OLIMPIADI di PROBLEM SOLVING

Informatica e pensiero computazionale nella scuola dell'obbligo

Alcune regole generali:

--le fasi di gara

--le scadenze

--la modalità di qualificazione

--i materiali

--la compilazione delle risposte

--le strategie

LICEO SCIENTIFICO e CLASSICO STATALE "G.Peano-S.Pellico"
Via Monte Zovetto, 8 – C.so G. Giolitti, 11 – 12100 Cuneo
tel. 0171 692906 – fax 0171 435200 – c.f. 80009910045
www.liceocuneo.it – mail: liceopeanopellico@gmail.com
Sez. staccata: Via Mazzini, 3 – 12100 Cuneo

--che cosa sono

OLIMPIADI di PROBLEM SOLVING
Informatica e pensiero computazionale nella scuola dell'obbligo

Il Ministero dell'Istruzione, dell'Università e della Ricerca, per l'**anno scolastico 2019-2020**, propone le "**Olimpiadi di Problem Solving. Informatica e pensiero algoritmico nella scuola dell'obbligo**" rivolte all'intero ciclo della scuola dell'obbligo.

La competizione si propone di:
- stimolare la crescita delle competenze di problem solving e valorizzare le eccellenze presenti nelle scuole;
- favorire lo sviluppo e la diffusione del pensiero computazionale;
- promuovere la diffusione della cultura informatica come strumento di formazione nei processi educativi (metacompetenze);
- sottolineare l'importanza del pensiero computazionale come strategia generale per affrontare i problemi, come metodo per ottenere la soluzione e come linguaggio universale per comunicare con gli altri;
- stimolare l'interesse a sviluppare le capacità richieste in tutte le iniziative attivate per la valorizzazione delle eccellenze;
- integrare le esperienze di coding in un riferimento metodologico più ampio che ne permetta la piena valorizzazione educativa.

--che cosa sono

OLIMPIADI di PROBLEM SOLVING
Informatica e pensiero computazionale nella scuola dell'obbligo

...che cosa propongono

...materiale disponibile

...cosa bisogna sapere e saper fare

...

--le fasi di gara

OLIMPIADI di PROBLEM SOLVING
Informatica e pensiero computazionale nella scuola dell'obbligo

Destinatari: gli studenti del primo biennio.

Le competizioni si articolano in **tre fasi** (istituto, regionale e nazionale) precedute da un periodo di allenamento e si svolgono: **a squadre** costituite da quattro allievi e **individuale.**

Sono previste 4 prove di Istituto da novembre a febbraio, seguite dalla fase regionale e nazionale.

OLIMPIADI di PROBLEM SOLVING
Informatica e pensiero computazionale nella scuola dell'obbligo

--le date

Fase di istituto:
Prove a squadre: 5 novembre
4 dicembre
13 gennaio
12 febbraio

Prove individuali: 7novembre
6 dicembre
16 gennaio
14 febbraio

La **fase regionale** si svolgerà il **19 marzo 2020**

OLIMPIADI di PROBLEM SOLVING
Informatica e pensiero computazionale nella scuola dell'obbligo

--le iscrizioni

--entro lunedì 4 novembre

--mail di riferimento: opspeanopellico@gmail.com

--materiale per prepararsi ed esercitarsi

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).

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.

Altra tipologia:
elementi di pseudo-linguaggio di programmazione.

OLIMPIADI di PROBLEM SOLVING

Informatica e pensiero computazionale nella scuola dell'obbligo

Edizione 2016-2017

REGOLE E DEDUZIONI/1

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.

REGOLE E DEDUZIONI/2

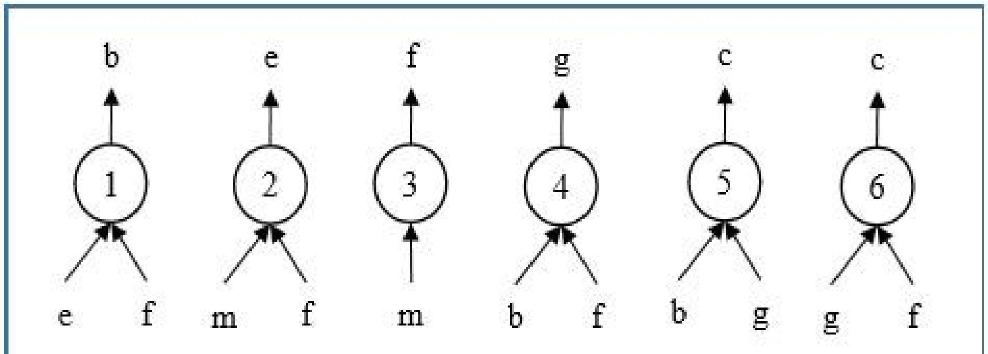


Figura 1

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).

REGOLE E DEDUZIONI/3

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:

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

.

L1	
L2	

OLIMPIADI di PROBLEM SOLVING

Informatica e pensiero computazionale nella scuola dell'obbligo

Edizione 2016-2017

REGOLE E DEDUZIONI/4

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.

FATTI E CONCLUSIONI/1

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 $x \in A \cap C$.

Olimpiadi di Problem Solving
Informatica e pensiero computazionale nella scuola dell'obbligo
Edizione 2016-2017

FATTI E CONCLUSIONI/2

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

FATTI E CONCLUSIONI/3

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

OLIMPIADI di PROBLEM SOLVING
Informatica e pensiero computazionale nella scuola dell'obbligo
Edizione 2016-2017

FATTI E CONCLUSIONI/4

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			

FATTI E CONCLUSIONI/5

SOLUZIONE

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

Olimpiadi di Problem Solving
Informatica e pensiero computazionale nella scuola dell'obbligo
Edizione 2016-2017

FATTI E CONCLUSIONI/6

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 : cannottaggio ------> 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

OLIMPIADI di PROBLEM SOLVING

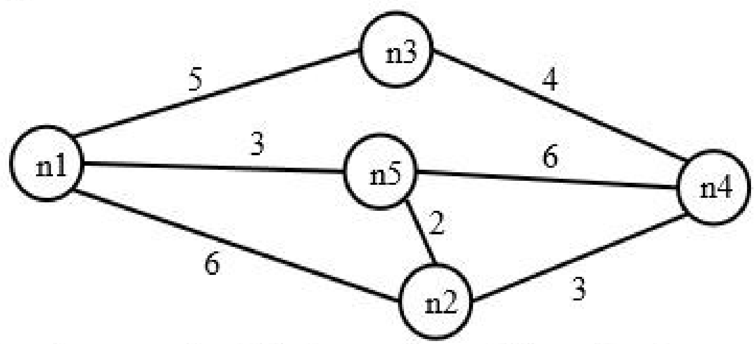
Informatica e pensiero computazionale nella scuola dell'obbligo

Edizione 2016-2017

GRAFI/1

PREMESSA

Un *grafo* si può pensare come l'astrazione di una carta geografica: per esempio il seguente grafo descrive i collegamenti esistenti fra alcune (5) città: queste sono rappresentate da *nodi* di nome n1, n2, …, n5 e i collegamenti sono rappresentati da segmenti tra i nodi, detti *archi*. A ogni arco è associata una lunghezza, come illustrato in figura.



Questo grafo può essere descritto da un elenco di termini, ciascuno dei quali definisce un arco tra due nodi del grafo con la indicazione della relativa distanza in chilometri:

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 *valenza* del nodo; per esempio nel grafo in figura, il nodo n3 ha valenza 2, gli altri hanno valenza 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.

OLIMPIADI di PROBLEM SOLVING
Informatica e pensiero computazionale nella scuola dell'obbligo
Edizione 2016-2017

GRAFI/2

PROBLEMA 2
Un grafo (che corrisponde alla rete di strade che collegano delle città) è descritto dal seguente elenco di archi:

a(n1,n2,13)	a(n2,n3,3)	a(n3,n4,13)	a(n1,n4,3)
a(n4,n5,3)	a(n5,n1,5)	a(n2,n5,7)	a(n3,n5,11)

Disegnare il grafo e trovare:

1. la lista L1 del percorso semplice più breve tra n1 e n3;

2. la lista L2 del percorso semplice più lungo tra n1 e n3.

L1	
L2	

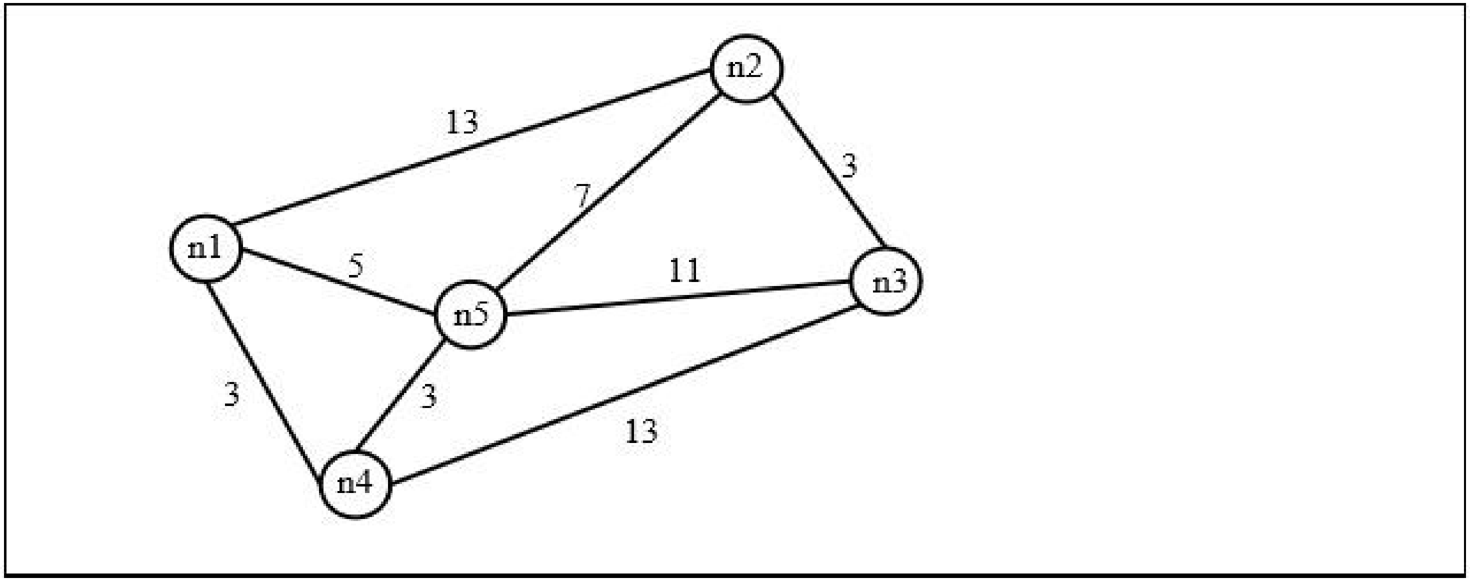
GRAFI/3

SOLUZIONE

L1	[n1, n5, n2,n3]
L2	[n1, n2, n5, n4, n3]

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 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.



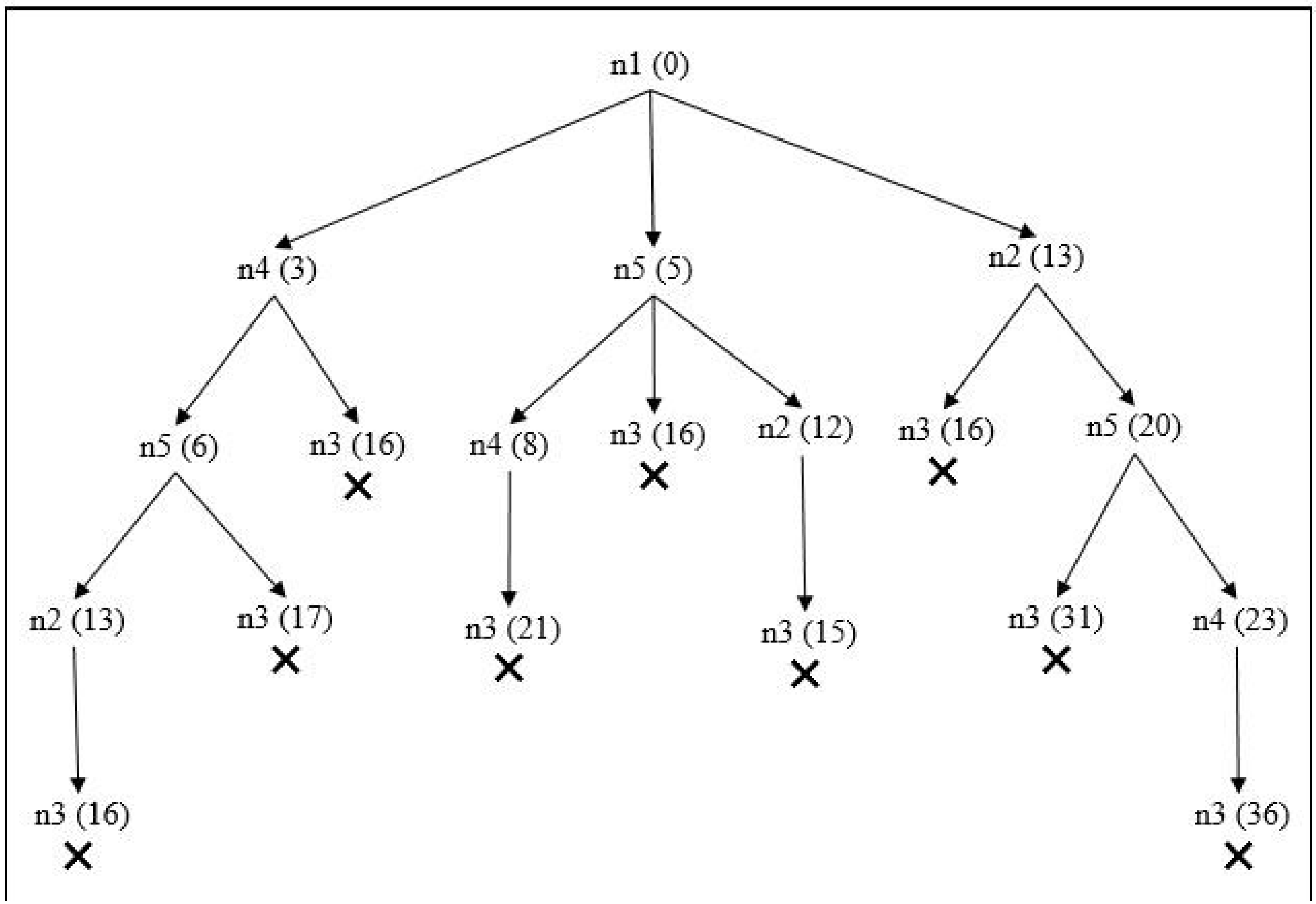
Si noti che le lunghezze degli archi che compaiono nei termini (che rappresentano delle strade) *non* sono necessariamente proporzionali a quelle degli archi del grafo (che sono segmenti di retta).

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.

GRAFI/4

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 ✕ (in questo caso tutte).



Olimpiadi di Problem Solving
Informatica e pensiero computazionale nella scuola dell'obbligo
Edizione 2016-2017

KNAPSACK/1

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:

tab(<sigla del minerale>, <valore in euro>, <peso in Kg>).

Il deposito contiene i seguenti minerali:

tab(m1,15,35) tab(m2,19,46) tab(m3,14,25) 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< … .

L	
V	

KNAPSACK/2

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 la 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.

OLIMPIADI di PROBLEM SOLVING
Informatica e pensiero computazionale nella scuola dell'obbligo
Edizione 2016-2017

KNAPSACK/3

PROBLEMA:

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 L1 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	

KNAPSACK/4

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	L2
[m1,m3,m4]	117	182	185	
[m1,m3,m5]	116	184	185	
[m1,m3,m6]	121	185	185	
[m1,m4,m5]	114	178	180	
[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	

PIANIFICAZIONE/1

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:

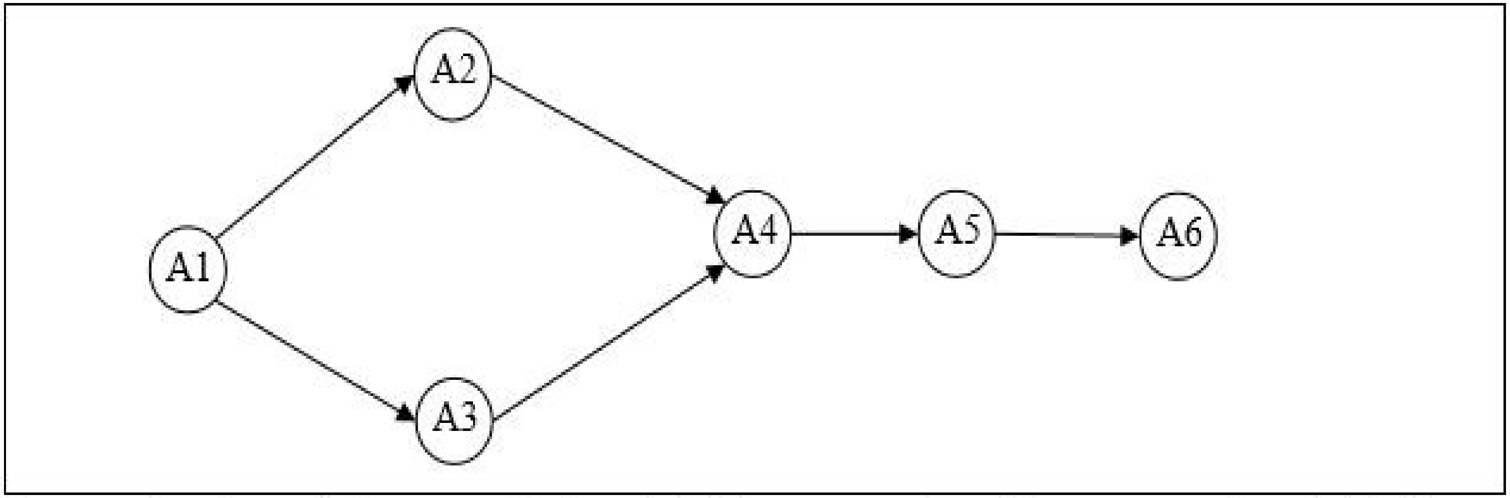
[A1,A2], [A1,A3], [A2,A4], [A3,A4], [A4,A5], [A5,A6].

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.

PIANIFICAZIONE/2

SOLUZIONE

N	10
RM	7



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).

OLIMPIADI di PROBLEM SOLVING

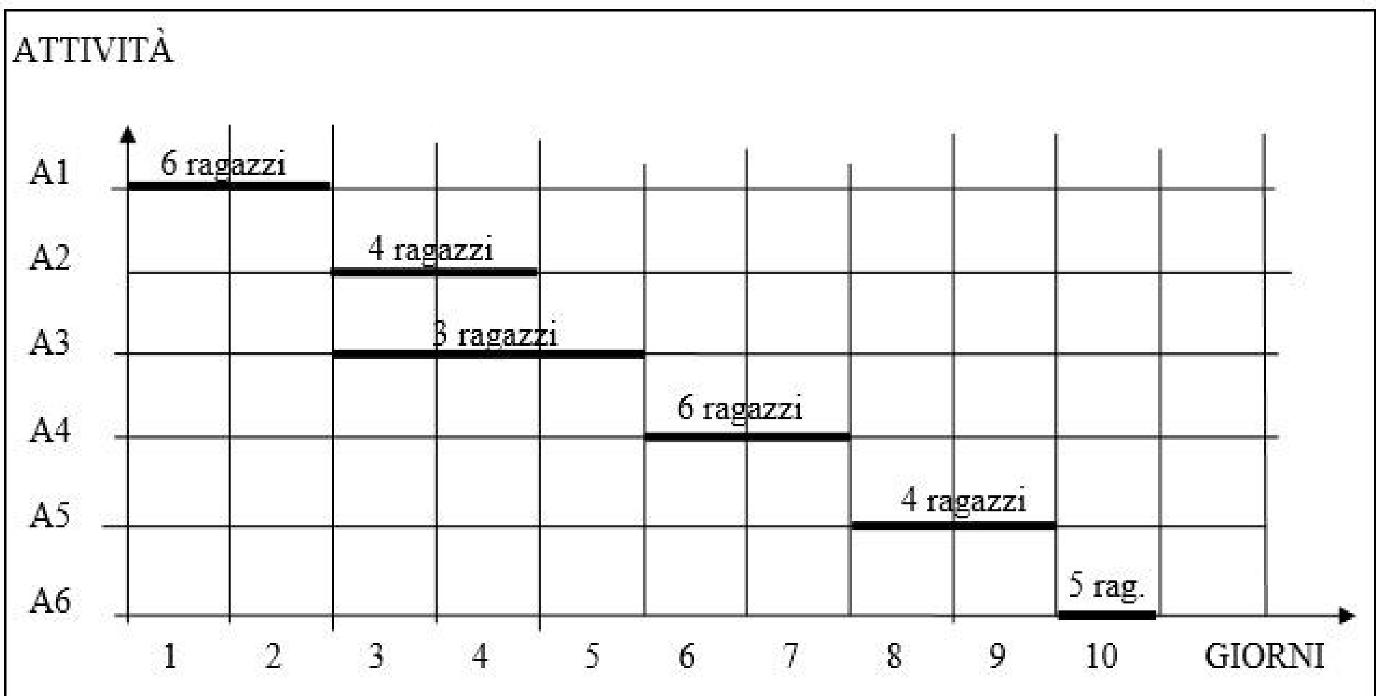
Informatica e pensiero computazionale nella scuola dell'obbligo

Edizione 2016-2017

PIANIFICAZIONE/3

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.



PIANIFICAZIONE/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	

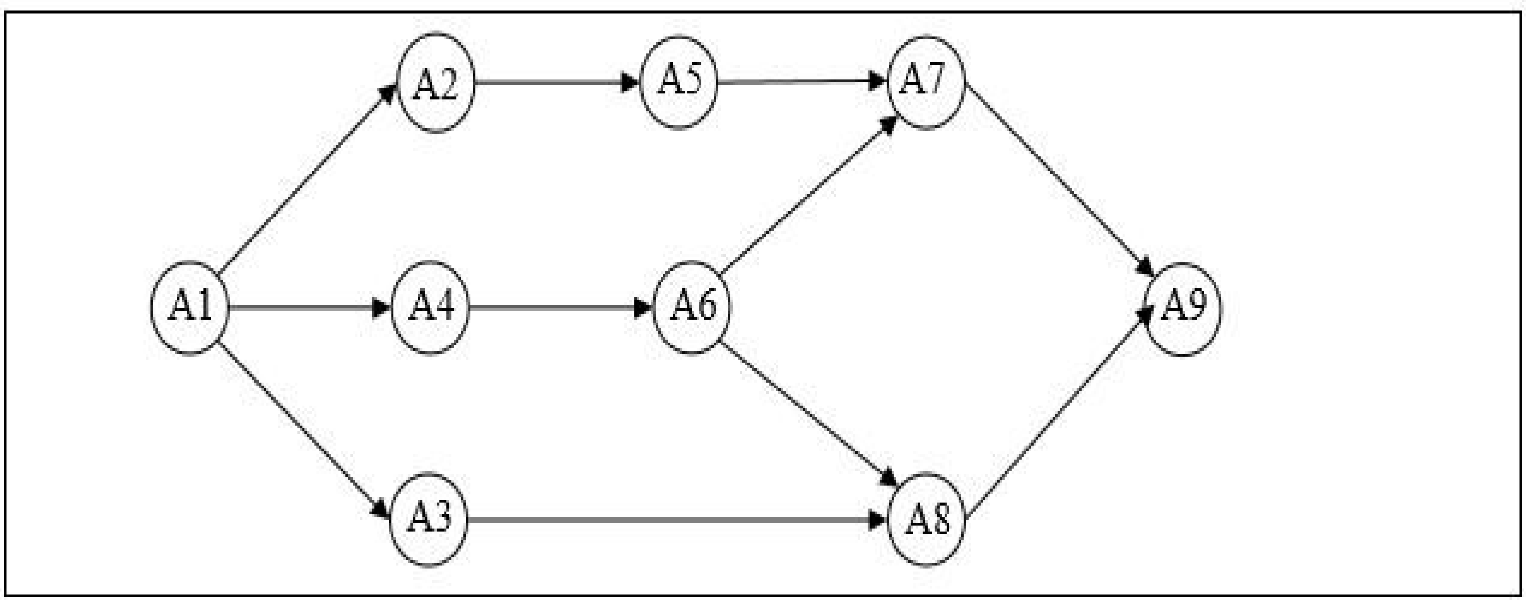
PIANIFICAZIONE/5

SOLUZIONE

N	12
PM	11

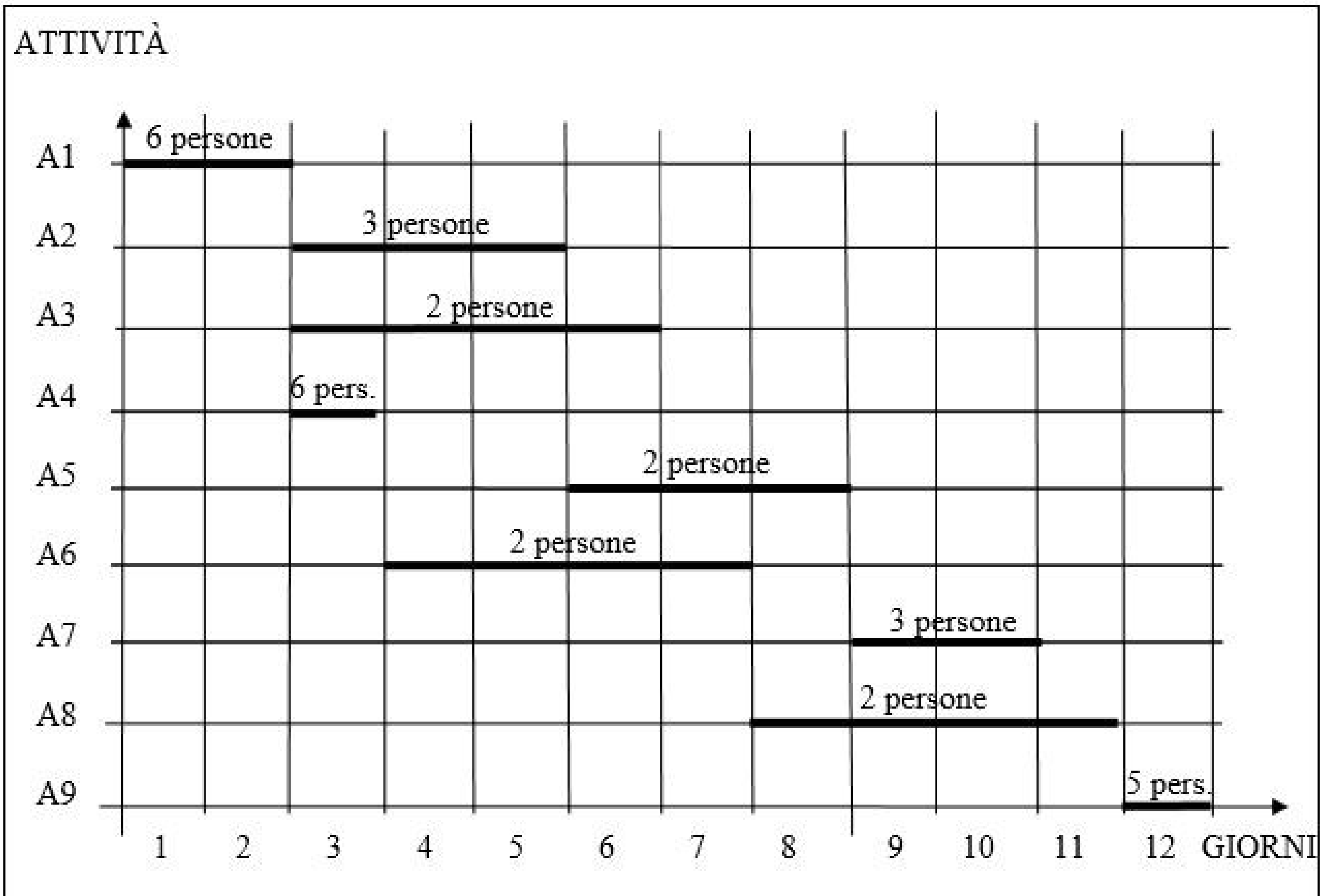
COMMENTI ALLA SOLUZIONE

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.



Olimpiadi di Problem Solving
Informatica e pensiero computazionale nella scuola dell'obbligo
Edizione 2016-2017

PIANIFICAZIONE/6



Olimpiadi di Problem Solving
Informatica e pensiero computazionale nella scuola dell'obbligo
Edizione 2016-2017

CRITTOGRAFIA/1

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 «tqnrunfin 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]

Olimpiadi di Problem Solving
Informatica e pensiero computazionale nella scuola dell'obbligo
Edizione 2016-2017

CRITTOGRAFIA/2

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

CRITTOGRAFIA/3

Olimpiadi di Problem Solving
Informatica e pensiero computazionale nella scuola dell'obbligo
Edizione 2016-2017

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»

Olimpiadi di Problem Solving
Informatica e pensiero computazionale nella scuola dell'obbligo
Edizione 2016-2017

CRITTOGRAFIA/4

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	l	e	m	s	o	l	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.

Olimpiadi di Problem Solving
Informatica e pensiero computazionale nella scuola dell'obbligo
Edizione 2016-2017

CRITTOGRAFIA/5

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

OLIMPIADI di PROBLEM SOLVING

Informatica e pensiero computazionale nella scuola dell'obbligo

Edizione 2016-2017

MOVIMENTI ROBOT/1

PREMESSA – PRIMA PARTE

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,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 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]).

MOVIMENTI ROBOT/2

Olimpiadi di Problem Solving
Informatica e pensiero computazionale nella scuola dell'obbligo
Edizione 2016-2017

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]

MOVIMENTI ROBOT/3

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.

OLIMPIADI di PROBLEM SOLVING
Informatica e pensiero computazionale nella scuola dell'obbligo
Edizione 2016-2017

MOVIMENTI ROBOT/4

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]

MOVIMENTI ROBOT/5

PROBLEMA 2

In un campo di gara il robot (come descritto nella prima parte della premessa) è nella casella [9,9] con orientamento verso sinistra: trovare la lista L dei comandi da assegnare al robot per fargli compiere il percorso descritto dalla seguente lista di caselle [[9,9],[8,9],[7,9],[6,9],[6,8],[6,7],[6,6],[6,5]]

L	

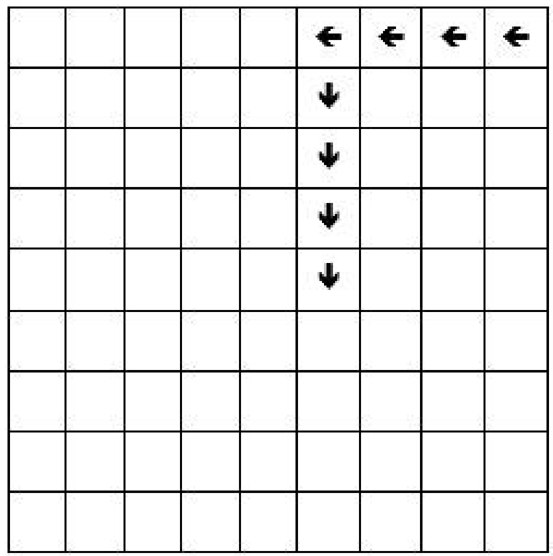
MOVIMENTI ROBOT O PEZZO SCACCHI/6

SOLUZIONE

L	[f,f,f,a,f,f,f,f]

COMMENTI ALLA SOLUZIONE

Per risolvere il problema è conveniente visualizzare il percorso, come nella figura che segue.



Dalla figura è immediato che la sequenza di comandi relativa al percorso è la seguente:

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

Si noti che il quarto comando fa voltare il robot verso il basso, per dargli l'orientamento opportuno per proseguire il percorso, ma non gli fa cambiare posizione.

SOTTOSEQUENZE/1

k) 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) *decrescente*, o quella (strettamente) *crescente*, oppure quella i cui elementi godono di certe proprietà.
N.B. "strettamente" indica che non ci sono elementi ripetuti.

OLIMPIADI di PROBLEM SOLVING

Informatica e pensiero computazionale nella scuola dell'obbligo

Edizione 2016-2017

SOTTOSEQUENZE/2

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. Aiutatelo a scegliere il più grande insieme di programmi che può guardare senza infrangere la regola imposta, se la sequenza delle durate dei programmi (espresse in minuti) è la seguente:

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

Scrivere la risposta nella casella sottostante.

SOTTOSEQUENZE/3

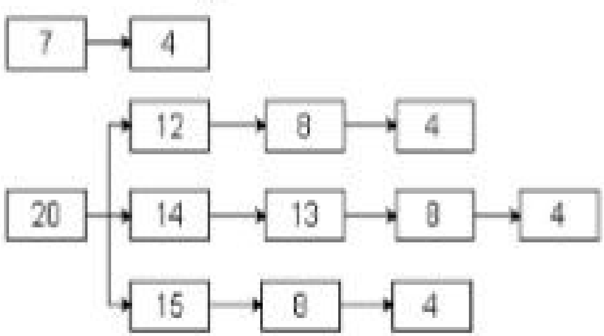
SOLUZIONE

[20,14,13,8,4]

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 costruisce, 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.

SOTTOSEQUENZE/4

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.

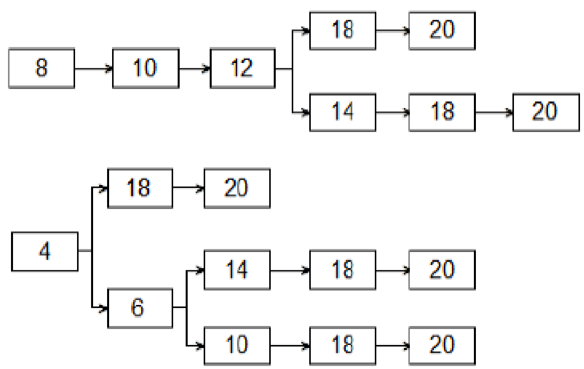
Scrivere la risposta nella casella sottostante.

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.

PSEUDOLINGUAGGIO/1

1. INTRODUZIONE: LA METAFORA

In una cassettiera ci sono dei cassetti 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 cassetti 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 cassetti 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*".

PSEUDOLINGUAGGIO/2

2. L'ASPETTO FORMALE

Invece di parlare di cassetti 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 cassetti) 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;
input A, B, C;
D ← A+B;
E ← C+B-A;
F ← A+B-C;
output D, E, F;
endprocedure;
```

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

Con la scrittura "input" si assegnano dei valori a certe variabili (si scrivono dei numeri sui foglietti contenuti in certi cassetti).

Con la scrittura "output" si fa vedere il valore di certe variabili (si leggono i numeri sui foglietti contenuti in certi cassetti).

Se si *esegue* la procedura PROVA1 e in input alle variabili A, B, C vengono assegnati rispettivamente i valori 5, 8 e 3, in output, 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*).

PSEUDOLINGUAGGIO/3

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*) "input",

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
input 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)
output A, C1, D;	si rendono disponibile all'esterno i valori delle variabili A, C1, D
endprocedure;	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.

PSEUDOLINGUAGGIO/4

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 in *output* di M è 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;
 ...
```

PSEUDOLINGUAGGIO/5

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 ESEMPIO;
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;
output A, B;
```

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

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

PSEUDOLINGUAGGIO/6

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;
output 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.

PSEUDOLINGUAGGIO/7

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
procedure ESEMPIO3;	
variables A, B, C integer;	dichiarazione di variabili intere
variables TF, SF, R float;	dichiarazione di variabili razionali
B ← 2;	assegnazione (←) di costante intera
A ← 6;	assegnazione (←) di costante intera
TF ← 2.0;	assegnazione (←) di costante razionale
SF ← 5.0;	assegnazione (←) di costante razionale
C ← A/B;	calcolo e assegnazione (←) con valori interi
R ← SF/TF;	calcolo e assegnazione (←) con valori razionali
output C, R;	per C l'output è 3; per R l'output è 2.5
endprocedure;	

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.

PSEUDOLINGUAGGIO/8

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, la estrazione e la determinazione della lunghezza.
I seguenti sono esempi di procedure che usano variabili string e intere.

PSEUDOLINGUAGGIO/9

PROBLEMA1
Si consideri la seguente procedura PROVA1.

```
procedure PROVA1;
variables A, B, C, D integer;
input A, B;
C ← A + B;
D ← A × B;
A ← C+B;
```



```
B ← (A+B) ×(A- B);
output C, D, A, B;
endprocedure;
```

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

Olimpiadi di Problem Solving
Informatica e pensiero computazionale nella scuola dell'obbligo
Edizione 2016-2017

PSEUDOLINGUAGGIO/10

PROBLEMA2

Si consideri la seguente procedura PROVA2.

```
procedure PROVA2;
variables A, B, M, N, K integer;
input A;
M ← 0;
N ← 0;
for K from 1 to 10 step 1 do;
        input B;
        if A > B     then M ← M + A;   endif;
        if A < B     then N ← N + A;   endif;
endfor;
output M, N;
endprocedure;
```

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