OPS – CALABRIA

CATANZARO 11-12 febbraio 2016

Antonio Teolis
antonio.teolis@unibo.it

OPS-CALABRIA perché OPS 1

http://studiumanistici.unipv.it/?pagina=corsi&id=18397

UNIVERSITÀ DI PAVIA

Insegnamento: STORIA GRECA
Corsi di laurea: Lettere
Anno accademico: 2014/2015
Obiettivi: Il corso, a carattere …
Metodi didattici: Lezioni frontali
Discussioni
Problem solving
Analisi e traduzioni di testi in greco
Uso di mappe interattive

OPS-CALABRIA perché OPS 2

https://esse3web.unisa.it/unisa/Guide/PaginaCorso.do?corso_id=500210&ANNO_ACCADEMICO=2015

UNIVERSITÀ DI SALERNO

Corso di laurea magistrale: LINGUE E LETTERATURE MODERNE

….

Capacità di applicare conoscenza e comprensione

….

L'estensione e l'approfondimento delle aree linguistiche e culturali di competenza forniscono agli studenti molteplici strumenti di riflessione, di ***problem-solving*** e di applicazione interdisciplinare anche in campi nuovi o non familiari.

OPS-CALABRIA perché OPS 3

http://studiumanistici.unipv.it/?pagina=corsi&id=18407

UNIVERSITÀ DI PAVIA

Insegnamento: LINGUA INGLESE (CORSO AVANZATO)

Corsi di laurea: Storia d'Europa

Anno accademico: 2014/2015

Obiettivi: Il corso si propone di…

Programma

MODULE 1: WRITING

1) Introduction to writing …

….

8) Classroom practice: **Problem solving** in writing.

OPS-CALABRIA presentazione

PRIMA PARTE

- Storia, Struttura ed Organizzazione delle OPS
- Obiettivi didattici, *problem solving* e pensiero algoritmico (anche al I ciclo)
- Formalizzare e risolvere problemi: esempi

SECONDA PARTE

- Lavori di gruppo
- Riflessioni e commenti

OPS 1 storia

- MIUR/Università di Bologna;
- obbiettivo: complementare le OII, allargando la platea dei selezionabili;
- caratteristiche: a squadre (poi anche individuali), allenamenti/gare, estensione alle elementari.

OPS 2 struttura 1

- 6/7 prove: 4/5 allenamenti mensili (da novembre a marzo)
  selezione regionale (marzo o aprile)
  finale (aprile o maggio)
- squadre di 4 partecipanti;
- tre livelli: elementari, a squadre;
  medie, a squadre e individuali
  biennio delle superiori, a squadre e individuali
- prove a squadre: 10-12 problemi, da risolvere in 90 minuti
- prove individuali: 6-8 problemi, da risolvere in 90 minuti

OPS 3 struttura 2

- tre tipi di problemi: - ricorrenti (fissati anno per anno),
  - lettura e comprensione di programmi scritti in uno “pseudolinguaggio” di programmazione,
  - “nuovi”, in inglese;
- sito con materiali e strumenti di gestione (in rifacimento);
- “semplici” convenzioni formali (nella scrittura dei problemi e in quella delle risposte).

OPS 4 struttura 3

I testi dei problemi:

- vengono prodotti da appositi gruppi di lavoro (attualmente a Bologna e Biella);
- successivamente vengono sottoposti a controllo di qualità da parte di un ulteriore gruppo di lavoro (attualmente a Biella);
- da ultimo sono passati al *web master* che cura la pubblicazione sul sito, la gestione delle gare, la raccolta e valutazione dei risultati.

Si cerca di distribuire la gare (dei tre livelli, singole ed individuali) in giorni della settimana di volta in volta differenti

OPS 5 struttura 4

- Uno dei punti di forza delle OPS è che, dopo le gare, insieme alle risposte attese, viene pubblicato anche un commento talvolta sintetico, talvolta molto dettagliato (a seconda dei casi) sulla tecnica risolutiva dei vari problemi.
- Tale commento può essere un utile spunto di discussione e approfondimento con i ragazzi.
- Inoltre è a disposizione un “servizio” di domande (e risposte) denominato FAQ, sia durante le prove che dopo.

OPS 6 caratteristiche/obbiettivi

- Istruzione/addestramento attraverso l'allenamento/gara;
- rivalutazione della "atmosfera" (blandamente) competitiva;
- coinvolgere (e trascinare) tutti i ragazzi (e farsi trascinare!);
- trasformare la "prova" da evento inusuale in *business as usual* poi in **divertimento (!!!!)**;
- problemi ricorrenti: per insegnare a tutti certi algoritmi di base;
- problemi "nuovi": per stimolare la curiosità e rompere la *routine*; (ieri è stato suggerito di aumentarne il numero)

OPS 7 auspici

INVITO

- controllare (non fatto centralmente per vari motivi), da parte dei singoli insegnanti, (evitando l'effetto selezione) la correlazione dei risultati OPS con:
    - risultati scolastici,
    - risultati delle prove Invalsi,
    - risultati extra scolastici;
- partecipazione più attiva degli insegnanti con (per esempio):
    - giudizi sulla difficoltà *percepita* dei vari problemi,
    - proposte di innovazioni o tipi di problemi da inserire nelle prove,
    - suggerimenti per migliorare ed espandere l'iniziativa.

OPS 8 convenzioni

I testi delle gare sono scritti in linguaggio naturale (italiano o inglese), con le seguenti eccezioni:

- vengono proposte semplici procedure scritte in uno pseudolinguaggio di programmazione procedurale *Algol-like*: devono essere “capite” ed “eseguite”;
- nel testo dei problemi e nelle risposte vengono usate “particolari” scritture: *termini* e *liste* (come nel linguaggio Prolog);
- le risposte sono trattate come stringhe di caratteri e come tali sono confrontate con quella attesa: quindi devono essere uniche

Elementi sintattici 1

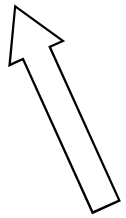
ELEMENTI FORMALI NEI TESTI DEI PROBLEMI

- termine: <nome del termine>(<argomento>)
  <nome del termine>(<argomento>,<argomento>)
  <nome del termine>(<argomento>,<argomento>,<argomento>)
  ….
- lista: []
  [<elemento>]
  [<elemento>,<elemento>]
  [<elemento>,<elemento>,<elemento >]
  ….

N.B. **NON ci sono spazi nella scrittura di termini e liste.**

Elementi sintattici 2 esempio 1

definizione: minerale(<nome>,<peso in chili>,<valore in euro>)

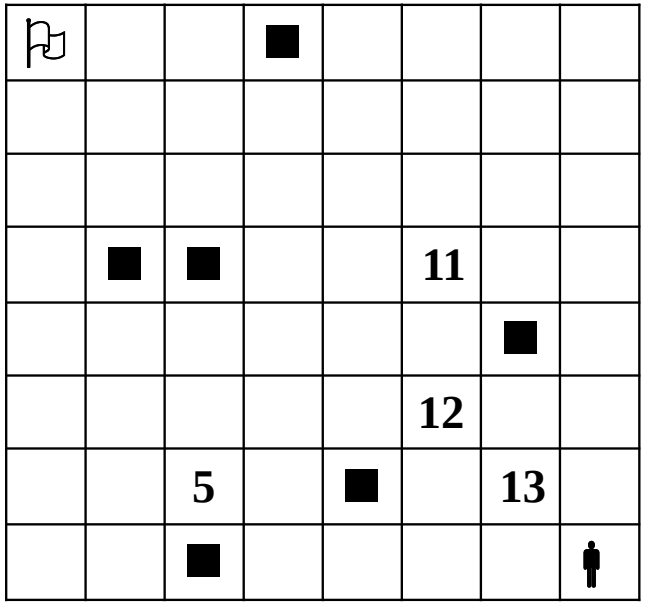


sintassi e semantica

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

minerale		
nome	peso in chili	valore in euro
m1	30	213
m2	35	200
m3	38	230

Elementi sintattici 3 esempio 2



le caselle inibite sono quelle le cui coordinate sono nella lista:
[[3,1],[5,2],[2,5],[3,5],[4,8],[7,4]]
i premi sono descritti dalla lista: [[3,2,5],[7,2,13],[6,3,12],[6,5,11]]

Problem solving 1

problem solving: significati più disparati (e non utili).

- articoli su Wikipedia in italiano e in inglese;
- "il *problem solving* crea stress, meglio pensare per obiettivi";
- le dita del marziano.

Poche eccezioni; esempio:

"La scrittura: un caso di *problem solving*" di Maria Emanuela Piemontese in "Laboratorio di scrittura. Non solo temi all'esame di Stato. Idee per un curricolo", Quaderni del Giscel, La Nuova Italia, Firenze, 2002;

Problem solving 2

Breve storia.

- George Polya, “How to solve it”, 1945: perché solo allora?
- l’informatica (EDVAC, 1951) cambia tutto:
  - entro un lustro compare un nuovo mestiere, il programmatore;
  - appaiono (numerosi) linguaggi (artificiali) di programmazione;
  - appaiono i primi corsi di informatica (*computer science*);
  - c’è bisogno di *molti* programmatori;
- 1968: *big bang*; si acquista una “nuova consapevolezza” (dettagli dopo): i programmi sono “pieni” di errori;
- per evitare gli errori occorre usare “correttamente” i linguaggi (artificiali) di programmazione $\leftrightarrow$ per evitare gli errori occorre “pensare” correttamente;

Problem solving 3 digressione

- Seconda guerra mondiale: il calcolo di una traiettoria di un pezzo di artiglieria “medio” richiedeva circa 500-1000 operazioni; una ***tavola*** di tiro conteneva 2000-4000 traiettorie;
- Approvvigionamento “di mercato” per USA Army;
- 1943: il BRL (Balistic Research Laboratory) incarica la Moore School (Pennsylvania University, Philadelphia) di costruire una macchina per automatizzare la produzione di tavole di tiro;
- Viene incaricato il sottotenente Herman Goldstine (un matematico) di seguire il progetto;
- Agosto 1944: Goldstine, alla stazione di Aberdeen (Maryland), incontra John von Neumann.

Problem solving 4

- per risolvere *correttamente* i problemi occorre “pensare” *correttamente*;
- “pensare” correttamente riguardo a un problema $\leftrightarrow$ esprimersi correttamente su quel problema (con un opportuno linguaggio);
- linguaggio (artificiale) “corretto” $\leftrightarrow$ pensiero (razionale conscio) algoritmico (?);
- (linguaggio naturale libero $\leftrightarrow$ pensiero poetico).

il pensiero algoritmico è un fatto linguistico

Problem solving 5

Il *problem solving* compare come disciplina autonoma perché ci sono *tanti* programmatori (cioè solutori di problemi) perché ci sono *tanti* problemi (resi possibili dall'informatica o creati (!!) dall'informatica: F35).

Successivamente la dizione viene cooptata da altre discipline per descrivere processi al loro interno.

Negli anni '80 del secolo scorso (il 1900!) si diffonde la dizione "pensiero algoritmico" per indicare

all'inizio

il processo mentale che porta alla formulazione di un problema in modo che ammetta una soluzione "algoritmica" o "computazionale" (cioè ottenibile scrivendo un "programma", che è un testo in un linguaggio artificiale).

Problem solving 6

Estremizzando si può dire che il *problem solving* (e il pensiero algoritmico)

è

la formulazione del problema in maniera algoritmica, cioè la formulazione del problema in un linguaggio artificiale "opportuno" (per esempio di programmazione).

Digressione 1

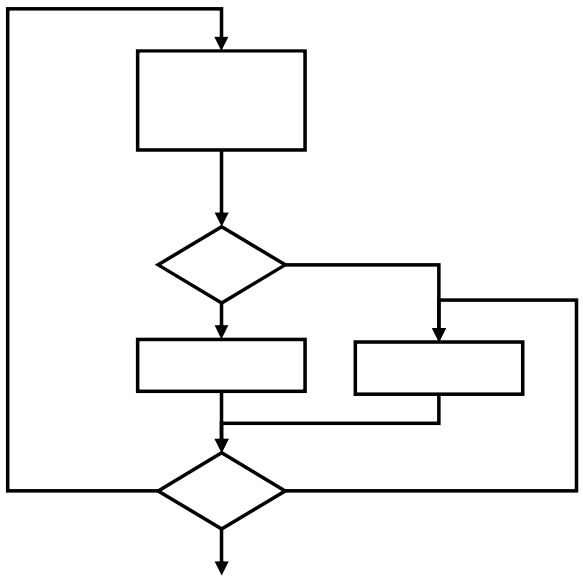
- *Nato conference "Software Engineering", Garmisch, October 7-11 1968*. Report del gennaio 1969.
- "*A discipline of programming*" di E. Dijkstra, 1976.

(disciplina ha il doppio significato di "corpo di conoscenze" e "regola di comportamento (severa!)", nel titolo è usata nella seconda accezione.)

- *no flow chart* (rappresentazione grafica bidimensionale);
- (il linguaggio $\leftrightarrow$ pensiero razionale conscio) è monodimensionale;
- programmazione strutturata, *goto-less programming*, …

Digressione 2

flow chart

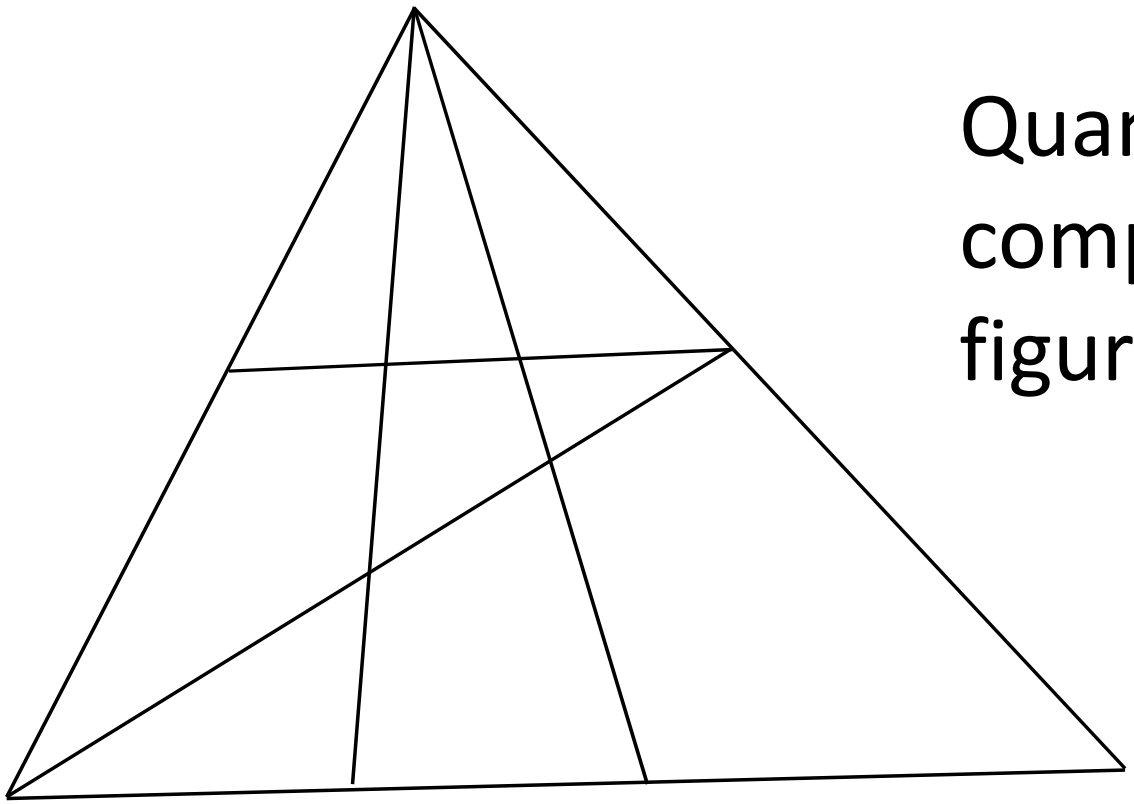


programma

```
            ……………..
            …… go to label2.
            ……………
label1…………….
            ……………
            …… go to label1
            ……………
label2……………..
            ……………
```

NO! occorre *pensare* direttamente nel linguaggio (artificiale) di programmazione (opportunamente *disciplinato*).
Teorema di Böhm-Jacopini.

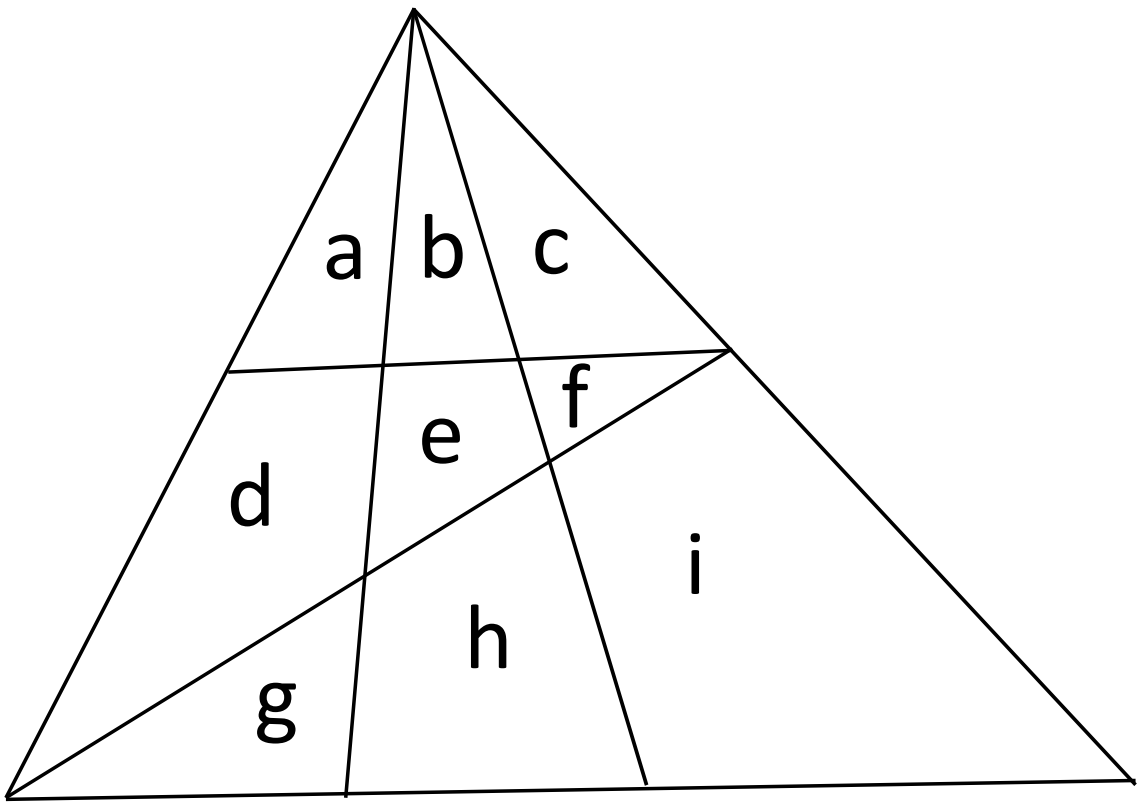
Triangoli 1 il problema



Quanti triangoli compaiono nella figura?

(squadra delle elementari al lavoro)

Triangoli 3 poter parlare/ragionare



DARE I NOMI

poligoni elementari: a, b, c, d, e, f, … (lettere)

triangoli: [a], [b], …, [a,b], …, [d,e,f], … (liste ordinate!)

Triangoli 4 costruire

	a	b	c	d	e	f	g	h	i
1									
2									
3									
4		classificare							
5									
6									
7									
8									
9									

Triangoli 5 costruire

	a	b	c	d	e	f	g	h	i
1	[a]	[b]	[c]			[f]	[g]		
2	[a,b] [a,d]	[b,c] [b,e]	[c,f]		[e,f]		[g,h]		
3	[a,b,c] [a,d,g]	[b,e,h]	[c,f,i]	[d,e,f]			[g,h,i]		
4	[a,b,d,e]	[b,c,e,f]							
5									
6	[a,b,c,d,e,f] [a,b,d,e,g,h]	[b,c,e,f,h,i]							
7									
8									
9	[a,b,c,d,e,f,g,h,i]								

Knapsack 1 il 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 del minerale>, <valore in euro>, <peso in Kg>)

Nel deposito sono presenti i seguenti minerali:

minerale(m1,40,53) **minerale(m2,42,55)** **minerale(m3,51,51)**

minerale(m4,50,54) **minerale(m5,55,56)** **minerale(m6,53,51)**

Disponendo di un motocarro con portata massima di 104 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.

N.B. la lista deve essere ordinata: m1 < m2 < m3 < …

Knapsack 2 oggetti e strutture

- oggetti: m1, m2, m3, m4, m5, m6
- strutture: *insiemi* di due elementi
- rappresentazione delle strutture: liste *ordinate*!

  (l'insieme associato a [m1,m2] è lo stesso di quello associato a [m2,m1])
- costruzione dello "spazio" delle possibili soluzioni

  (forza bruta = metodo e sistema; talvolta: euristica)

Knapsack 3 costruire con metodo

	VALORE	PESO	TRASPORTABILE	
[m1,m2]	82	108	no	
[m1,m3]	91	104	si	
[m1,m4]	90	107	no	
[m1,m5]	95	109	no	
[m1,m6]	93	104	si	
[m2,m3]	93	106	no	
[m2,m4]	92	109	no	
[m2,m5]	97	111	no	
[m2,m6]	95	106	no	
[m3,m4]	101	105	no	
[m3,m5]	106	107	no	
[m3,m6]	104	102	si	massimo valore
[m4,m5]	105	110	no	
[m4,m6]	103	105	no	
[m5,m6]	108	107	no	

Knapsack e triangoli

Osservazioni importanti sui due problemi appena visti:

- il secondo problema è (stato) completamente formalizzato: si può scrivere un programma per risolverlo, con le informazioni formali (linguistiche) contenute nel testo del problema;
- il primo problema NON è stato completamente formalizzato:
    - chi "sono" a, b, …?
    - cosa vuol dire "essere un triangolo"?
- come si può immaginare di formalizzare il problema (e quali sono le "primitive" da usare)?

- Domanda retorica: non si attende una risposta, ma rivolta ai ragazzi …

Knapsack 4 altro problema

VARIAZIONE

Disponendo di un motocarro con portata massima di 102 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.

saper usare la forza bruta (è fondamentale!!!); prima, però, sempre una occhiata intelligente

Grafi 1 il problema

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

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

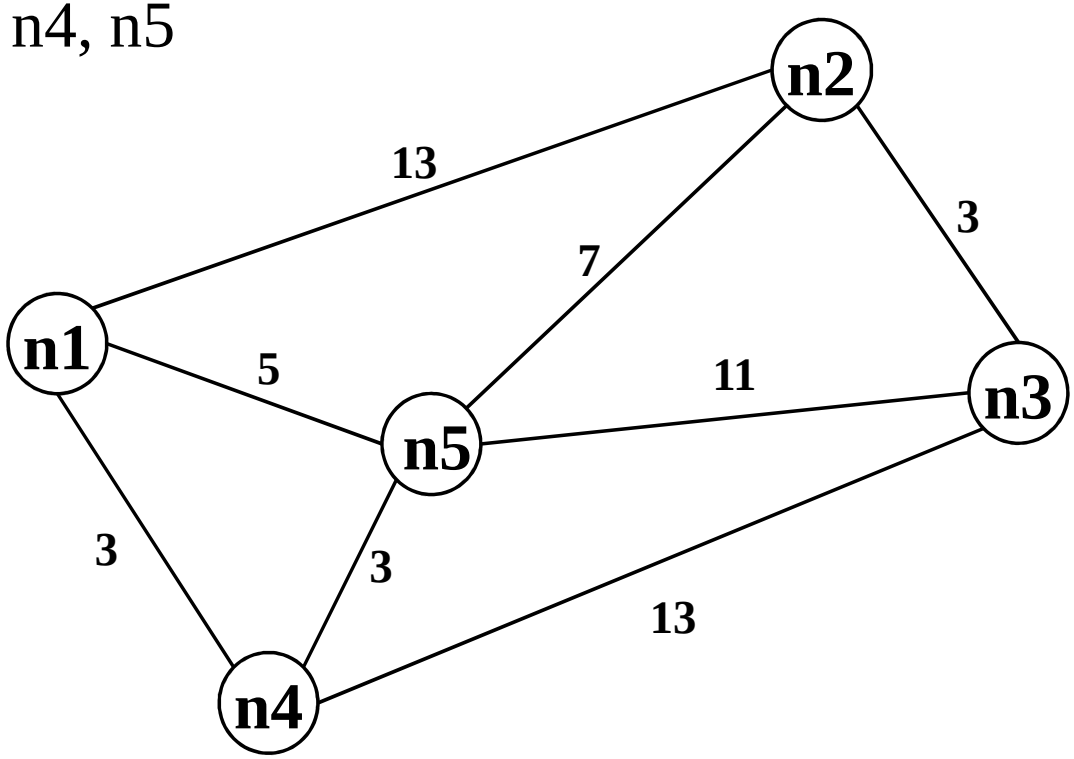
N.B. definizione: arco(<nodo1>,<nodo2>,<distanza>)

Disegnare il grafo e trovare:

- la lista L1 del percorso *semplice* più breve tra n1 e n3;
- la lista L2 del percorso *semplice* più lungo tra n1 e n3.

Grafi 2 oggetti

oggetti: n1, n2, n3, n4, n5
archi
lunghezze

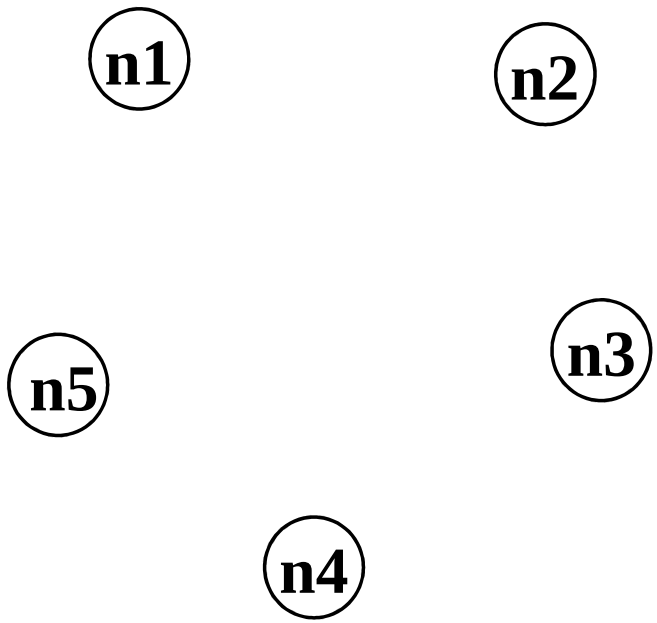


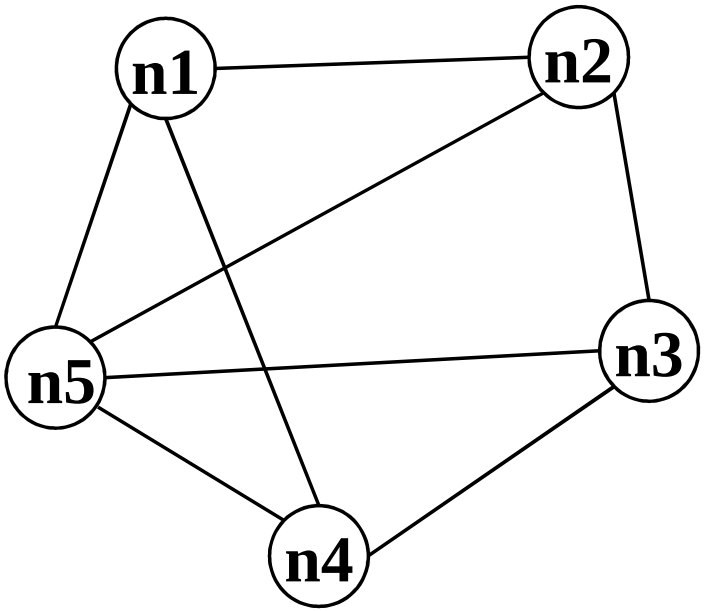
Si noti che le lunghezze degli archi che compaiono nei termini (che rappresentano delle strade) *non* sono necessariamente proporzionali a quelle degli archi

Grafi 3 per tentativi

oggetti: n1, n2, n3, n4, n5

archi: arco(n1,n2,13) arco(n2,n3,3) arco(n3,n4,13) arco(n1,n4,3)
arco(n4,n5,3) arco(n5,n1,5) arco(n2,n5,7) arco(n3,n5,11)



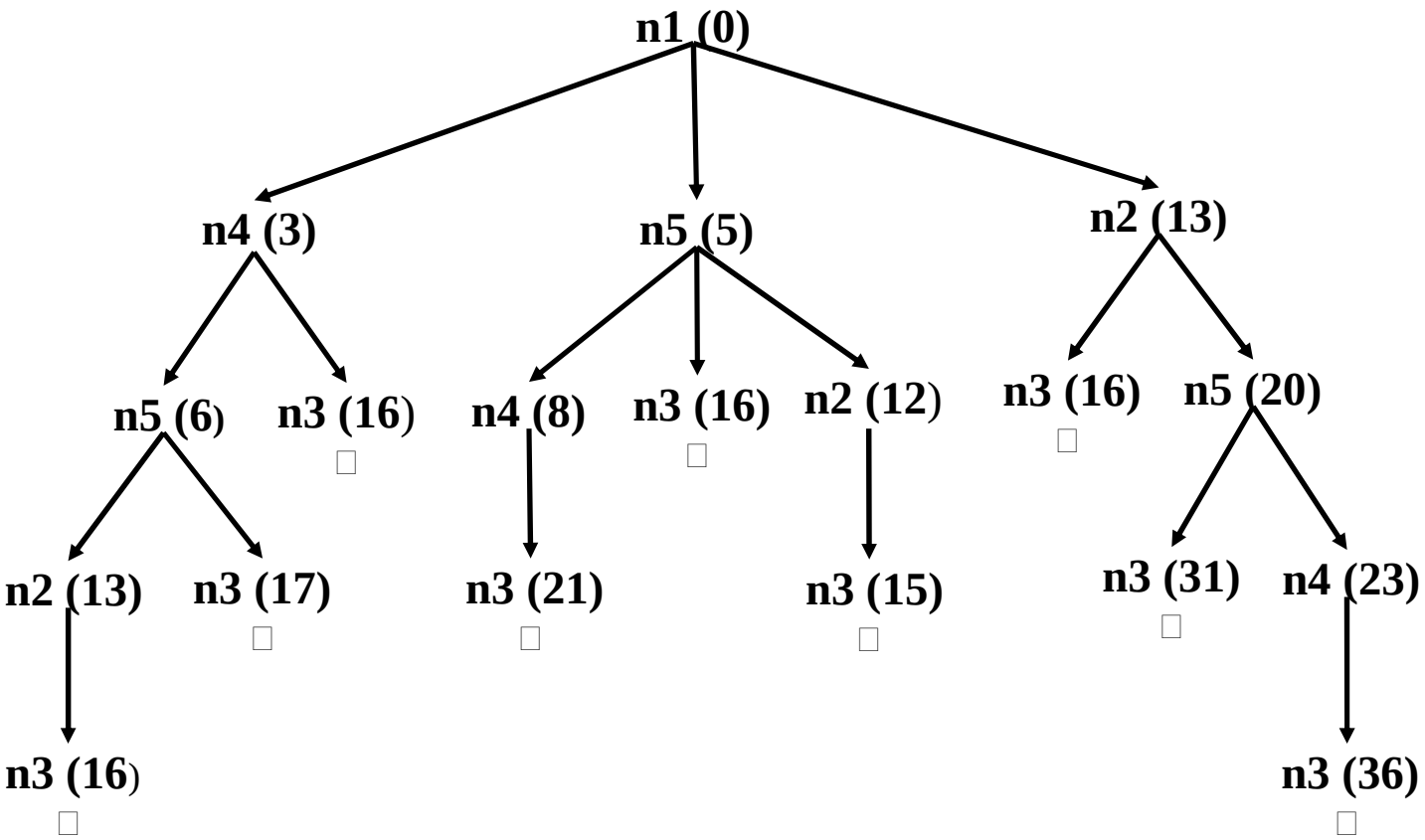


Grafi 4 costruire

Strutture: cammini semplici
costruzione dello spazio delle possibili soluzioni

CAMMINI	LUNGHEZZE	
[n1, n2, n3]	16	
[n1, n2, n5, n3]	31	
[n1, n2, n5, n4, n3]	36	L2
[n1, n5, n3]	16	
[n1, n5, n2, n3]	15	L1
[n1, n5, n4, n3]	21	
[n1, n4, n3]	16	
[n1, n4, n5, n3]	17	
[n1, n4, n5, n2, n3]	16	

Grafi 5 un metodo



Regole 1 il problema

Siano date le seguenti regole:

regola(1,[j,k],h)	regola(2,[f,t],u)	regola(3,[u,p],w)
regola(4,[x],j)	regola(5,[f,t],p)	regola(6,[x,j],k)
regola(7,[j,h,k],w)	regola(8,[u,k],w)	regola(9,[u],h)

definizione: regola(<nome>,<antecedenti>,<conseguente>)

N.B. con la regola <nome> dagli <antecedenti> si deduce (o calcola) il <conseguente>.

Trovare le liste:

- L1 che descrive il procedimento per dedurre **k** a partire da **x**;
- L2 che descrive il procedimento per dedurre **w** a partire da **f** e **t**.

N.B. le liste sono liste di nomi di regole (devono essere “opportune”)

Regole 2 esempi

ESEMPI

Dalla formule:

$$\text{ipotenusa}^2 = \text{cateto1}^2 + \text{cateto2}^2$$

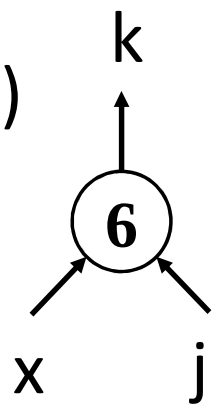
$$\text{circonferenza} = \text{raggio} \times 3{,}14$$

si ricavano le regole

- regola(pitagora1,[cateto1,cateto2],ipotenusa)
- regola(pitagora2,[ipotenusa,cateto1],cateto2)
- regola(cerchio1,[raggio], circonferenza)
- regola(cerchio1,[circonferenza], raggio)

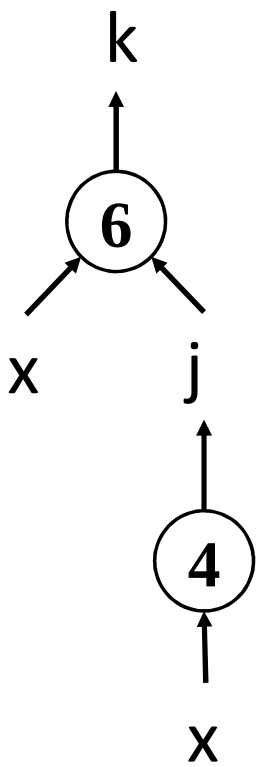
Regole 4 uso

regola(6,[x,j],k)



regola(4,[x],j)

procedimento
[4,6]



Regole 5 metodi di deduzione/calcolo

1. metodo *backward*: partire dall'obbiettivo/risultato e procedere costruendo l'albero di deduzione dall'alto, arrestandosi quando tutte le foglie sono dei dati;
2. metodo *forward*: partire dai dati e costruire l'albero di deduzione dal basso, arrestandosi quando la radice è l'obbiettivo/risultato.
3. il primo è usato (di solito) nei sistemi automatici di deduzione.
4. nelle deduzioni ``manuali˝ si usano entrambi: **è importante saper sospendere un modo di ragionare e iniziarne un altro.**

Regole 6 esempio di deduzione

procedimento per dedurre **w** a partire da **f** e **t**

metodo backward:

regola(3,[u,p],w) regola(7,[j,h,k],w) regola(8,[u,k],w)

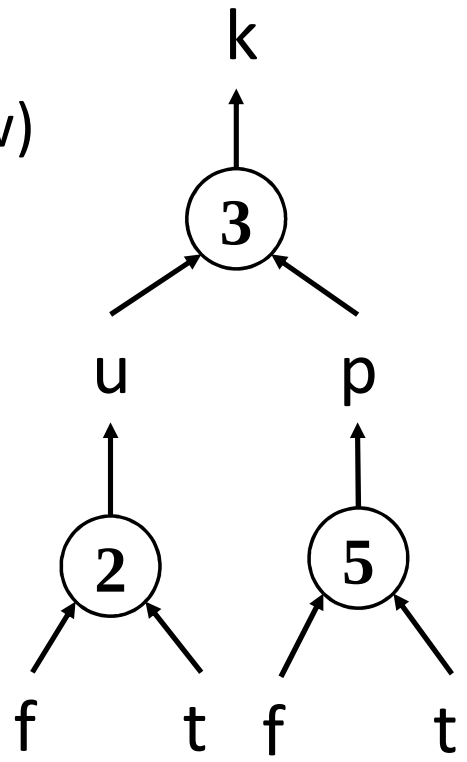
quale scegliere?

metodo forward:

regola(2,[f,t],u) regola(5,[f,t],p)

è chiaro che la soluzione è:

[2,5,3] lista "opportuna" (non lo è [5,2,3])



Inglese/tre composto problema

A 18000-gallon pool can be filled by two inlet pipes. The first pipe can fill the pool in 36 hours; the second pipe could fill the same pool in 45 hours. How long does it take to fill this pool if both pipes are used?

Versione per le elementari: la versione per le medie non ha la capacità della vasca.

Metodo anglosassone: **think 1** (riduzione all'unità)

Inglese/tre composto soluzione

- quello che succede in **“1”»** (di solito) può essere sommato;
- in **una** ora: primo condotto → 500 galloni (18000/36);
  secondo condotto → 400 galloni (18000/45);
  entrambi i condotti → 900 galloni (somma);
- l’intera vasca, con i due condotti → 20 ore (18000/900);

- nella versione per le medie si sommano le frazioni “egiziane”;

N.B. i problemi del “tre composto” sono abbastanza frequenti; comunque, in generale, quelli in inglese non sono *ricorrenti*.

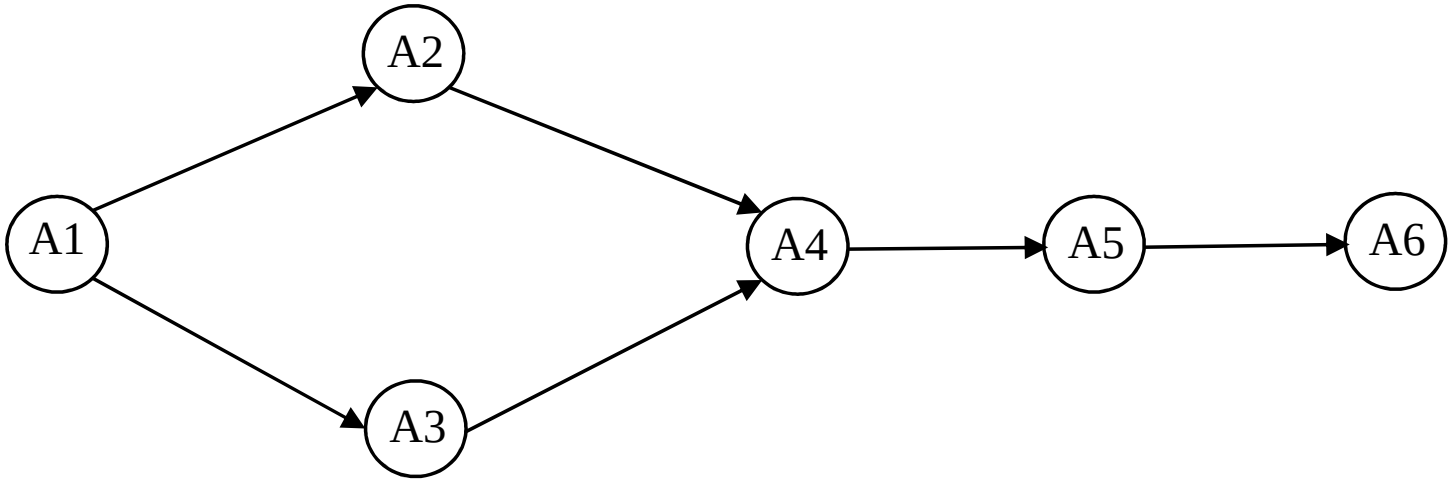
Project management 1 problema

La tabella che segue descrive le attività (indicate rispettivamente con le sigle A1, A2, A3, …) necessarie per un progetto.

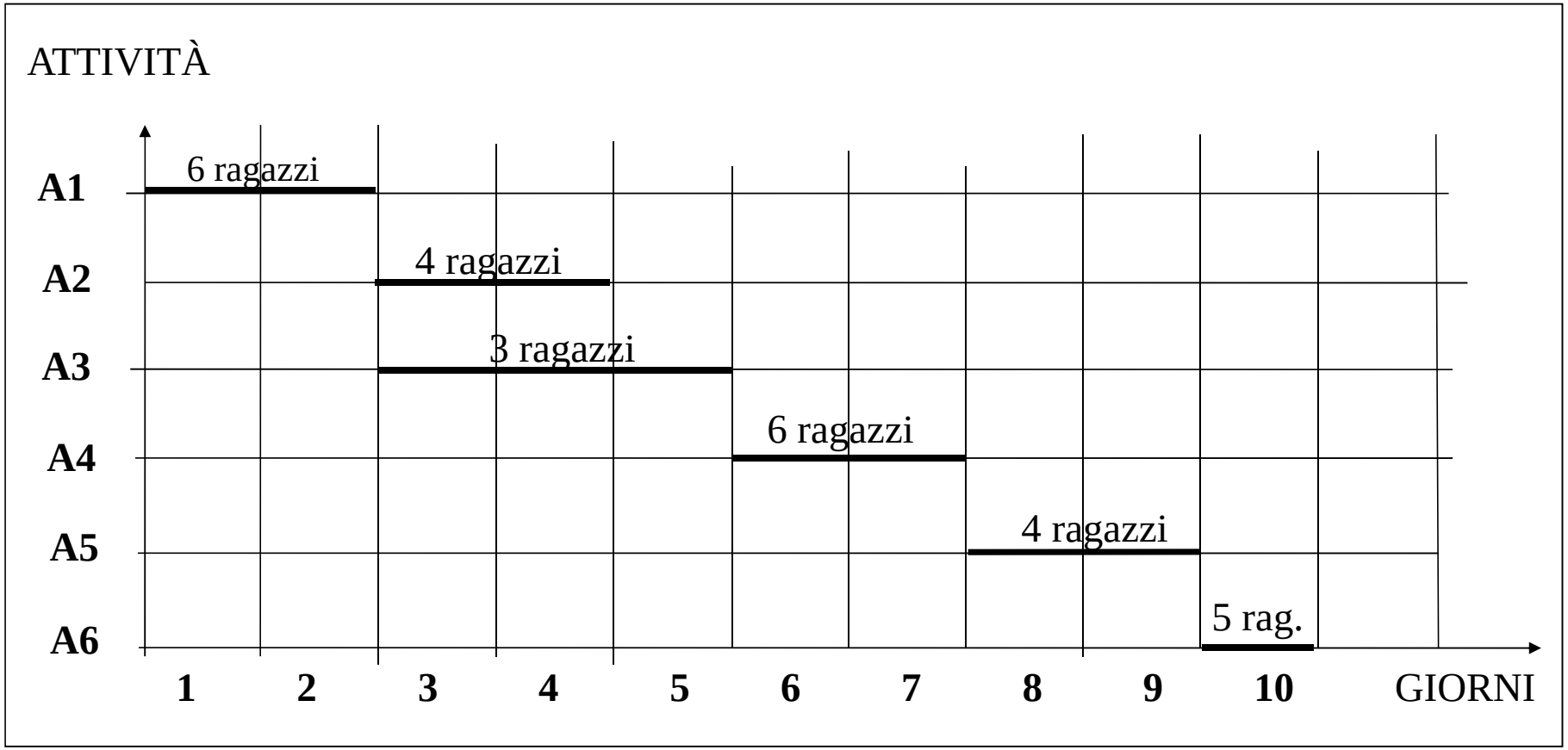
ATTIVITÀ	RAGAZZI	GIORNI
A1	6	2
A2	4	2
A3	3	3
A4	6	2
A5	4	2
A6	5	1

Le priorità tra le attività sono: [A1,A2], [A1,A3], [A2,A4], [A4,A5], [A3,A4], [A5,A6].

Project management 2 diagramma delle precedenze



Project management 3 diagramma di Gantt



Project management 3 problema 2

Il testo del problema, per ragioni "storiche" è scritto in maniera anomala: ha una tabella e un grafo è dato con liste.

Definizioni:

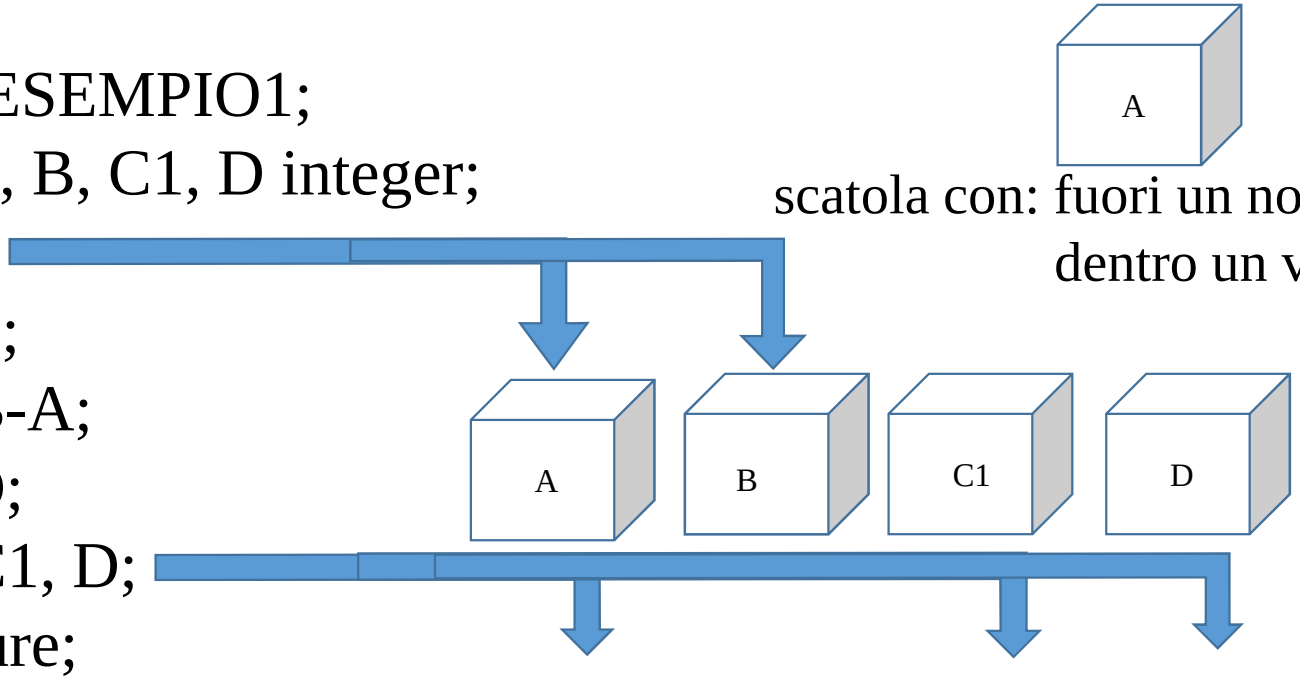
attività(<nome di attività>, <numero di persone>, <durata in giorni>)

precedenza(<nome di attività precedente>,<nome di attività>)

attività(A1,6,2)	precedenza(A1,A2)
attività(A2,4,2)	precedenza(A1,A3)
attività(A3,3,3)	precedenza(A2,A4)
attività(A4,6,2)	precedenza(A4,A5)
attività(A5,4,2)	precedenza(A3,A4)
attività(A6,5,1)	precedenza(A5,A6)

Modello a scatola 1 primo esempio

procedure ESEMPIO1;
variables A, B, C1, D integer;
input A, B;
C1 ← A+B;
D ← C1+B-A;
A ← C1+D;
output A, C1, D;
endprocedure;



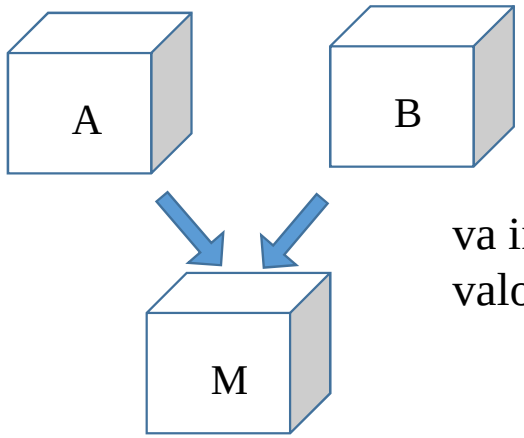
scatola con: fuori un nome e dentro un valore

valori di input per A e B**:** 5, 9
valori di output per A, C1, D**:** 32, 14 e 18

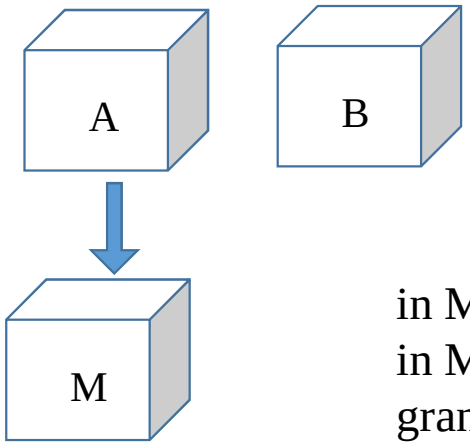
Modello a scatola 2 secondo esempio

variables A, B, M integer;
…
if A > B
then M ← A;
else M ← B;
endif;
…

M ← B;
if A > B then M ← A;
endif;



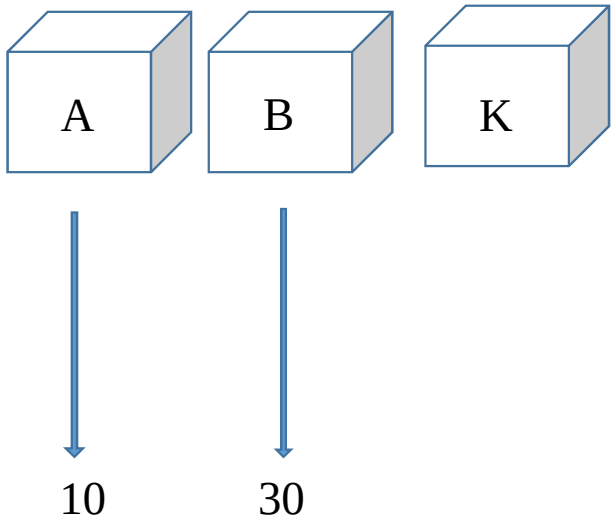
va in M il più grande tra il valore di A e quello di B



in M va il valore di B;
in M va il valore di A se è più grande di quello di B

Modello a scatola 3 terzo esempio

```
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;
output A, B;
endprocedure;
```

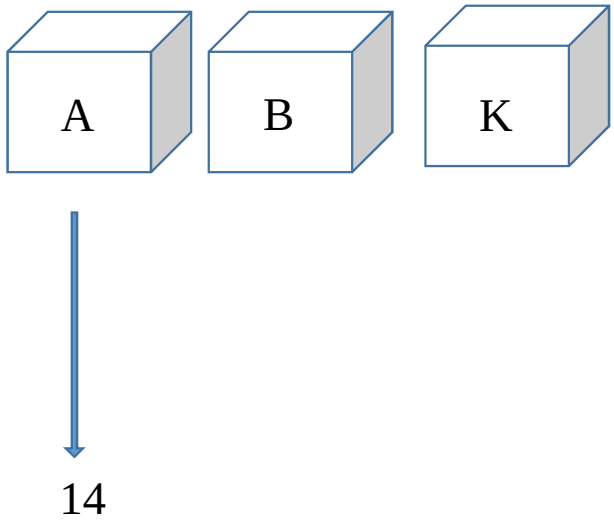


valori successivi di K: 1, 2, 3, 4

N.B. differenza tra variabile e costante

Modello a scatola 4 quarto esempio

```
procedure ESEMPIO3;
variables A, B, K integer;
B ← 10;
A ←  0;
K  ← 0;
while A < B do;
    K ← K + 1;
    A ← K × K + A;
endwhile;
output A;
endprocedure;
```



valori successivi di K: 0, 1, 2, 3

Workshop 1

Esercizi proposti

1. regole
2. tre composto (inglese)
3. *knapsack*
4. percorsi in un grafo
5. *project management*
6. pseudolinguaggio
7. inglese

Workshop 2

metodo: collaborazione di ogni partecipante del gruppo;

obbiettivo: individuare gli spunti didattici da trarre da ciascun (tipo di) esercizio;

problema: gli esercizi sono di difficoltà medio-alta: occorre immaginarli scalati (in alto o in basso) al livello scolastico a cui opera ogni partecipante.