OPS – SEMINARIO

2016/2017

Antonio Teolis

antonio.teolis@unibo.it

OPS 2 presentazione

…

- Storia, Struttura ed Organizzazione delle OPS.
- *Problem solving*, pensiero algoritmico e computazionale: definizione e obiettivi didattici (anche per il I ciclo).
- Formalizzare e risolvere problemi: esempi e discussione.
- …

OPS 3 storia

- MIUR-Università di Bologna;
- obbiettivo: complementare le OII, allargando la platea dei *selezionabili*;
- iniziativa: addestramento mediante allenamenti/gare, a squadre (poi anche individuali), partendo dalle elementari.

OPS 4 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 o individuali;
- tre livelli: elementari, a squadre;
  medie, a squadre e individuali,
  biennio delle superiori, a squadre e individuali;
- prove a squadre: 10-12 problemi, da trattare in ~90 minuti;
- prove individuali: 6-8 problemi, da trattare in ~90 minuti.

OPS 5 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 6 struttura 3

I testi dei problemi:

- vengono prodotti da appositi gruppi di lavoro;
- successivamente vengono sottoposti a controllo di qualità da parte di un ulteriore gruppo di lavoro (diverso dai precedenti); (N.B. purtroppo a volte rimangono delle "imprecisioni");
- 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 7 struttura 4

- Uno dei punti di forza delle OPS: dopo le gare, insieme con le risposte attese, vengono pubblicati dei "commenti", talvolta sintetici, talvolta (molto) dettagliati sulla tecnica risolutiva dei vari problemi.
- Tali commenti possono essere utili spunti di discussione e approfondimento con i ragazzi.
- Inoltre è a disposizione un "servizio" di domande (e risposte) denominato (impropriamente) FAQ, sia durante le prove che dopo.

OPS 8 caratteristiche/obbiettivi

- Istruzione/addestramento attraverso l'allenamento/gara;
- rivalutazione della "atmosfera" (~~moderatamente~~) competitiva;
- coinvolgere (e trascinare) **tutti** i ragazzi (e farsi trascinare!);
- trasformare la "prova", la "gara", da evento inusuale in *business as usual*, poi in divertimento atteso;
- problemi ricorrenti: per essere sicuri che tutti padroneggino certe tecniche di base;
- problemi "nuovi": per stimolare la curiosità e abituare all'imprevisto.

OPS 9 auspici

AUSPICABILE DA PARTE DEI SINGOLI INSEGNANTI:

- controllo (evitando l'*effetto selezione*) della correlazione dei risultati OPS con:
  - risultati scolastici,
  - risultati delle prove Invalsi,
  - risultati extra scolastici;
- partecipazione più attiva con (per esempio):
  - giudizi sulla difficoltà *percepita* dei vari problemi,
  - proposte di nuovi tipi di problemi da inserire nelle prove,
  - suggerimenti per migliorare ed espandere l'iniziativa.

OPS 10 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.

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

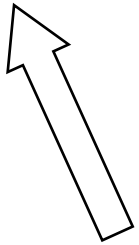
ELEMENTI FORMALI NEI TESTI DEI PROBLEMI

- **argomento o elemento:** <sigla>
  <stringa>
  <numero>
  <termine>
  <lista>
  …

Elementi sintattici 3 esempio 1

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

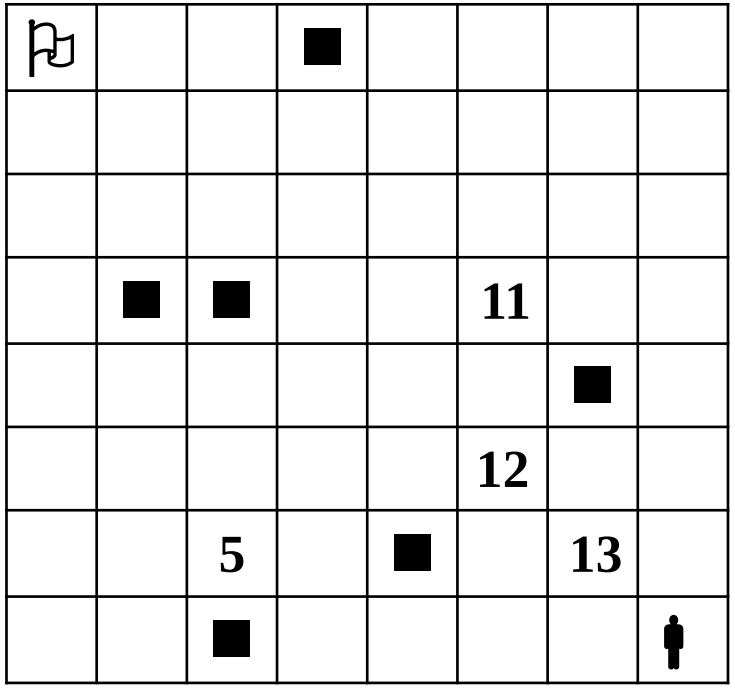
istanze: minerale(m1,30,213)
minerale(m2,35,200)
minerale(m3,38,230)



sintassi e semantica

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

Elementi sintattici 4 esempio 2



la bandiera è nella casella di coordinate [1,8]
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: locuzione usata con significati i più disparati (e non utili per questo contesto). In Internet:

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

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

Problem solving 3

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.

Problem solving 4

http://www.scienze.unibo.it/it/corsi/insegnamenti/insegnamento/2014/366512/

UNIVERSITÀ DI BOLOGNA

Insegnamento: CHIMICA ANALITICA CON LABORATORIO
Corsi di laurea: Chimica
Anno accademico: 2014/2015
…
Conoscenze e abilità da conseguire:
… Al termine del corso, lo studente è in grado di comprendere i principi che definiscono la chimica analitica come "***problem solving***".

Breve storia.

1945:

- George Polya: *How to solve it*
- Jacques Hadamard: *Essay on the Psychology of Invention in the Mathematical Field*,
- Karl Duncker: *On Problem Solving* (postumo)
- Max Wertheimer: *Productive Thinking* (postumo)

Perché nel 1945?

Ragione occasionale:

7/8-9 maggio, 15 agosto/2 settembre

aumentano le pubblicazioni

Ragione profonda:

dall'inizio del secolo aumentano i problemi che la produzione (bellica in particolare) chiede di risolvere (ai matematici)

Problem solving 6 storia 3

George Polya, "*How to solve it*"

- a chi si rivolgeva?
  - ai "futuri matematici" (meglio, a chi studiava matematica);
- perché?
  - aumentano i corsi di matematica e chi vi partecipa;
- perché?
  - aumentano i (*particolari*) problemi "matematici" che la produzione industriale (bellica *in primis*), l'ingegneria, la finanza, la contabilità, ecc. pongono (in maniera pressante dall'inizio del secolo) quindi:
  - aumenta il bisogno di solutori dei problemi;
- quali sono i problemi di cui si occupa Polya?

Problem solving 6 storia 4

Quando si parla di problemi in matematica si pensa, per esempio, a:

1. le grandi Accademie:
   - Accademia delle scienze di Berlino,
   - Accademia delle scienze di Parigi,
   - Accademia delle scienze di San Pietroburgo,
   - Accademia reale di scienze, lettere e belle arti del Belgio,

   che, dalla seconda metà del settecento, per tutto l'ottocento, "bandivano" problemi (assegnando premi ai solutori);
2. i problemi di Hilbert; i "problemi del Millennio";

I solutori sono "grandi" matematici (pochi): non ha senso che sia insegnato loro "*how to solve it*".

STRUTTURA DEL LIBRO

Tre parti:

- The phases: - understanding the problem
  - devising a plan
  - carrying out the plan
  - looking back
- A dictionary of techniques

  e.g. backward
- Problems, hints, solutions.

UN (ESEMPIO DI) PROBLEMA CHE CAMBIA TUTTO

- 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 (*First draft of a report on EDVAC*, 1945).

L'informatica (EDVAC, 1951) rivoluziona il mondo:

- entro un lustro si diffonde un nuovo mestiere: il programmatore;
- appaiono (numerosi) linguaggi (artificiali) di programmazione;
- appaiono i primi corsi di informatica (*computer science*);
- esplodono le applicazioni informatiche;
- i problemi posti dal mondo produttivo convergono nel gran fiume dell'informatica;
- "scrivere un programma" e "risolvere un problema" (di quelli visti) diventano sinonimi.

In realtà, quando occorre un programma per risolvere un problema?

- quando la soluzione è molto laboriosa (ci si impiega troppo tempo a mano);
- quando un problema deve essere risolto più volte (con dati diversi: cioè devono essere risolte più istanze del problema).

Comunque nella maggioranza dei casi:

scrivere un programma $\leftrightarrow$ risolvere un problema.

Problem solving 9 storia 8

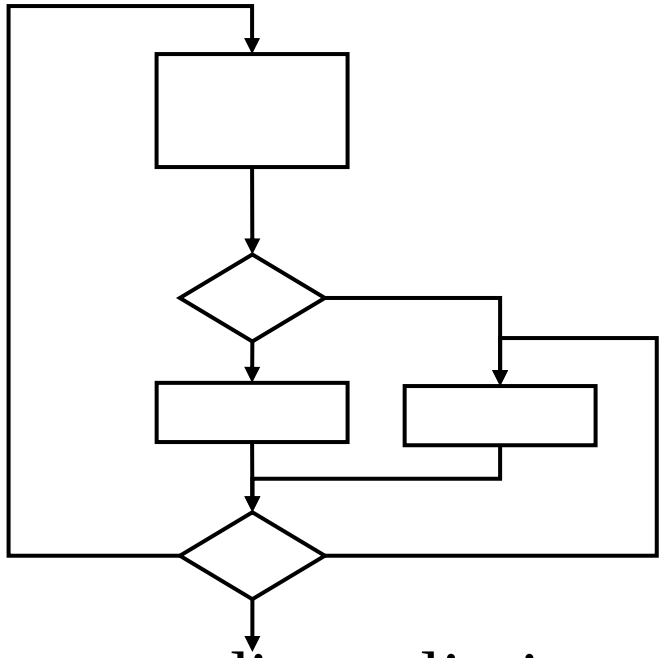
- nella seconda metà degli anni ’50 per scrivere programmi occorre (solamente) essere *smart*;
- successivamente, però, si prende via via coscienza che i programmi sono sempre più scritti male (cioè i problemi sono risolti male);
- **1968**: inizia una crisi (dettagli dopo); si diffonde la consapevolezza che “bisogna fare qualcosa”.
- prima intuizione: per evitare gli errori occorre usare “correttamente” i linguaggi (artificiali) di programmazione $\leftrightarrow$ per evitare gli errori occorre “pensare” correttamente.

Problem solving 11 storia 10

- "*Go to statement considered harmful*", E. Dijkstra, lettera ai CACM, marzo 1968.
- *Nato conference "Software Engineering", Garmisch, October 7-11 1968*. Report del gennaio 1969.
  - perché *software engineering*? (analogia con il miglioramento dei processi di produzione industriale; William Edwards Deming e il controllo di qualità in Giappone.)
  - obbiettivo della conferenza: "ingegnerizzare i processi di produzione del *software*";
- Quali conseguenze?

Problem solving 12 storia 12

flowchart



- Nella preparazione dei programmi, o come stadio preliminare di essi, venivano molto usati i *flowchart*;
- venivano, inoltre, usati i costrutti più sintetici dei vari linguaggi: questo rendeva i programmi di difficile rilettura;
- spesso si faceva una gara nell'impiegare trucchi vari per rendere un programma più "compatto";

Problem solving 14 storia 13

Il processo di ripensamento successivo giunge a un traguardo nel 1976 con

- "*A discipline of programming*" di E. Dijkstra. (disciplina: doppio significato: "corpo di conoscenze" e "regola di comportamento (severa!)"; nel titolo è usata nella seconda accezione.)
- *no flow chart* (rappresentazione grafica bidimensionale);
- il linguaggio (anche di programmazione) è monodimensionale;
- disciplina nell'uso dei linguaggi: programmazione strutturata, *ego-less programming*, …

Problem solving 15 storia 14

- Linguaggio come (strumento di) pensiero;
- occorre *pensare* direttamente nel linguaggio (artificiale) di programmazione (che quindi deve essere opportunamente *disciplinato*);
- si fa strada la consapevolezza del linguaggio (di programmazione) come strumento di (supporto del) pensiero.

slogan:

il *problem solving* (algoritmico)
è un atto linguistico disciplinato!!!

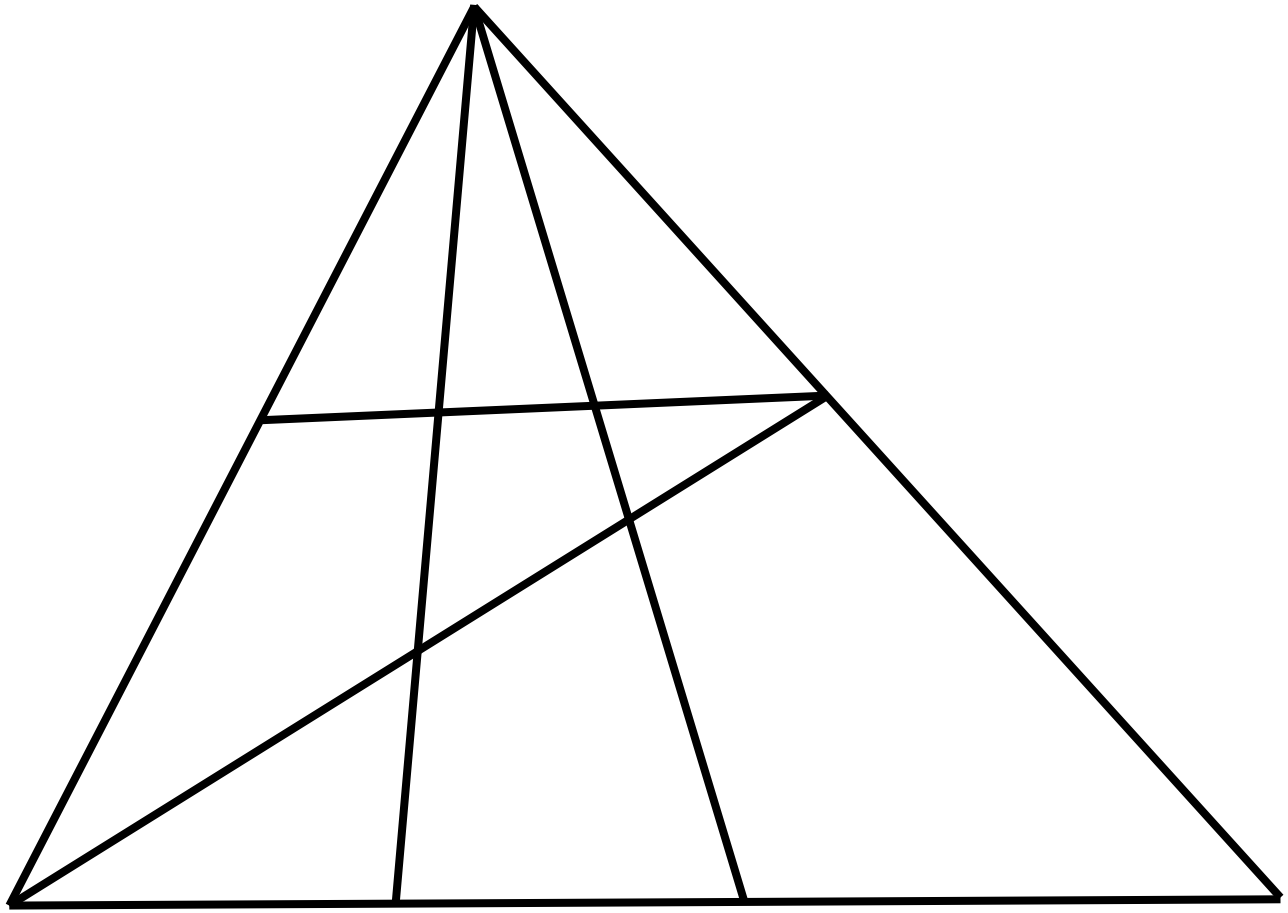
(in senso naïf; comunque: John Langshaw Austin, John Searle, Adolf Reinach, ecc.)

Problem solving 20 dueesempi

Interrompiamo la narrazione storica con due esempi per:

- mostrare la natura linguistica del *problem solving*;
- condividere l'idea di programma;
- servire di base per costruire una definizione (non formale, ma precisa) dell'ambito delle OPS

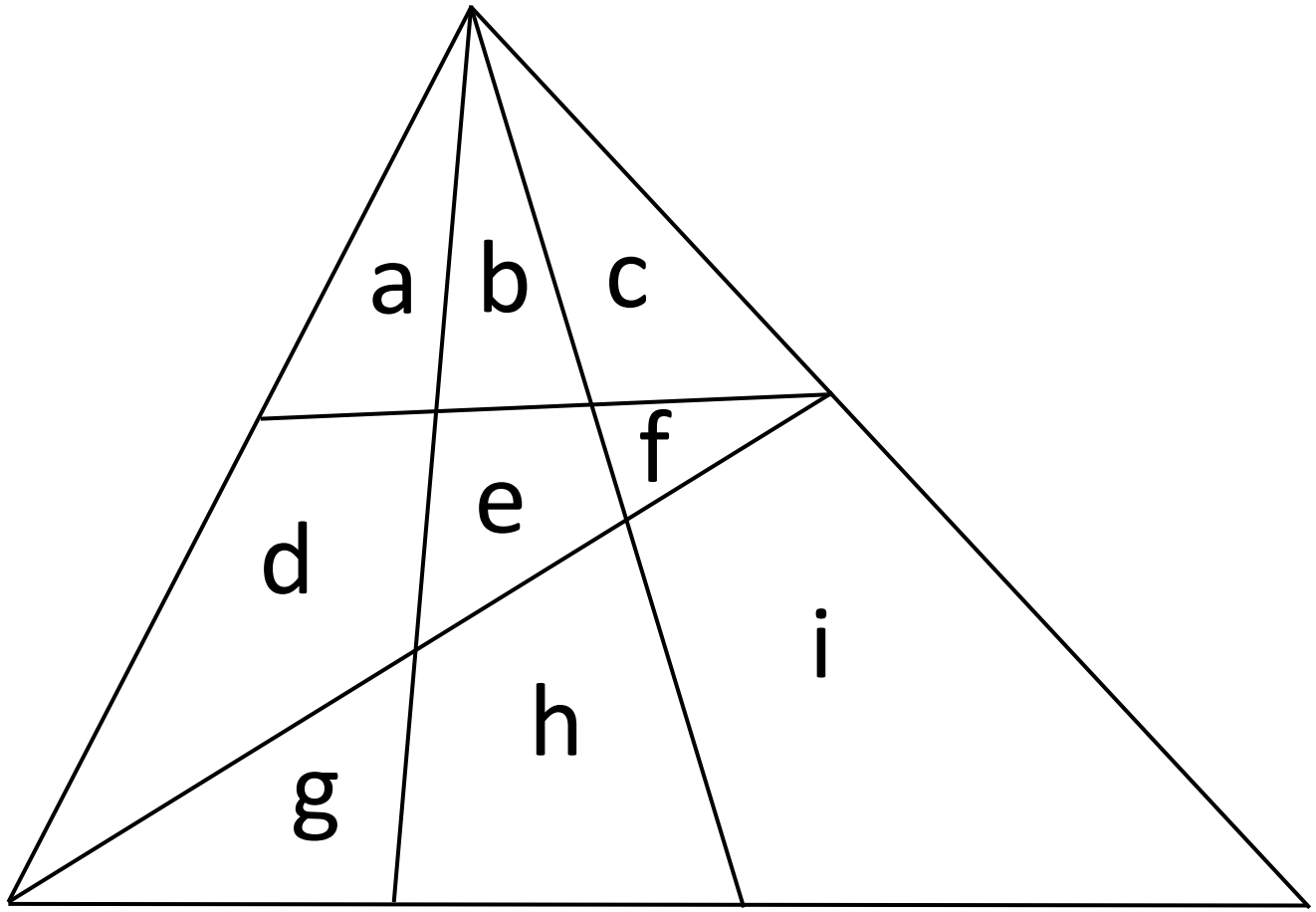
Triangoli 1 il problema



Quanti triangoli compaiono nella figura?

(squadra delle elementari al lavoro; necessità del linguaggio; primo tentativo: ostensione)

Triangoli 2 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*****:**
[a,b] = ~~[b,a]~~)

doppia articolazione!

Triangoli 3 costruire 1

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

TUTTI !!!
CLASSIFICARE

Triangoli 4 costruire 2

	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 rappresentato da [m1,m2] è lo stesso di quello rappresentato da [m2,m1])
- costruzione dello "spazio", di ***tutte*** le 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
[m4,m5]	105	110	no
[m4,m6]	103	105	no
[m5,m6]	108	107	no

TUTTI !!!

massimo valore: soluzione

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 …
- … e l'importanza delle figure?

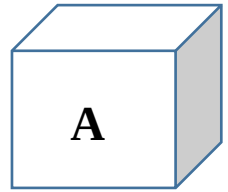
Knapsack 4 altro problema

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.

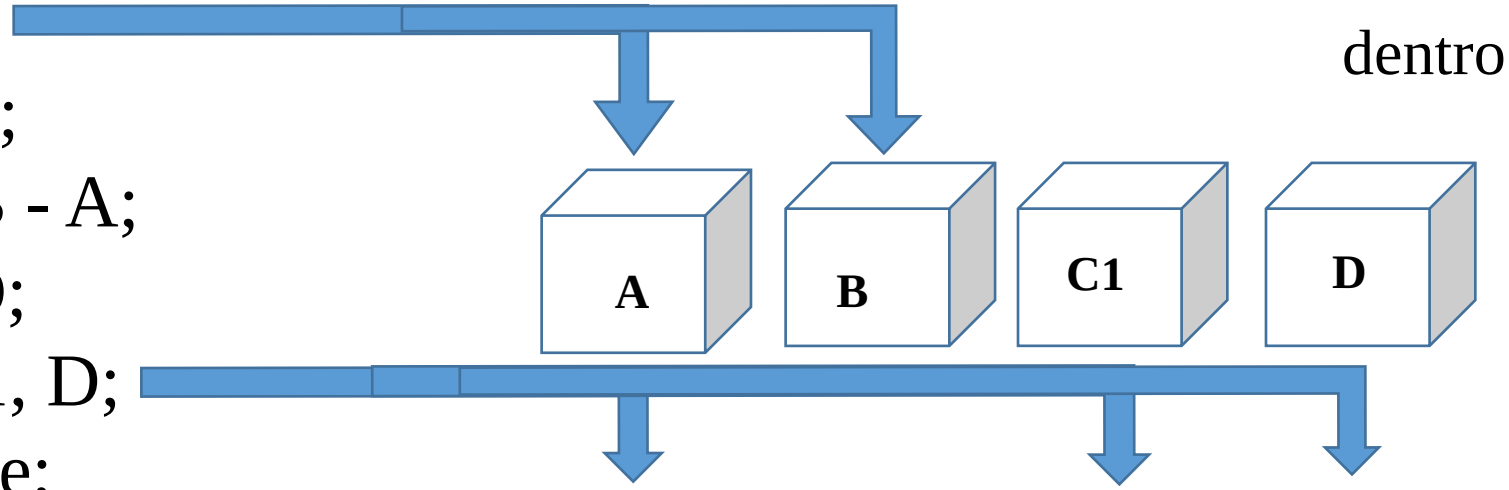
saper usare la forza bruta (è fondamentale!!!), come dare una occhiata intelligente

Modello a scatola 1 primo esempio



scatola con: fuori un nome e dentro un valore

```
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;
```



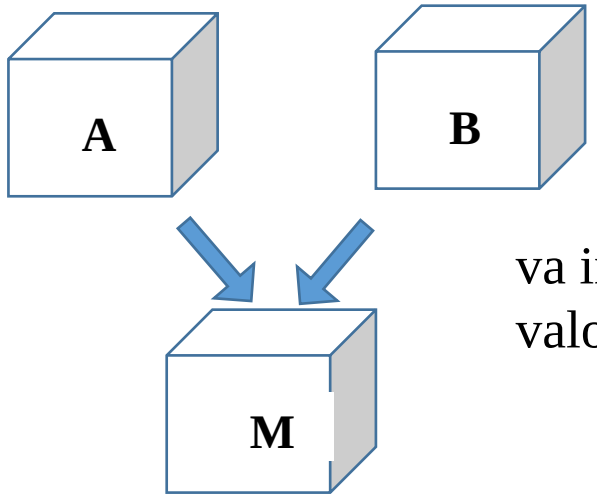
valori di input per A e B**:** 5,9

valori di output per A, C1, D**?** (32,14,18)

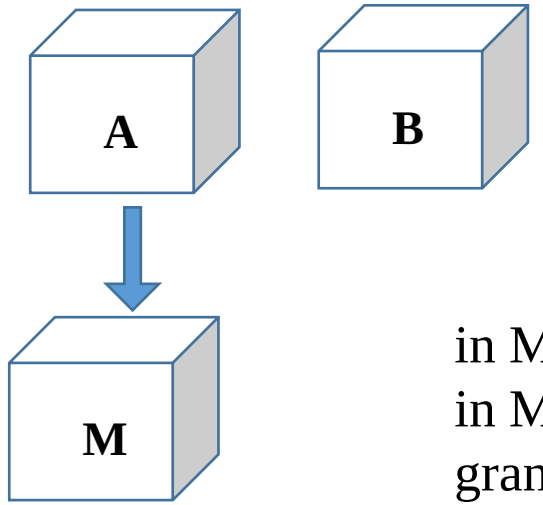
Modello a scatola 2 secondo esempio

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

M ← B;
if A > B then M ← A;
endif;
```



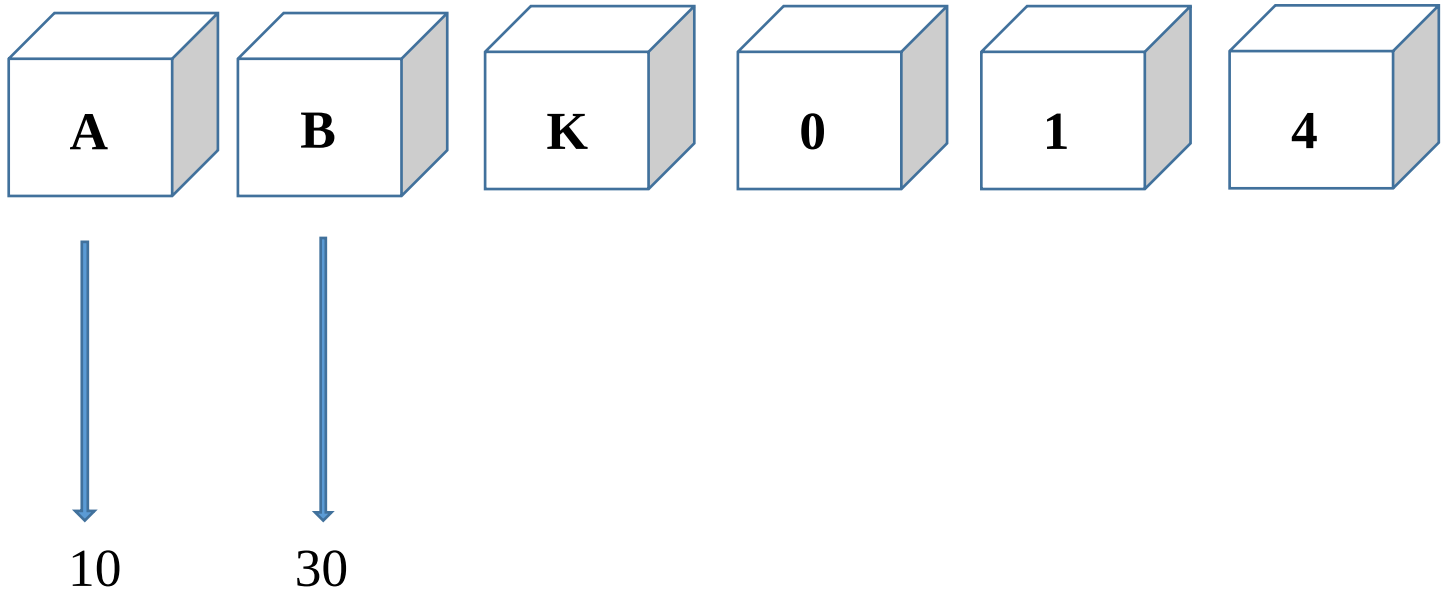
va in M il più grande tra il valore di A e quello di B



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;
```

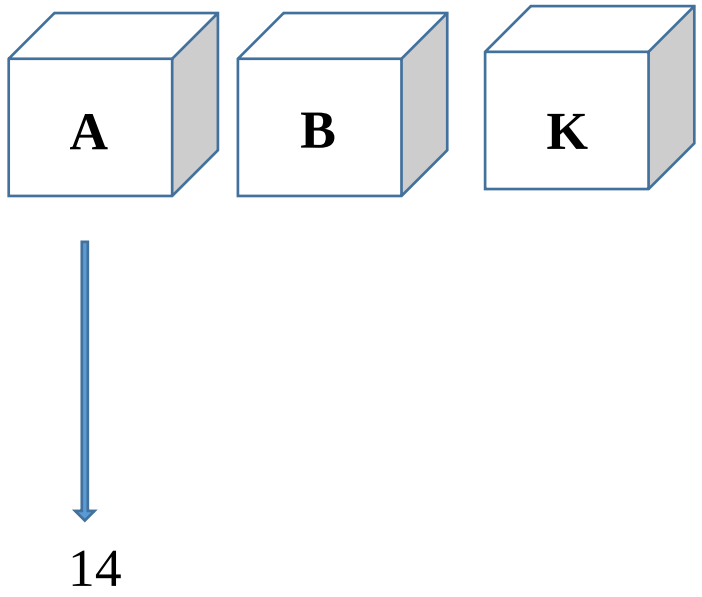


valori successivi di K: 1,2,3,4
valori successivi di A: 0,1,3,6,10
valori successivi di B: 0,1,5,14,30

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;
```



Le costanti non sono più mostrate

valori successivi di K: 0,1,2,3
valori successivi di A: 1,5,14

problema: enunciato, *completamente specificato* in un linguaggio convenzionale, che richiede *una* soluzione;

***classe* di problemi**: insieme di problemi, associato ad un enunciato *non completamente specificato* in un linguaggio convenzionale (richiede un insieme di soluzioni).

Evento: concetto primitivo (significato intuitivo: qualcosa che causa modifiche dell'ambiente);

esempio: operazione aritmetica (causa il risultato, cioè la ''comparsa'' di un *valore*).

- Su un insieme di eventi si può dare un ordine;

esempio: nel tempo.

-Due tipi di eventi: **regolari** (es.: divisione per 3),

irregolari (es.: divisione per zero).

processo: catena di eventi (ordine *totale* su un insieme di eventi);

eseguire un processo: far "accadere" nell'ordine (temporale) gli eventi che lo costituiscono;

processo finito: è finito l'insieme di eventi che lo costituiscono.

programma: testo (*sintatticamente corretto*, in un linguaggio *artificiale*) che "descrive" una famiglia di processi.

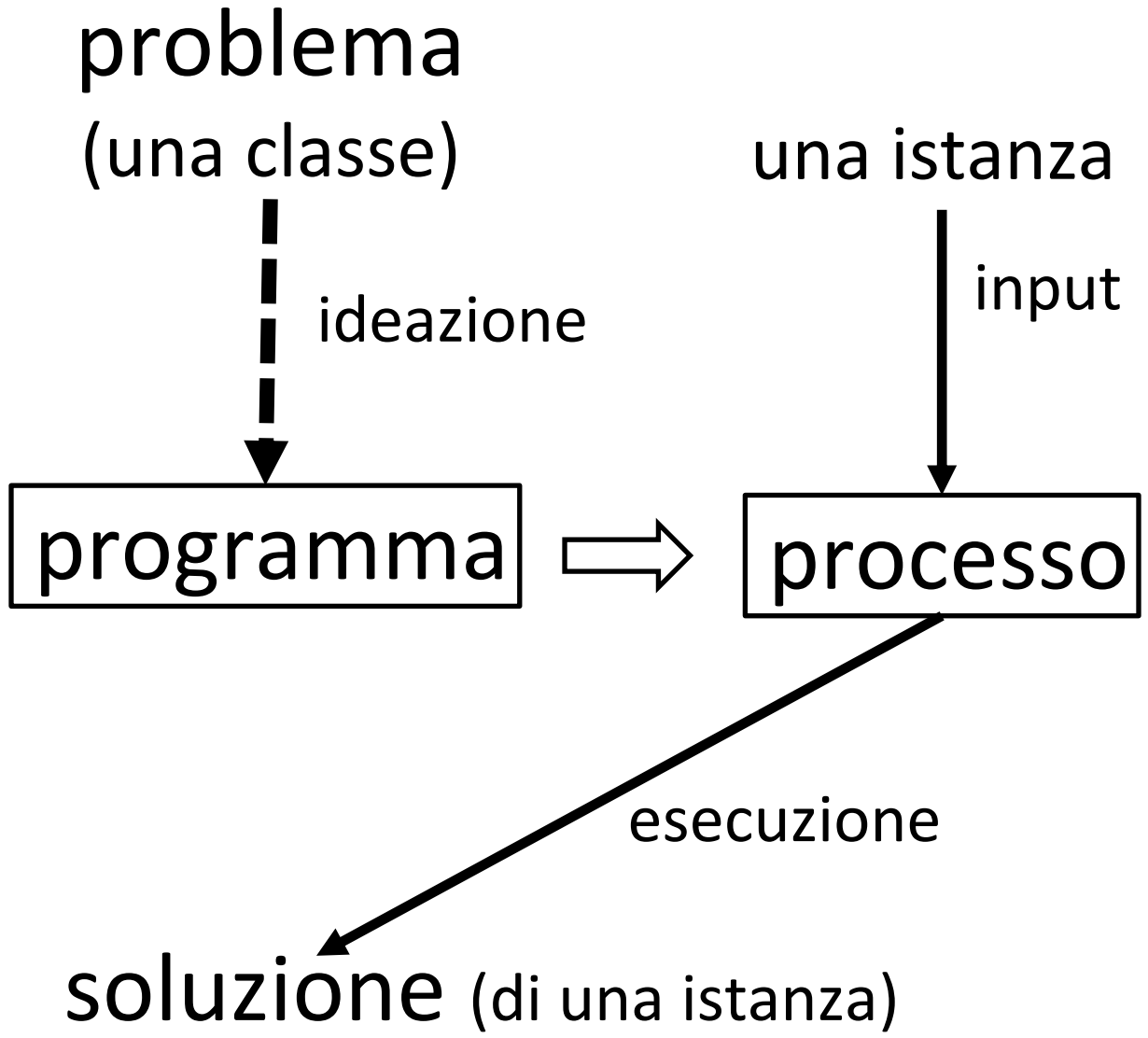
N.B. si dice anche che la famiglia di processi è il *significato* del programma.

programma = testo = successione di costrutti linguistici;

significato del programma = "composizione" del *significato* dei vari costrutti.

Attanti:
- committente
- programmatore
- computer

OII



Relativamente a una **classe di problemi:**

algoritmo: programma cui è associata una famiglia di processi ***finiti*** e con eventi ***regolari***;

programma *corretto*: programma che:

- è un algoritmo,
- descrive dei processi che generano la soluzione (corretta) di ciascun elemento della classe.

Il ***linguaggio (artificiale) di programmazione*** deve avere le "caratteristiche" per:

- descrivere gli eventi (e quindi i processi),

- supportare l'*ideazione* di un programma corretto.

Doppia natura: comincia a intravvedersi nel 1968; appare evidente negli anni '90

Problema algoritmico: problema (appartenente a una classe) per cui la soluzione si può trovare con un programma (corretto).

Pensiero algoritmico: lo strumento mentale per risolvere i problemi algoritmici, nel modo visto per esempio per le OII.

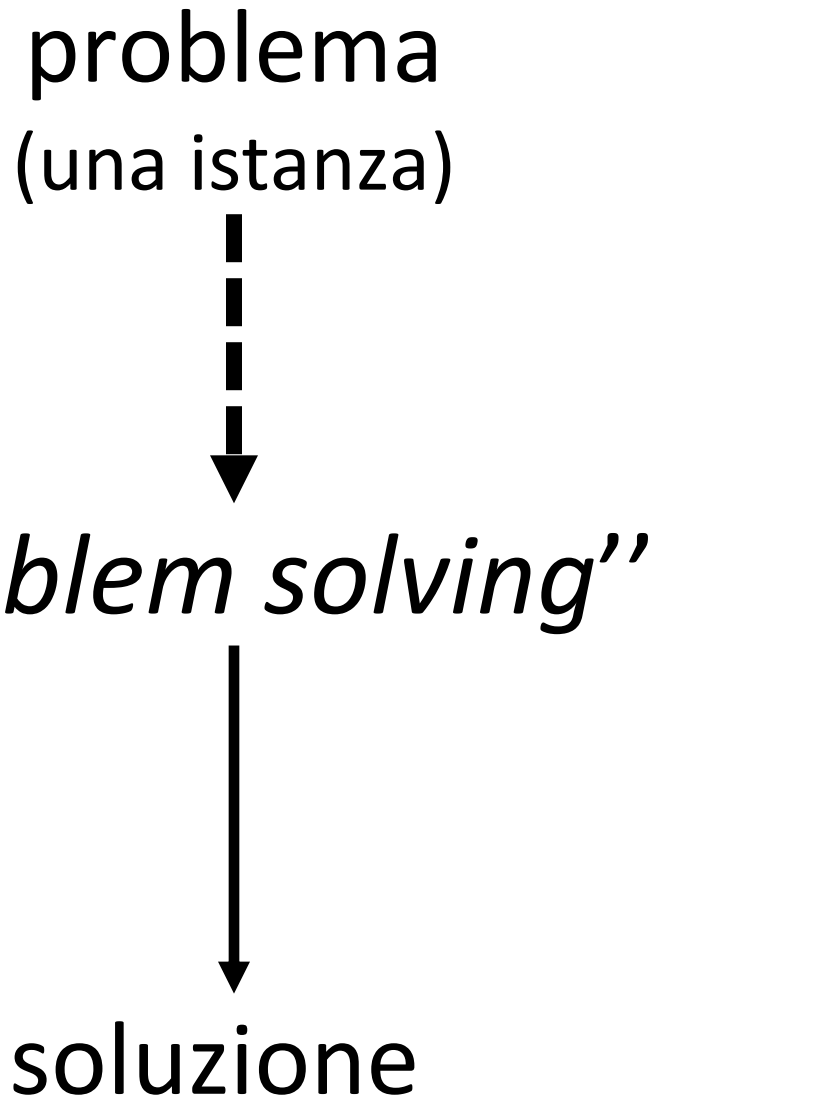
Problem solving 8- schema 2

Attanti:

- committente
- solutore

"amalgama" di ideazione
e processo

OPS



Pensiero computazionale: lo strumento mentale per risolvere i problemi algoritmici, nel modo visto, per le OPS

La maggioranza dei problemi delle OPS sono algoritmici (eccezione: il problema di Italiano)

Grafi 1 il problema

Un grafo (non orientato), che corrisponde alla rete di strade che (senza intersecarsi) 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)

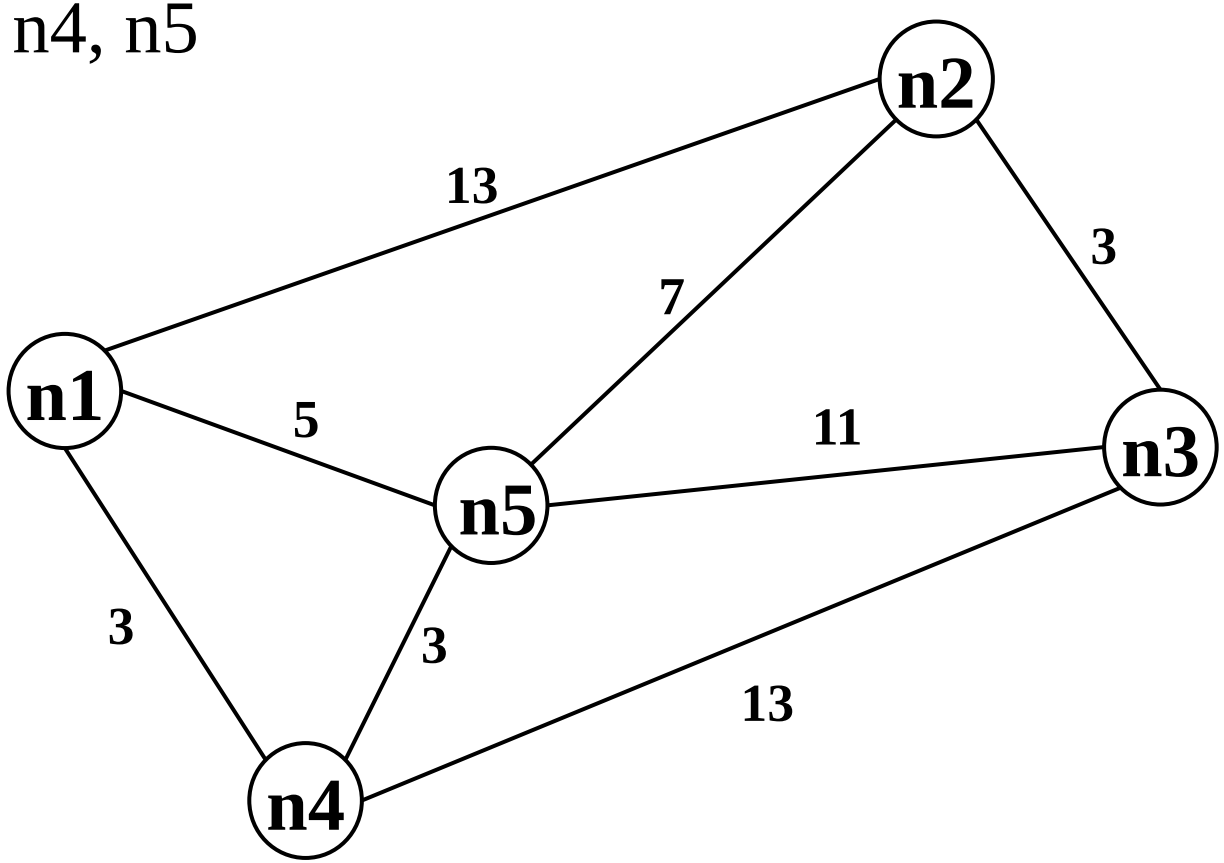
N.B. definizione: arco(<nodo1>,<nodo2>,<distanza>)

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

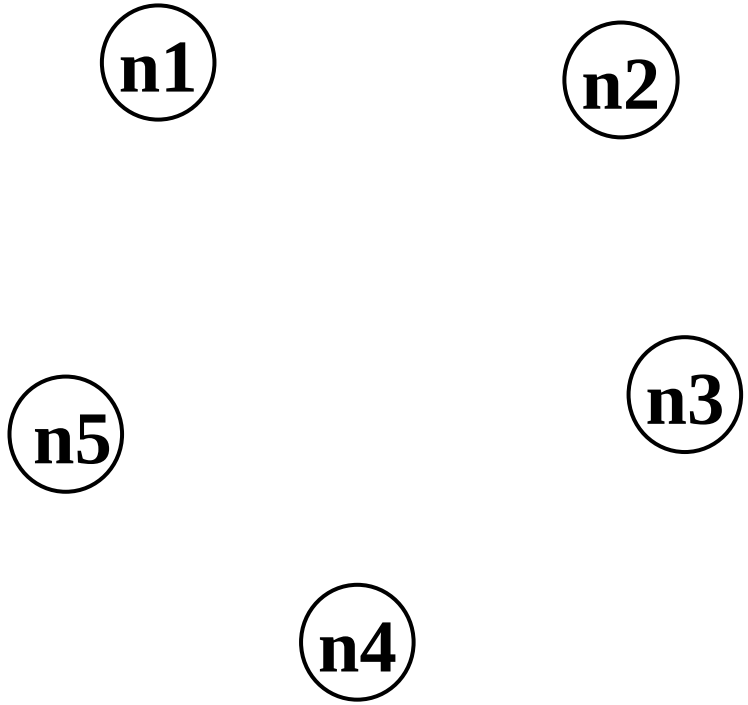


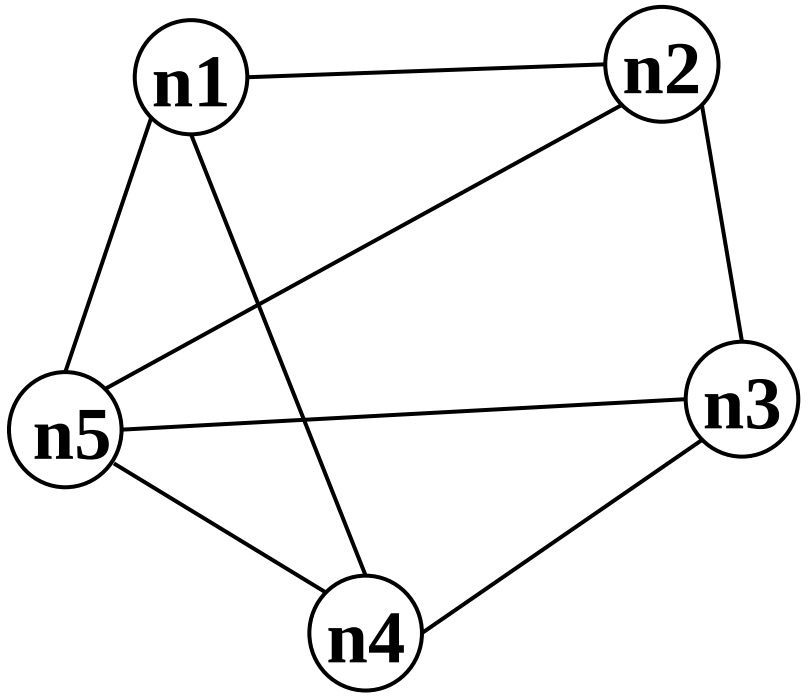
Si noti che le lunghezze degli archi che compaiono nei termini (che rappresentano delle strade) *non* sono necessariamente proporzionali a quelle degli archi

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)





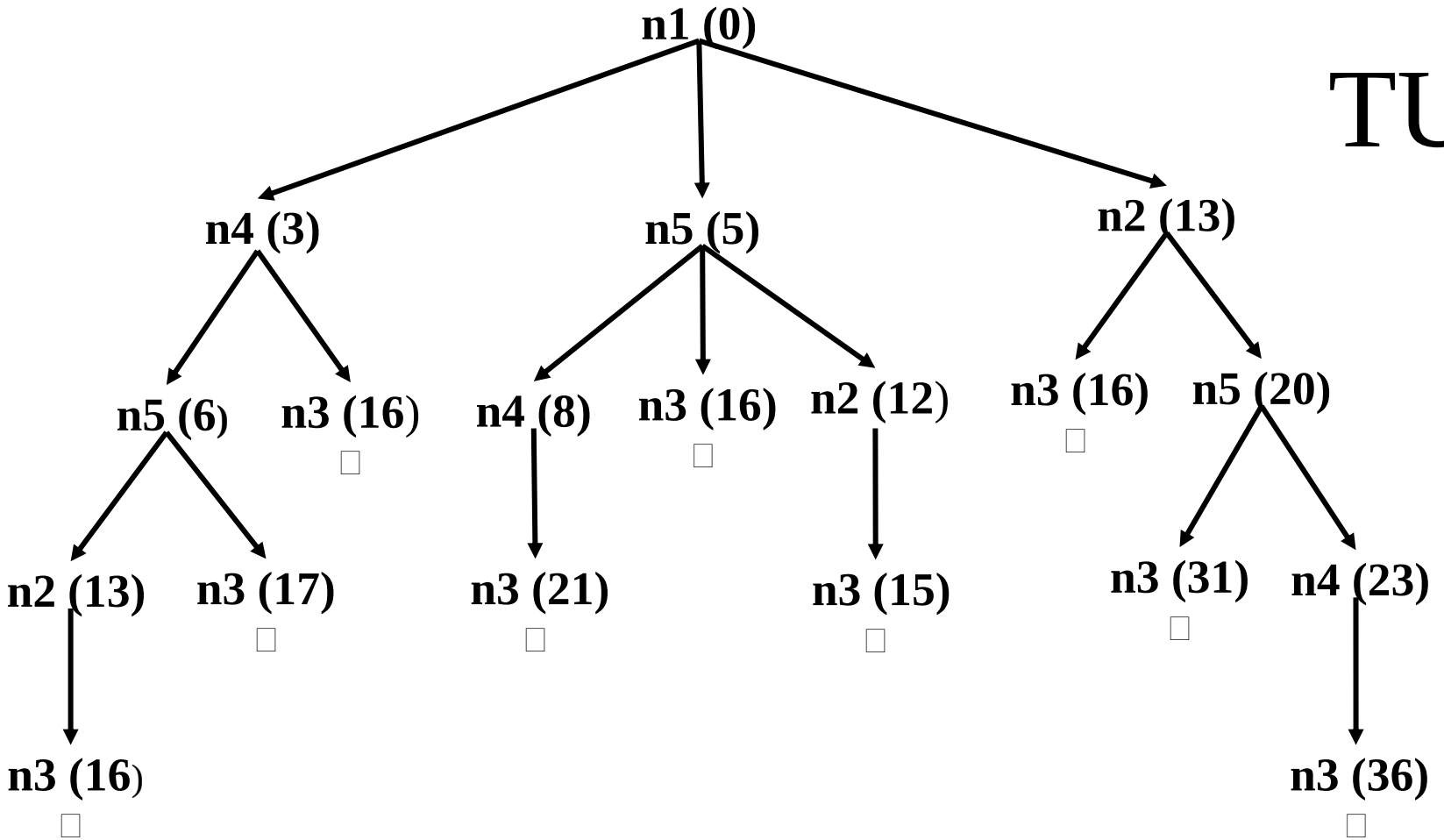
Grafi 4 costruire

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



TUTTI !!!

Regole 1 il problema

Siano date le seguenti regole (di deduzione):

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>)

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 paradigma

Esempi:

Dal teorema di Pitagora si ricavano due regole:

- dati i due cateti di un triangolo rettangolo, l'ipotenusa è uguale alla radice quadrata della somma dei quadrati dei cateti;
- dato un cateto e l'ipotenusa di un triangolo rettangolo, l'altro cateto è uguale alla radice quadrata della differenza tra il quadrato dell'ipotenusa e il quadrato del cateto dato.

regola(pitagora1,[cateto1,cateto2],ipotenusa)

regola(pitagora2,[cateto1,ipotenusa],cateto2)

N.B. per semplicità usiamo i nomi di elementi geometrici per la loro misura.

Regole 3 paradigma

Primo teorema di Euclide: in un triangolo rettangolo il quadrato di un cateto è uguale al prodotto della sua proiezione sull'ipotenusa per l'ipotenusa stessa.

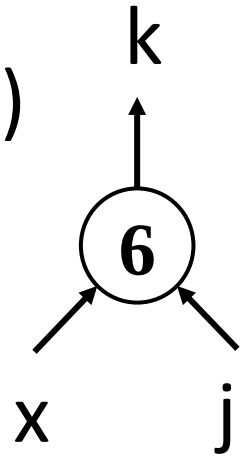
- regola(euclide1a,[proiezionecateto,ipotenusa], cateto)
- regola(euclide1b,[proiezionecateto,cateto],ipotenusa)
- regola(euclide1c,[ipotenusa, cateto],proiezionecateto)

Secondo teorema di Euclide: in un triangolo rettangolo il quadrato dell'altezza relativa all'ipotenusa è uguale al prodotto delle proiezioni dei cateti sull'ipotenusa.

- regola(euclide2a,[proiezionecateto1,proiezionecateto2],altezza)
- …

Regole 4 interpretazione

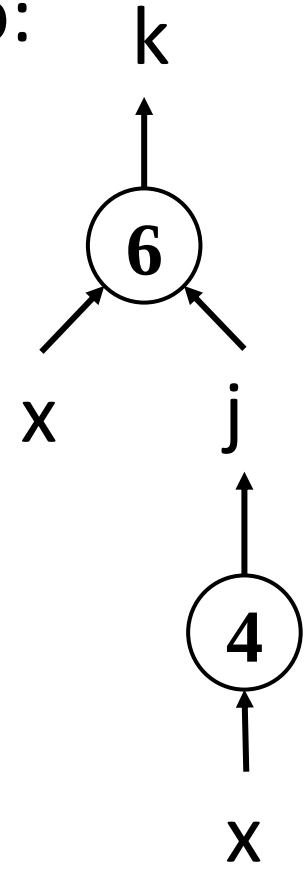
regola(6,[x,j],k)



regola(4,[x],j)

Procedimento:

L1 è [4,6]



Regole 5 metodi di deduzione

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.**

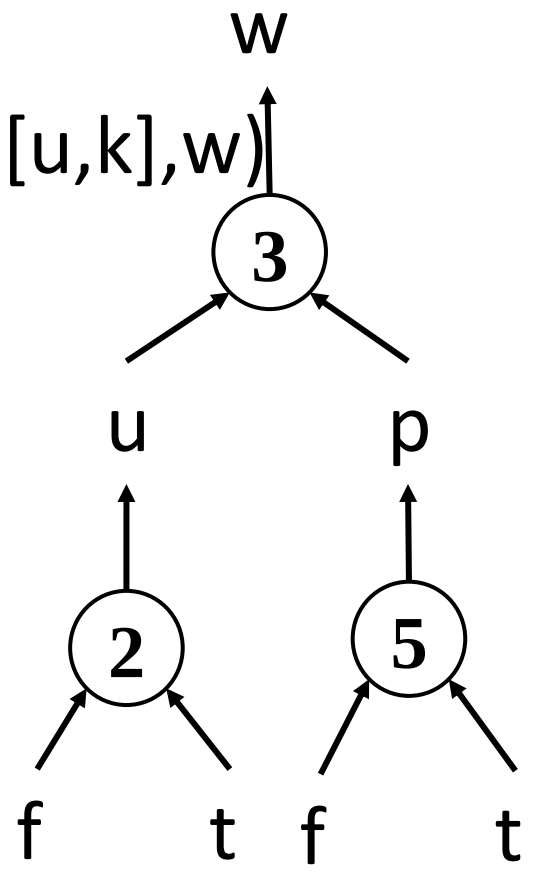
Regole 6 esempio di deduzione

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])

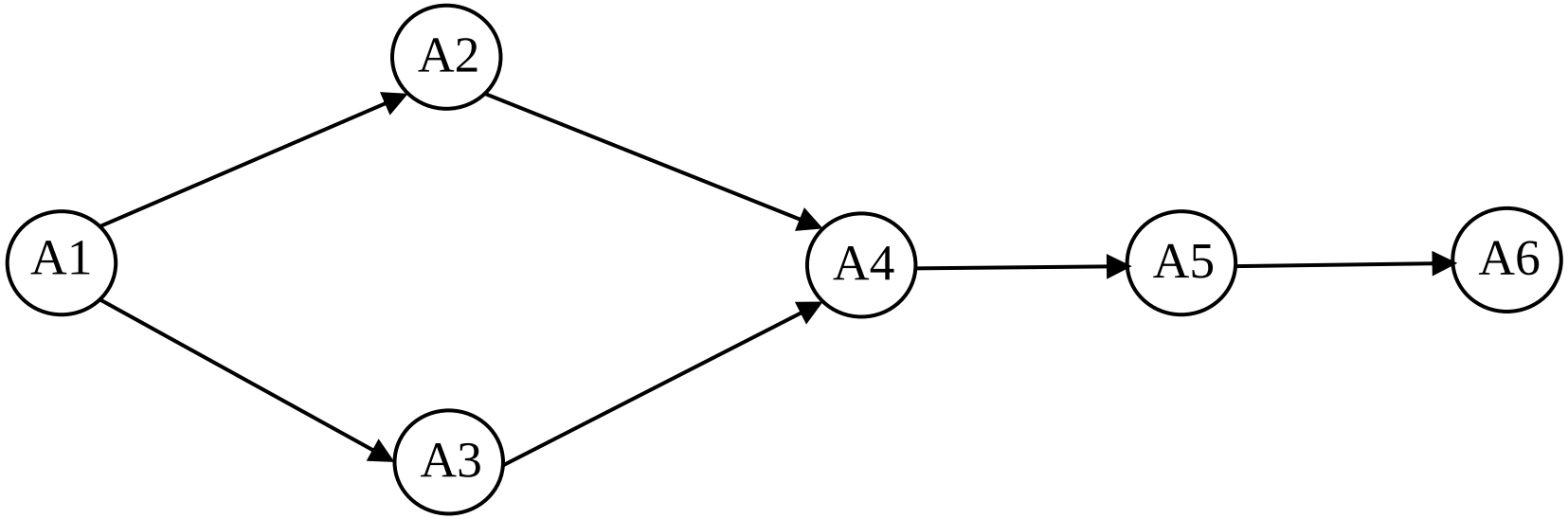
Project management 1 problema

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

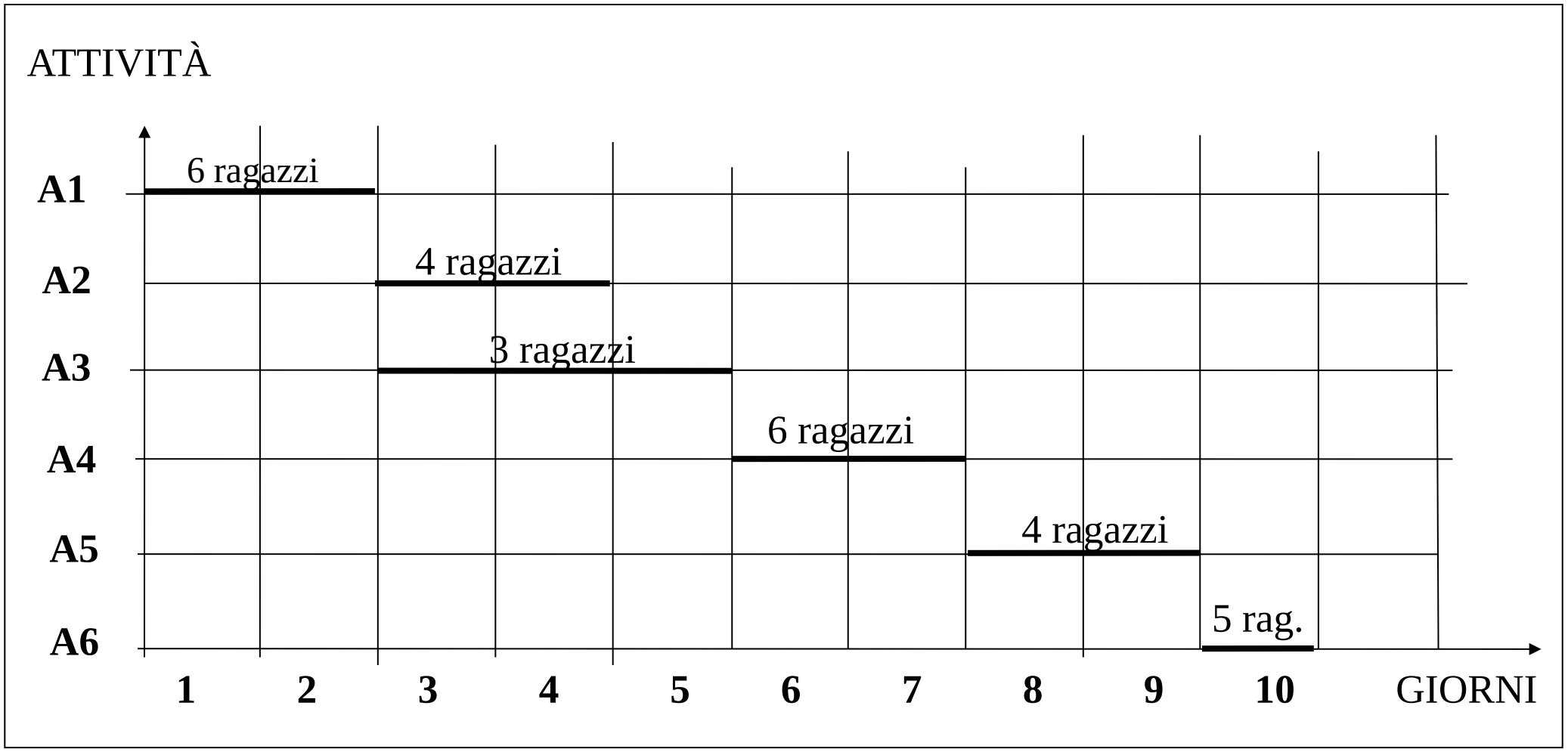
ATTIVITÀ	RAGAZZI	GIORNI
A1	6	2
A2	4	2
A3	3	3
A4	6	2
A5	4	2
A6	5	1

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

precedenza(A1,A2)
precedenza(A1,A3)
precedenza(A2,A4)
precedenza(A4,A5)
precedenza(A3,A4)
precedenza(A5,A6)

ALTRI 2 1

A knockout tournament is a series of games. Two players compete in each game; the loser is “knocked out” (i.e. doesn’t play anymore), the winner carries on. The winner of the tournament is the player that is left after all other players have been knocked out. Suppose there are 1234 players in a tournament. How many games are played before the tournament winner is decided?

ALTRI 2 2

TIPS FOR THE SOLUTION

At the beginning (after 0 games) we had 1234 players; after every game the number of players decreases by one, hence the following schema holds.

	GAMES	PLAYERS (LEFT)
beginning	0	1234
step 1	1	1233
….		
step N – 1	N – 1	1

INVARIANT: GAMES + PLAYERS = 1234

ALTRI 2 3

Assume you have a chocolate bar consisting, as usual, of a number N of small squares arranged in a rectangular pattern. Your task is to split the bar into the small squares (always breaking along the lines between the squares) with a *minimum* number of breaks. How many will it take?

ALTRI 2 4

TIPS FOR THE SOLUTION

At the beginning (after 0 breaks) we had 1 piece. After 1 break we got 2 pieces. After every break the number of pieces increases by 1. So the following schema holds:

	BREAKS	PIECES
beginning	0	1
step 1	1	2
…		
step N-1	N - 1	N final state

INVARIANT: PIECES – BREAKS = 1.

SEMINARI OPS

operazione di marketing (o vendita?)

"L'attività di vendita si incentra sulle necessità del venditore/produttore, quella di marketing sulle necessità dell'acquirente", Theodore Levitt, Harvard Business School; Marketing Myopia (1960), The marketing imagination (1983).

Problem solving (ultimo) commento

Solving problems brings pleasure.

When I say “problem solving” …, I mean any cognitive work that succeeds; it might be understanding a difficult passage of prose, planning a garden, or sizing up an investment opportunity. There is a sense of satisfaction, of fulfillment, in successful thinking.

In the last ten years neuroscientists have discovered that there is overlap between the brain areas and chemicals that are important in learning and those that are important in the brain’s natural reward system.

by Michael B. Horn.