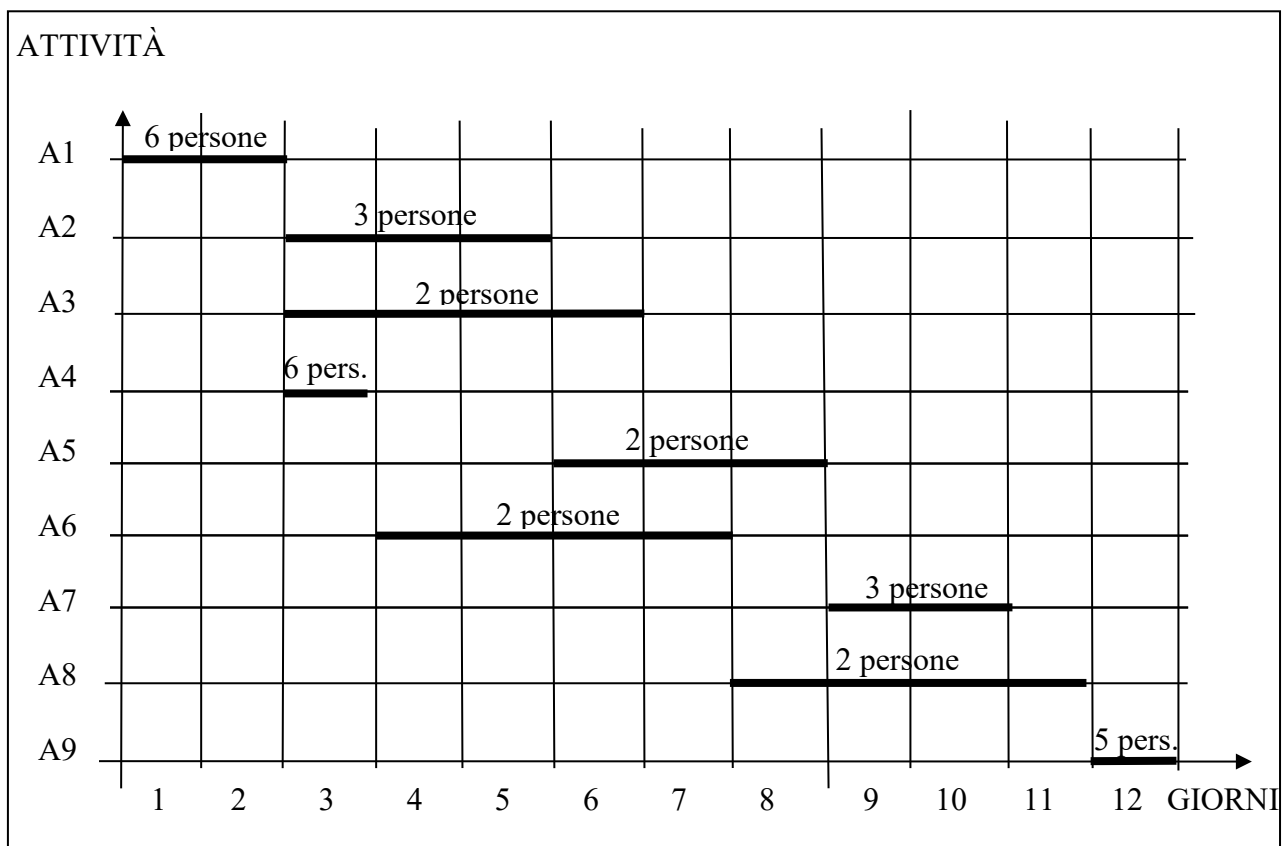
Esiste una attività che compare solo a sinistra nelle coppie che descrivono le priorità: questa è l’attività *iniziale* (in questo caso A1); il nodo corrispondente deve essere disegnato alla sinistra di tutti gli altri.

Esiste una attività che compare solo a destra nelle coppie che descrivono le priorità: questa è l’attività *finale* (in questo caso A9); il nodo corrispondente deve essere disegnato alla destra di tutti gli altri.

Poi per ogni coppia che descrive le priorità si disegna una freccia che connette i nodi coinvolti in quella coppia. Alla fine, in generale, si otterrà un grafo con frecce che si incrociano: tenendo fissi il nodo iniziale e il nodo finale si spostano gli altri nodi per cercare di ottenere un grafo con frecce che non si incrociano.

Poi dal grafo e dalla tabella che descrive le attività, si può compilare il diagramma di Gantt; questo riporta sull’asse verticale le attività (dall’alto verso il basso), sull’asse orizzontale il tempo, in questo caso misurato in giorni. Su ogni linea orizzontale (parallela all’asse dei tempi e in corrispondenza a una attività) è sistemato un segmento che indica l’inizio e la durata della corrispondente attività (e il numero di persone che devono svolgerla): la posizione di tale segmento deve rispettare il diagramma delle precedenze.

Così, per esempio, l’attività A1 inizia il giorno 1 e dura due giorni; quando è terminata, il giorno 3 possono iniziare le attività A2, A3 e A4 (che quindi si svolgono parzialmente in parallelo); inoltre l’attività A7, come altro esempio, può iniziare solamente quando è terminata sia la A5, sia la A6.



Dal Gantt si vede che il progetto dura 12 giorni e che il numero *massimo* di persone al lavoro contemporaneamente è 11 (il terzo giorno): quindi per realizzare il progetto occorre almeno la disponibilità contemporanea di 11 persone.

f) CRITTOGRAFIA

CIFRARIO DI CESARE

È un cifrario a sostituzione monoalfabetica in cui ogni lettera del testo in chiaro è sostituita nel testo cifrato dalla lettera che si trova un certo numero di posizioni dopo nell'alfabeto. Questi tipi di cifrari sono detti anche **cifrari a sostituzione** o **cifrari a scorrimento** a causa del loro modo di operare: la sostituzione avviene lettera per lettera, scorrendo il testo dall'inizio. Il numero di posti di scorrimento è detto chiave. Ad esempio per la chiave 5 abbiamo

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
5	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e

Il testo «olimpiadi di problem solving» viene cifrato come «tqnrfin in uwtgqjr xtqansl»
In modo analogo il testo cifrato con chiave 5 «knsj xjyynrfsf» viene decrittato come «fine settimana»

Altri esempi:

LISTA ORIGINALE	CHIAVE	LISTA CRITTOGRAFATA
[n,a,p,o,l,i]	5	[s,f,u,t,q,n]
[r,o,m,a]	2	[t,q,o,c]
[r,o,m,a]	20	[l,i,g,u]

PROBLEMA

Usando la semplice crittografia di Giulio Cesare:

data la lista [m,i,l,a,n,o] trovarne la corrispondente L1 crittografata con chiave 3;

data la lista [b,o,l,o,g,n,a] trovarne la corrispondente L2 crittografata con chiave 4;

data la lista [w,j,g,j,b,i,v] trovarne la corrispondente L3 crittografata con chiave 5;

L1	
L2	
L3	

SOLUZIONE

L1	[p,l,o,d,q,r]
L2	[f,s,p,s,k,r,e]
L3	[b,o,l,o,g,n,a]

COMMENTI ALLA SOLUZIONE

È sufficiente compilare la tabella in cui la prima riga è il normale alfabeto e le tre successive siano “ruotate” rispettivamente di 3, 4, 5.

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
3	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c
4	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d
5	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e

CIFRARIO MONOALFABETICO GENERICO

Viene fornita una tabella testo in chiaro / testo cifrato in cui le lettere della cifratura sono associate in modo casuale

Ad esempio la tabella seguente

testo in chiaro	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
testo cifrato	e	a	k	p	b	l	r	g	q	c	m	u	h	w	y	d	n	x	z	f	o	i	s	t	v	j

trasforma il testo «problem solving» nel testo cifrato «dxyaubh zyuiqwr»

CIFRARIO POLIALFABETICO DI VIGENÈRE

Il metodo è una generalizzazione del cifrario di Cesare; invece di spostare sempre dello stesso numero di posti la lettera da cifrare, questa viene spostata di un numero di posti variabile ma ripetuto, determinato in base ad una parola chiave, da concordarsi tra mittente e destinatario, e da scrivere ripetutamente sotto il messaggio, carattere per carattere; la chiave è detta *verme*, per il motivo che, essendo in genere molto più corta del messaggio, deve essere ripetuta molte volte sotto questo, come nel seguente esempio: cifrare «olimpiadi di problem solving» con verme CUNEO.

Esempio.

Testo in chiaro	o	l	i	m	p	i	a	d	i	d	i	p	r	o	b	i	e	m	s	o	i	v	i	n	g		
verme	C	U	N	E	O	C	U	N	E	O	C	U	N	E	O	C	U	N	E	O	C	U	N	E	O		
Testo cifrato	q	f	v	q	d	k	u	q	m	r	k	j	e	s	p	n	y	z	w	c	n	p	v	r	u		
		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
chiave 2	C	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b
chiave 20	U	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t
chiave 13	N	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m
chiave 4	E	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d
chiave 14	O	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n

Il testo viene inizialmente compresso in una lunga stringa eliminando gli spazi di separazione tra le parole.

Si procede poi alla suddivisione della stringa in blocchi di lettere grandi come quello della parola CUNEO.

C	o	i	i	l	l	lettere cifrate con chiave di Cesare 2
U	l	a	p	e	v	lettere cifrate con chiave di Cesare 20
N	i	d	r	m	i	lettere cifrate con chiave di Cesare 13
E	m	i	o	s	n	lettere cifrate con chiave di Cesare 4
O	p	d	b	o	g	lettere cifrate con chiave di Cesare 14

A questo punto ha inizio la cifratura :

Si osserva che

la lettera C compare come prima lettera nel cifrario di Cesare con chiave 2

la lettera U compare come prima lettera nel cifrario di Cesare con chiave 20

la lettera N compare come prima lettera nel cifrario di Cesare con chiave 13

la lettera E compare come prima lettera nel cifrario di Cesare con chiave 4

la lettera O compare come prima lettera nel cifrario di Cesare con chiave 14

Di conseguenza le lettere del testo associate nella tabella vanno crittate con il dato cifrario.

Nota : Con questo metodo lettere uguali possono avere cifrature differenti

g) MOVIMENTI DI UN ROBOT

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).

									S				
					P								
→													

Ogni casella può essere individuata da due numeri (interi); per esempio la casella contenente P è individuata da essere nella sesta colonna (da sinistra) e nella terza riga (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 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; lo stato del robot può quindi essere individuato da tre “valori”: due per le coordinate della casella che occupa e uno per indicare il suo orientamento. Per quest’ultimo si possono usare i simboli della stella dei venti: E, S, W, N: per indicare che il robot è rivolto, rispettivamente, a *destra*, in *basso*, a *sinistra*, in *alto* (con riferimento a chi guarda il foglio); lo stato del robot, rappresentato dalla freccia nella figura è [1,1,E].

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,f,a,f,f] fa spostare il robot dalla posizione e orientamento iniziali mostrati in figura fino alla casella P; le caselle via via occupate (quella di partenza e quella di arrivo comprese) sono quelle della lista:

[[1,1],[2,1],[3,1],[4,1],[5,1],[6,1],[6,2],[6,3]].

Stessa casella di arrivo si raggiunge con la lista di comandi [a,f,f,o,f,f,f,f], ma il percorso è diverso ed è descritto dalla lista

[[1,1],[1,2],[1,3],[2,3],[3,3],[4,3],[5,3],[6,3]].

Inoltre, nel primo caso lo stato l’orientamento finale del robot è verso l’alto (stato [6,3,N]), mentre nel secondo caso l’orientamento finale è verso destra (stato [6,3,E]).

PROBLEMA

In un campo di gara il robot si trova nella casella (8,2) con direzione West e deve eseguire la seguente lista di comandi [o,f,f,o,f,f,o,f,f].

Determinare:

1. lo stato S1 in cui si trova il robot prima di aver eseguito tutti i comandi
2. lo stato S2 in cui si trova il robot dopo aver eseguito tutti i comandi

S1	
S2	

SOLUZIONE

S1	[2,8,W]
S2	[4,8,S]

COMMENTO

La direzione è indicata con le iniziali delle parole Nord (alto), Sud (basso), Est (destra) e West (sinistra).

La lista di comandi è [o,f,f,o,f,f,o,f,f]. La posizione iniziale è (2,8) e la direzione iniziale del robot è West. Quindi S1 è la lista [2,8,W].

Per determinare S2, è conveniente visualizzare il percorso, come nella figura che segue (che mostra solo parzialmente il campo di gara, con il valore delle coordinate). Nelle caselle attraversate dal robot è stato inserito un numero. I numeri mostrano l'ordine in cui le caselle sono attraversate.

12								
11								
10		.3	.4	.5				
9		.2		.6				
8		.1		.7				
7								
	1	2	3	4	5	6	7	8

Osservando la figura è semplice determinare la sequenza di comandi che fa compiere tale percorso. Si deve prestare attenzione all'orientamento del robot. Inizialmente il robot si trova in [2,8] con direzione West, ovvero ha stato [2,8,W]. Il primo comando modifica la direzione, portandola a Nord, ovvero trasforma lo stato in [2,8,N]. Il secondo comando fa percorrere un passo lungo la direzione del robot, e quindi lo stato diviene [2,9,N].

Ragionando in modo analogo, si ricostruiscono tutti i movimenti, riassunti nella seguente tabella che mostra, per ogni comando, l'evoluzione dello stato del robot.

Stato di partenza	Comando	Stato di arrivo
[2,8,W]	o	[2,8,N]
[2,8,N]	f	[2,9,N]
[2,9,N]	f	[2,10,N]
[2,10,N]	o	[2,10,E]
[2,10,E]	f	[3,10,E]
[3,10,E]	f	[4,10,E]
[4,10,E]	o	[4,10,S]
[4,10,S]	f	[4,9,S]
[4,9,S]	f	[4,8,S]

h) SOTTOSEQUENZE

PREMESSA

Una sequenza può essere pensata come una lista; per esempio la seguente è una sequenza di numeri interi (non necessariamente distinti):

[15,6,12,18,9,8,10,20,8,4,7]

Una *sottosequenza* è una lista che contiene una parte degli elementi di quella originale, posti nello stesso ordine. Esempi di sottosequenze della lista precedente sono:

[15,18,20,4], [15,6,12,18,7], [9,8,10, 8,4,7]

Non è una sottosequenze della lista precedente la seguente:

[15,18,12,9,8,20,10,4,]

perché gli elementi non compaiono nello stesso ordine di quella data (per esempio 18 e 12 oppure 20 e 10).

In certi casi, data una sequenza, è rilevante determinare sottosequenze con certe proprietà; un caso tipico è determinare la sottosequenza più lunga (strettamente) *decescente*, o quella (strettamente) *crescente*, oppure quella i cui elementi godono di certe proprietà.

N.B. “strettamente” indica che non ci sono elementi ripetuti.

PROBLEMA 1

Un bambino ama molto guardare i cartoni animati in TV. I genitori gli impongono la regola che dopo aver guardato un programma di durata d , può guardare solo programmi di durata minore di d . Al bambino viene data la Guida TV del suo canale preferito che contiene l’elenco di tutti i programmi della giornata con le relative durate. Aiutatelò a scegliere il più grande insieme di programmi che può guardare senza infrangere la regola imposta, se la sequenza delle durate dei programmi (esprese in minuti) è la seguente:

[7,20,12,14,13,15,8,4]

Scrivere la risposta nella casella sottostante.

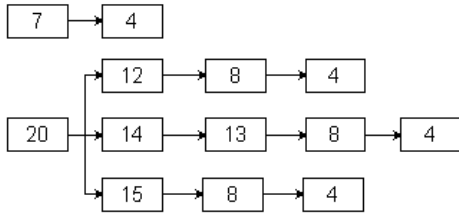
SOLUZIONE

COMMENTI ALLA SOLUZIONE

Il problema chiede di determinare la *sottosequenza decrescente più lunga* estratta dalla lista data. Nei casi semplici come questo si può procedere “ad occhio”; nei casi più complessi occorre essere sistematici ed esaminare *tutte* le possibili sequenze: a partire da ogni elemento della lista si costrui-

sce, in generale, un albero di sequenze (questo processo si può chiamare “sviluppo dell’elemento”); quindi, si otterrà una “foresta”.

Una osservazione che abbrevia di molto il lavoro è la seguente: è inutile sviluppare un nodo se questo compare nello sviluppo di uno già sviluppato; quindi le sequenze da esaminare (e costruire) sono solo le seguenti:



La soluzione ([20,14,13,8,4]) segue immediatamente.

PROBLEMA 2

Si consideri la sequenza di numeri interi:

[8,10,11,12,4,18,6,7,14,10,18,20].

Determinare la più lunga sottosequenza strettamente crescente di numeri pari.

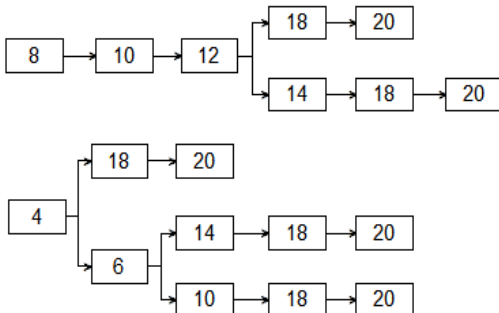
Scrivere la risposta nella casella sottostante.

SOLUZIONE

[8,10,12,14,18,20]

COMMENTI ALLA SOLUZIONE

Si possono individuare 5 sottosequenze, come mostrato dalla figura che segue.



Tale figura è costituita da alberi che, per comodità disegniamo orizzontali; la radice di un albero è un elemento della lista data che non è già comparso in un albero precedente. Ogni nodo dell'albero ha come figli gli elementi della lista che sono contemporaneamente:

1. a lui successivi, cioè a destra nella lista data (si cercano le sottosequenze),
2. pari (si cercano le sottosequenze di numeri pari),
3. di lui maggiori (si cercano le sottosequenze crescenti),
4. minori dei fratelli (si cercano le sottosequenze più lunghe).

(Per il punto 4 si noti che se l'elemento aggiunto non fosse minore dei fratelli, la sequenza a cui appartiene sarebbe sicuramente più corta di quella a cui appartiene il fratello.)

La soluzione [8,10,12,14,18,20] segue immediatamente.

4. ELEMENTI DI PSEUDOLINGUAGGIO

N.B. Un documento più completo (dal titolo “La programmazione, ovvero esprimersi chiaramente) è a disposizione sul sito delle OPS

PRIMA PARTE

1. INTRODUZIONE: LA METAFORA

In una cassettera ci sono dei cassettei individuati dalle lettere A, B, C, D, E, F. In ciascun cassetto ci può essere un foglietto su cui è scritto un numero *intero*. La scrittura

$$C = A+B;$$

significa: “*sommare i numeri scritti sui foglietti dei cassettei A e B, scrivere il risultato su un nuovo foglietto e inserire questo foglietto nel cassetto C, dopo aver eliminato il foglietto (eventualmente) presente in C*”. Se all’inizio nei foglietti di A e B sono scritti rispettivamente i numeri 12 e 7, a operazione eseguita in C si trova un foglietto su cui è scritto 19.

Così, la scrittura:

$$D = C+A-B;$$

significa: “*sommare i numeri scritti sui foglietti dei cassettei C e A, sottrarre alla somma il numero scritto sul foglietto del cassetto B, scrivere il risultato su un nuovo foglietto e inserire questo foglietto nel cassetto D dopo aver eliminato il foglietto (eventualmente) presente in D*”.

N.B. Per brevità diciamo, ad esempio, “*il numero contenuto in C*” invece di “*il numero scritto sul foglietto contenuto in C*”.

2. L’ASPETTO FORMALE

Invece di parlare di cassettei e di numeri scritti su foglietti (come nell’esercizio precedente), si può ricorrere a un’altra descrizione, più astratta.

Le lettere maiuscole A, B, C, ... sono chiamate “*variabili*” (invece di cassettei) e i numeri sui foglietti sono detti “*valori*” di quelle variabili. La sequenza di calcoli dell’ESERCIZIO precedente può essere presentata come la *procedura* seguente.

```
procedure PROVA1;
variables A, B, C, D, E, F integer;
read A, B, C;
D = A+B;
E = C+B-A;
F = A+B-C;
write D, E, F;
end procedure;
```

Con la scrittura “variables A, B, C, D, E, F, integer” si dice che esistono sei cassettei (detti appunto variabili) e che sui foglietti in ognuno dei cassettei può essere scritto (solo) un numero intero.

Con la scrittura “read” si assegnano dei valori a certe variabili (si scrivono dei numeri sui foglietti contenuti in certi cassettei).

Con la scrittura “write” si fa vedere il valore di certe variabili (si leggono i numeri sui foglietti contenuti in certi cassettei).

Se si *esegue* la procedura PROVA1 e alle variabili A, B, C vengono assegnati (read) rispettivamente i valori 5, 8 e 3, in write, per le variabili D, E, F vengono resi visibili rispettivamente i valori 13, 6 e 10 che sono soluzione del problema precedente, di cui PROVA1 è la trascrizione *formale*.

N.B. Ogni riga della procedura si dice *statement* (o *istruzione*).

3. UN NUOVO ESEMPIO

Si ricordi che quando si assegna un nuovo valore a una variabile (cioè si inserisce un nuovo foglietto in un cassetto) l'eventuale valore in essa preesistente viene distrutto (cioè il vecchio foglietto viene buttato e non è più recuperabile).

Ricapitolando, con le lettere A, B, C, ... (o in generale con nomi scritti con lettere maiuscole e numeri) si indicano, in una *procedura*, delle *variabili* che possono acquisire valori mediante

una *istruzione* (o *statement*) “read”,

una *istruzione* (o *statement*) di *assegnazione*.

Si consideri la procedura ESEMPIO seguente, brevemente commentata.

procedura	commento
procedure ESEMPIO1;	inizio della procedura di nome ESEMPIO
variables A, B, C1, D integer;	si dichiara che si usano 4 variabili che assumono valori interi
read A, B;	si acquisiscono dall'esterno valori per le variabili A e B
C1 =A+B;	la variabile C1 acquisisce valore (di una espressione)
D =C1+B-A;	la variabile D acquisisce valore (di una espressione)
A =C1+D;	la variabile A acquisisce (un nuovo) valore (di una espressione)
write A, C1, D;	si rendono disponibile all'esterno i valori delle variabili A, C1, D
end procedure;	fine della procedura

Se in input alle variabili A e B vengono assegnati rispettivamente i valori 5 e 9, in output vengono restituiti i valori 32, 14 e 18 rispettivamente per A, C1, D.

4. L'ALTERNATIVA “if”

Durante la svolgimento di calcoli in una procedura si può porre una “alternativa” decisa dal valore di un *predicato*: se il predicato è *vero* si fanno alcune cose, se è *falso* se ne fanno altre. In una procedura, l'alternativa può essere descritta, per esempio, con la seguente struttura

```
variables A, B, M integer;
```

```
...
```

```
if A > B
```

```
    then M = A;
```

```
    else M =B;
```

```
endif;
```

```
output M;
```

```
...
```

Se per esempio il valore di A è 2 e quello di B è 5, dopo l'alternativa il valore di M è 5 perché essendo $2 > 5$ (falso) viene utilizzata l'istruzione $M=B=5$.

Naturalmente al posto di “A > B” si possono usare altri predicati, costruiti confrontando i valori di certe variabili: per esempio “A = B” oppure “A < B”.

Quando *manca* il ramo “else” (cioè quando occorre fare alcune cose se il predicato è *vero*, ma non si deve fare nulla se è *falso*), si può usare la forma abbreviata del costrutto “if” come nel seguente esempio (che ha lo stesso significato del precedente):

```
variables A, B, M integer;
```

```
...
```

```
M = B;
```

```

if A > B then M = A; endif;
output M;
...

```

SECONDA PARTE

5. LA RIPETIZIONE “for”

In una procedura si può prevedere di eseguire un insieme di operazioni (detto ciclo) un certo numero di volte; nell'esempio che segue il ciclo è fatto di due operazioni che vengono ripetute 4 volte:

```

procedure ESEMPIO2;
variables A, B, K integer;
A = 0;
B = 0;
for K from 1 to 4 step 1 do;
    A = A+K;
    B = B+K×K;
endfor;
write A, B;
end procedure;

```

procedura	commento
<pre> procedure ESEMPIO2; variables A, B, K integer; A = 0; B = 0; for K from 1 to 4 step 1 do; A = A+K; B = B+K×K; endfor; write A, B; end procedure; </pre>	<p>inizio del ciclo (che è ripetuto 4 <i>volte</i> con i valori di K: 1, 2, 3, 4) primo statement del ciclo (A assume via via i valori 1, 3, 6, 10) secondo statement del ciclo (B assume via via i valori 1, 5, 14, 30) segnala che il ciclo arriva fin qui</p>

I valori di write sono: per A 10 (la somma dei 4 valori di K) e per B 30 (la somma dei quadrati dei valori di K).

6. LA RIPETIZIONE “while”

La ripetizione di un gruppo di azioni può essere comandata non solo con la struttura “for” già vista, ma anche con la struttura “while”, illustrata dal seguente esempio.

```

B = 10;
A = 0;
K = 0;
while A < B do;
    K = K + 1;
    A = K × K + A;
endwhile;
write A;

```

Se il predicato $A < B$ è vero, il ciclo viene ripetuto; quando diventa falso si passa alla esecuzione della istruzione successiva a “endwhile”. In questo caso il valore di B rimane fisso a 10, mentre quello di A cambia dopo ogni iterazione assumendo i seguenti valori: 1, 5, 14. Dopo la terza

iterazione il valore di A non è più minore di quello di B e il ciclo si arresta: in output si ha quindi 14.

7. VARIABILI REALI

Oltre a valori interi le variabili possono contenere valori razionali, cioè numeri “con la virgola”: in questo contesto, però, si userà sempre il “.” come separatore decimale. Le variabili di questo tipo si dicono “float”. Corrispondentemente alle variabili di tipo float si usano le costanti di tipo float: si scrivono col punto decimale seguito da almeno una cifra, come in

5.45
5.0
45362.9877

Il seguente è un esempio di procedura che usa variabili float.

procedura	commento
<pre>procedure ESEMPIO3; variables A, B, C integer; variables TF, SF, R float; B =2; A =6; TF =2.0; SF =5.0; C =A/B; R = SF/TF; write C, R; end procedure;</pre>	<p>dichiarazione di variabili intere dichiarazione di variabili razionali assegnazione della costante intera B assegnazione della costante intera A assegnazione della costante razionale TF assegnazione della costante razionale SF calcolo e assegnazione alla variabile intera C calcolo e assegnazione alla variabile float R risulta C =3 e R =2.5</p>

Si ricorda che le normali operazioni aritmetiche sono indicate con +, −, ×, / (talvolta con ÷). Spesso è usato anche l’elevamento a potenza, col simbolo ^. Per esempio $2^3 = 8$; A^B : se A ha valore 2 e B ha valore 3, il risultato dell’espressione è 8; se A ha valore 2.0 e B ha valore 3, il risultato dell’espressione è 8.0.

7. VARIABILI STRINGA

Oltre a valori numerici le variabili possono contenere *stringhe*, cioè sequenze di caratteri. Le variabili di questo tipo si dicono “string”. Corrispondentemente alle variabili di tipo string si usano le costanti di tipo string: si scrivono come sequenze di caratteri racchiuse tra apici; per esempio:

‘alpha’
‘Giuseppe’
‘1200’
‘’
‘ ’

N.B. La (costante) stringa ‘1200’ *non* è il numero intero 1200: in particolare non si può sommare o sottrarre; la costante ‘’ è la stringa *vuota*; la costante ‘ ’ è la stringa *spazio*;
Consideriamo solo tre operazioni tra stringhe: la concatenazione, l’estrazione e la determinazione della lunghezza.

I seguenti sono esempi di procedure che usano variabili string e intere.

procedura	commento
<pre> procedure ESEMPIO4; variables L, integer; variables AS, BS, CS, DS string; BS = 'Aprile'; L = ? BS; AS = 'mese di ' & BS; CS = (1,3) BS; DS = (4,L) BS; end procedure;</pre>	<p>dichiarazione di variabile intera dichiarazione di variabili stringhe</p> <p>assegnazione di costante string assegnazione di valore intero e calcolo della lunghezza della stringa; l'operazione ? calcola la lunghezza del valore (stringa) di BS</p> <p>assegnazione di valore stringa, ottenuto concatenando una costante e (il valore di) una variabile: AS vale 'mese di Aprile': l'operazione & è il concatenamento</p> <p>assegnazione di valore string: l'operazione (1,3) estrae dalla stringa che è il valore di BS la sottostringa che va dal carattere 1 (primo) al carattere 3 (terzo); il valore di CS è 'Apr'</p> <p>assegnazione di valore string: l'operazione (4,L) estrae dalla stringa che è il valore di BS la sottostringa che va dal carattere 4 (quarto) al carattere finale (L vale la lunghezza della lista); il valore di DS è 'ile'</p>

procedura	commento
<pre> procedure ESEMPIO5; variables L, J, J1 integer; variables AS, BS, string; BS = 'Aprile'; L = ? BS; for J from 1 to L step 1 do; AS = (J,J) BS; write AS; endfor; end procedure;</pre>	<p>dichiarazione di variabile intera dichiarazione di variabili stringhe assegnazione di costante string assegnazione di valore intero e calcolo della lunghezza della stringa; l'operazione ? calcola la lunghezza del valore (stringa) di BS che è 6</p> <p>ripetizione L volte assegnazione di valore string: l'operazione (J,J1) estrae dalla stringa che è il valore di BS ogni volta un carattere (in successione) AS è di volta in volta: 'A', 'p', 'r', 'i', 'l', 'e'</p>

Per il confronto delle stringhe sono disponibili i predicati: = e ≠.

Seguono esempi di problemi che riguardano lo pseudolinguaggio.

PROBLEMA1

Si consideri la seguente procedura PROVA1.

```

procedure PROVA1;
variables A, B, C, D integer;
read A, B;
C = A + B;
D = A × B;
A = C+B;
B = (A+B) ×(A- B);
write C, D, A, B;
end procedure;
```

I valori in input sono: 4 per A, 2 per B; determinare i valori di output di A, B, C, D e scriverli nella seguente tabella.

A	
B	
C	
D	

SOLUZIONE

A	8
B	60
C	6
D	8

COMMENTI ALLA SOLUZIONE

Il problema si risolve eseguendo passo passo le operazioni indicate dalla procedura.

Si noti che i valori di certe variabili cambiano più volte: per esempio le variabili A e B acquisiscono un primo valore nello *statement* input e successivamente cambiano valore con le ultime due operazioni.

PROBLEMA2

Si consideri la seguente procedura PROVA2.

```

procedure PROVA2;
variables A, B, M, N, K integer;
read A;
M =0;
N =0;
for K from 1 to 10 step 1 do;
    read B;
    if A > B      then M = M + A;  endif;
    if A < B      then N = N + A;  endif;
endfor;
write M, N;
end procedure;

```

I valori di input per A è 5 e per B sono rispettivamente: 9, 3, 7, 2, 8, 5, 1, 4, 4, 5. Determinare i valori di output.

M	
N	

SOLUZIONE

M	25
N	15

COMMENTI ALLA SOLUZIONE

Basta eseguire, passo a passo, le operazioni indicate: in N va la somma di tante volte il valore di A quante volte questo è più piccolo di (quello di) B (5 volte); in M va la somma di tante volte il valore di A quante volte questo è più piccolo di (quello di) B (3 volte).

PROBLEMA3

Si consideri la seguente procedura (scritta in maniera sintatticamente scorretta: il simbolo X non è definito).

```
procedure P1;
variables A, B, C, D integer;
D = 0;
read A, B, C;
D = A + B + C + X;
write D;
end procedure;
```

Trovare, tra le variabili dichiarate nella procedura, il nome da sostituire a X per ottenere in output 21 per D se i valori in input sono 2 per A, 5 per B e 7 per C.

nome della variabile da sostituire a X	
--	--

SOLUZIONE

nome della variabile da sostituire a X	C
--	---

COMMENTI ALLA SOLUZIONE

Poiché A, B, C valgono rispettivamente 2, 5 e 7, occorre che il risultato di

$$A + B + C + X$$

cioè

$$2 + 5 + 7 + X$$

valga 21; questo richiede che il valore di X sia 7, cioè quello di C.

Una altra maniera di procedere alla soluzione è di esaminare tutti i possibili casi: poiché sono dichiarate quattro variabili: A, B, C, D, allora si possono esaminare i risultati delle quattro possibili sostituzioni:

l'espressione	$A + B + C + A$	vale	$2 + 5 + 7 + 2 = 16$
l'espressione	$A + B + C + B$	vale	$2 + 5 + 7 + 5 = 19$
l'espressione	$A + B + C + C$	vale	$2 + 5 + 7 + 7 = 21$
l'espressione	$A + B + C + D$	vale	$2 + 5 + 7 + 0 = 14$

La soluzione segue immediatamente.

PROBLEMA4

Si consideri la seguente procedura PROVA4.

```
procedure PROVA4;
variables J, L, C integer;
variables A, B string;
read A;
C = 0;
L = |?A;
for J from 1, step 1 to L do;
```

```

        B = |(J,J) A;
        if B = 'a'      then C = C + 1;  endif;
    endfor;
    write C;
end procedure;

```

I valori di input per A è: '9, a, b, 2, ac, 5, a, 4, 4a, 5b', Determinare il valore di output per C

C	
---	--

SOLUZIONE

C	4
---	---

COMMENTI ALLA SOLUZIONE

Basta eseguire, passo a passo, le operazioni indicate: in C va il numero di volte che la lettera 'a' compare nella stringa data in input. Si noti che

|?A

(il numero di caratteri di A) vale 31: naturalmente si contano anche le virgole e gli spazi.