# Come barare alla gara a squadre con Python / Sagemath

Federico Poloni

Università di Pisa

Incontri Olimpici 2018, Iseo (BS)

# Introduzione

"Fare i conti al computer" è diventato uno strumento utilissimo nella mia cassetta degli attrezzi per preparare gare a squadre.

Ovviamente c'è sempre dietro anche una dimostrazione matematica, ma avere un secondo controllo è utilissimo.

Gli stessi strumenti si possono usare per spiegare cose con esempi sottomano, ad es. combinatoria.

# Python/Sagemath

Sagemath (o Sage) è un ambiente che permette di fare matematica (conti numerici, simbolici, algebrici) con Python.

- Python 2
- Interfaccia interattiva
- Tante librerie matematiche integrate con un'interfaccia comune
- "Zucchero sintattico" utile ai matematici, ad es. ^ per l'elevamento a potenza.

# Vantaggi/svantaggi

- Python molto diffuso, anche come linguaggio per l'insegnamento.
- Molto leggibile (e scrivibile); facile gestire programmi lunghi (cfr. Mathematica).
- Molte librerie, buono anche per farci altro.
- Notazione che rispecchia la matematica (es. comprehensions).
- Svantaggio: performance non sempre al livello massimo ('problema dei due linguaggi'). Ci sono tentativi di risolvere il problema.

# Come calcolatrice

```
sage: 3^100
515377520732011331036461129765621272702107522001
sage: (1 + sqrt(2))^10
(sqrt(2) + 1)^10
sage: expand((1 + sqrt(2))^10)
2378*sqrt(2) + 3363
sage: N(expand((1 + sqrt(2))^10))
6725.99985132322
sage: var('a b')
(a, b)
sage: expand((a + b)^10)
a^10 + 10*a^9*b + 45*a^8*b^2 + 120*a^7*b^3 + 210*a^6*b^4 + 252
```

## Il telefono di DOC

Sull'elenco telefonico Matryx HOMFLY trova il numero di $\widehat{DOC}$: esso è pari a $(1) + (1+2) + \cdots + (1+2+\cdots+1918)$. Quante cifre ha?

$$\sum_{n=1}^{1918} \sum_{k=1}^{n} k$$

Via 'calcolo simbolico':

```
sage: var('k n')
(k, n)
sage: sum(k, k, 1, 1918)
1840321
sage: sum(sum(k, k, 1, n), n, 1, 1918)
1177805440
```

Via 'Python puro':

```
S = 0

for n in range(1, 1918+1):
    S = S + sum(range(1, n+1))

print(len(S.digits()))
```

**Warning**

Quali righe stanno dentro il `for` lo dice solo l'indentazione.

**Warning**

`range(a, b)` va da `a` a `b-1`; `b` è il primo escluso.
Sembra fastidioso, ma nel 'grande schema delle cose' è un'idea migliore.
Riduce di molto la quantità di $\pm 1$ arbitrari. Es.
`len(range(a, b)) == b-a`.

## Il Grande Almanacco delle Olimpiadi di Matematica ⋆

Ad ogni edizione delle Olimpiadi di Matematica, a partire dalla numero zero, Biff Tauber scommette sul vincitore grazie al Grande Almanacco. La vincita che ottiene all'edizione $n$ è di $a_n$ dollari, dove $a_0 = 0$, e per ogni intero $n$ valgono le relazioni $a_{3n} = a_n$, $a_{3n+1} = a_n - 1$, e $a_{3n+2} = a_n + 2$. Finita l'edizione numero 2018, Biff realizza che, dall'inizio delle scommesse, ha guadagnato una montagna di dollari. Quanti, di preciso?

```
a = [None] * 2019
a[0] = 0
for n in range(2019/3):
        a[3*n] = a[n]
        a[3*n+1] = a[n] - 1
        a[3*n+2] = a[n] + 2

print(sum(a[0:2019]))
```

# Commenti

- Abbiamo *preallocato* una lista di 2019 elementi. Più comodo in questo caso che non `a.append()`.
- Le liste possono contenere oggetti arbitrari (anche diversi). Una lista lunga `n` contiene elementi `a[0], a[1], ... , a[n-1]`.
- Funziona anche `a[-1]` = ultimo elemento, `a[-2]` = penultimo...
- `a[inizio:fine]` restituisce la sottolista nel 'range' `inizio:fine` (con le solite regole sui range!).

Ma, soprattutto: la logica del programma corrisponde abbastanza bene alla matematica.

# Oggetti combinatorici

```
sage: Combinations(4, 2)
Combinations of [0, 1, 2, 3] of length 2
sage: print Combinations(4, 2)
Combinations of [0, 1, 2, 3] of length 2
sage: list(Combinations(4, 2))
[[0, 1], [0, 2], [0, 3], [1, 2], [1, 3], [2, 3]]
sage: len(Permutations(4))
24
sage: list(Permutations(4, 2))

[[1, 2], [1, 3], [1, 4], [2, 1], [2, 3], [2, 4], [3, 1], [3, 2
sage: Permutations([1, 3, 5, 7, 9])
...
sage: Permutations('mamma')
...
```

# Iterare e comprehensions

```
sage: for p in Permutations(4):
....: print p[0],

1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3 3 4 4 4 4 4 4
```

Sintassi molto concisa per raccogliere in una lista questi risultati:

```
sage: [p[0] for p in Permutations(4)]
[1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 4, 4, 4
```

Fa la stessa identica cosa di sopra: "scorre" `Permutations(4)` e restituisce `p[0]`.

$$\{p(0) \colon p \in \mathfrak{S}_4\}.$$

# Comprehensions condizionali

```
sage: [p for p in Permutations(4) if p[0] == 2]

[[2, 1, 3, 4],
 [2, 1, 4, 3],
 [2, 3, 1, 4],
 [2, 3, 4, 1],
 [2, 4, 1, 3],
 [2, 4, 3, 1]]
```

stessa cosa di

```
for p in Permutations(4):
    if p[0] == 2:
        print(p)
```

# Verificare relazioni

```
sage: S1 = [p for p in Permutations(4) if p[0] == 2]
[[2, 1, 3, 4],
 [2, 1, 4, 3],
 [2, 3, 1, 4],
 [2, 3, 4, 1],
 [2, 4, 1, 3],
 [2, 4, 3, 1]]
sage: S2 = [[2,] + list(p) for p in Permutations([1,3,4])]
[[2, 1, 3, 4],
 [2, 1, 4, 3],
 [2, 3, 1, 4],
 [2, 3, 4, 1],
 [2, 4, 1, 3],
 [2, 4, 3, 1]]
sage: S1 == S2
True
```

# Contare doppie coppie

Problema classico di combinatoria — qual è la probabilità di avere una doppia coppia a poker?

```
sage: valori = range(1,14)
sage: semi = 'CFQP'
sage: mazzo = cartesian_product([valori, semi])
sage: list(mazzo)
[(1, 'C'),
 (1, 'Q'),
 (1, 'F'),
 (1, 'P'),
 (2, 'C'),
...
 (13, 'Q'),
 (13, 'F'),
 (13, 'P')]
```

```
sage: mani = Combinations(mazzo, 5)
sage: mani.cardinality()
2598960
sage: [mani[i] for i in range(5)]
[[(1, 'C'), (1, 'Q'), (1, 'F'), (1, 'P'), (2, 'C')],
 [(1, 'C'), (1, 'Q'), (1, 'F'), (1, 'P'), (2, 'Q')],
 [(1, 'C'), (1, 'Q'), (1, 'F'), (1, 'P'), (2, 'F')],
 [(1, 'C'), (1, 'Q'), (1, 'F'), (1, 'P'), (2, 'P')],
sage: m2 = Combinations(range(1,14)*4, 5)
sage: m2.cardinality()
6175
sage: [m2[i] for i in range(5)]
[[1, 1, 1, 1, 2],
 [1, 1, 1, 1, 3],
 [1, 1, 1, 1, 4],
 [1, 1, 1, 1, 5],
 [1, 1, 1, 1, 6]]
```

Ma gli elementi di `m2` non sono equiprobabili!

```
sage: def elimina_semi(mano):
....: return list(valore for valore,seme in mano)
....:
sage: elimina_semi(mani[0])
[1, 1, 1, 1, 2]
sage: elimina_semi(mani[1])
[1, 1, 1, 1, 2]
sage: [m for m in mani if elimina_semi(m) == m2[0]]
# comincia a metterci troppo...
sage: valori = range(1,5)
sage: mazzo = cartesian_product([valori, semi])
sage: mani = Combinations(mazzo, 5)
sage: [m for m in mani if elimina_semi(m) == m2[0]]
[[(1, 'C'), (1, 'F'), (1, 'Q'), (1, 'P'), (2, 'C')],
 [(1, 'C'), (1, 'F'), (1, 'Q'), (1, 'P'), (2, 'F')],
 [(1, 'C'), (1, 'F'), (1, 'Q'), (1, 'P'), (2, 'Q')],
 [(1, 'C'), (1, 'F'), (1, 'Q'), (1, 'P'), (2, 'P')]]
```

```
sage: [m for m in mani if elimina_semi(m) == m2[11]]
[[(1, 'C'), (1, 'F'), (2, 'C'), (2, 'F'), (4, 'C')],
 [(1, 'C'), (1, 'F'), (2, 'C'), (2, 'F'), (4, 'F')],
 [(1, 'C'), (1, 'F'), (2, 'C'), (2, 'F'), (4, 'Q')],
 [(1, 'C'), (1, 'F'), (2, 'C'), (2, 'F'), (4, 'P')],
 [(1, 'C'), (1, 'F'), (2, 'C'), (2, 'Q'), (4, 'C')],
 ...
sage: len([m for m in mani if elimina_semi(m) == m2[11]])
144
sage: 4*binomial(4,2)*binomial(4,2)
144
```

```
sage: def frequenze(mano):
....: """restituisce le frequenze di ogni valore in una mano"
....: valori = elimina_semi(mano)
....: return [valori.count(v) for v in set(valori)]
....:
sage: def doppia_coppia(mano):
....: """restituisce vero se la mano e' una doppia coppia"""
....: return sorted(frequenze(mano)) == [1,2,2]
....:
sage: doppia_coppia(mani[200])
True
sage: doppia_coppia(mani[199])
False
sage: mani[199]
[(1, 'C'), (1, 'F'), (2, 'F'), (2, 'Q'), (2, 'P')]
sage: mani[200]
[(1, 'C'), (1, 'F'), (2, 'F'), (2, 'Q'), (3, 'C')]
```

```
sage: sum(1 for m in mani if doppia_coppia(m))
1728
sage: mani.cardinality()
4368
sage: binomial(len(valori),2) * binomial(len(semi),2) *
....: binomial(len(semi),2) * (len(valori)-2) * len(semi)
1728
```

## L'incontro

Nel Maggio di moltissimi anni fa, diversi matematici si ritrovarono in una locanda; si accorsero subito di essere esattamente tanti quanti gli interi $n$, compresi tra 100 e 10000, tali che il loro fattoriale $n!$ è un multiplo di $2^{n-1}$. Dopo essersi contati, decisero che erano nel giusto numero per intraprendere il pellegrinaggio alla tomba di Archimede. Quanti erano?

```
sage: L = [n for n in range(1,10001)
    if (2^(n-1)).divides(factorial(n))]; print(len(L))
```

## Addestramento

Lungo il tragitto i pellegrini si trovarono nei pressi di un campo d'addestramento, nel quale alcuni soldati si allenavano a gruppetti, di dimensioni tutte diverse; notarono subito che il minimo comune multiplo della dimensione dei vari gruppetti era esattamente 160. Quante sono le possibili suddivisioni che essi possono aver visto?

```
sage: sum(1 for S in subsets(divisors(160)) if lcm(S)==160)
```

## Circuiti difettosi

I circuiti del tempo sono stati danneggiati da un fulmine: ora la DeuLerean può raggiungere solo gli anni di quattro cifre tali che la cifra delle migliaia sia maggiore o uguale alla somma delle altre tre. "Poco male" dice $\widehat{DOC}$, e calcola rapidamente in quanti anni diversi può viaggiare. Che numero trova? *Si intende che un numero di quattro cifre ha la cifra delle migliaia non nulla.*

```
def is_acceptable(n):
        d = Integer(n).digits()
        return d[3] >= d[0] + d[1] + d[2]

print len([n for n in range(1000,10000) if is_acceptable(n)])
```

Pattern tipico di molti esercizi, "conta gli $n$ che soddisfano questa condizione."

## Banditi matematici ⋆

"Whisky per i miei $n$ uomini!" tuona Bu4 Kampen nel saloon. "E quanti sono?" chiede il barista. "Ti dirò solo che $n$ è un intero positivo con esattamente 12 divisori positivi $1 = d_1 < d_2 < \cdots < d_{12} = n$, e che $d_{d_4-1} = d_8(d_1 + d_2 + d_4)$". Non volendo fare altre domande, per ogni possibile $n$ il barista prepara un vassoio con $n$ bicchieri. Quanti bicchieri riempie?

```
def check(n):
        d = divisors(n)
        d.insert(0, -1) #hack for one-based indexing
        return len(d) == 13 and d[4]-1<=12 and d[d[4]-1] == d[8

L = list(n for n in range(1,10000) if check(n))
```

## Direttamente nella tua casella

Il prof. Fredholm mostra a Frege una mappa dell'universo, a forma di scacchiera infinita bidimensionale; ogni casella rappresenta un settore del piano galattico. Indica col dito il centro del settore corrispondente alla Terra, e spiega: "La nostra astronave per le consegne ha un motore di mia invenzione che utilizza materia oscura come carburante. Con due palline di materia oscura, prodotte dal nostro Mordaglia, può spostarsi di un settore in orizzontale o in verticale; con tre può spostarsi di un settore in diagonale. Partendo dalla nostra base sulla Terra, possiamo fare consegne in qualunque settore raggiungibile con 60 palline di carburante o meno. Incluso quello di partenza, ovviamente", aggiunge ridacchiando. In quanti settori diversi può fare consegne la Planar Express?

```
def distance_from_origin(a,b):
                a = abs(a); b = abs(b)
                return 3*min(a, b) + 2*abs(a-b)

len([(a,b) for a in range(-40,40) for b in range(-40,40) if
                        distance_from_origin(a, b) <= 60])
```

## All your base are belong to us

Nella galassia, esistono molte specie intelligenti; esse hanno un numero di dita compreso tra 4 e 12, e, quindi, tutte le basi di numerazione tra 4 e 12 sono in uso. In quali di queste basi $b$ il numero $2013_b$ (cioè, quello che in base $b$ ha nell'ordine le cifre $2, 0, 1, 3$) ha lo stesso numero di divisori di $2013_{10}$? *Dare come risposta il prodotto di tutte queste basi (inclusa $b = 10$).*

```
prod(B for B in range(4,13) if
    len(divisors(ZZ('2013',base=B))) ==
    len(divisors(ZZ('2013',base=10))))
```

## Terzo grado

Il temibile Mond Vander sta interrogando la principessa Liea per ottenere le coordinate della base ribelle segreta. Ella si fa infine sfuggire l'informazione cruciale: il numero del settore in cui si trova la base è dato dalla somma tra numeratore e denominatore della frazione $q(-4/3)/q(-2)$ (ridotta ai minimi termini). Per fortuna il polinomio $q(x)$ è complicato da costruire, e gli ufficiali dell'Impero stanno ancora cercando di calcolarlo. Per ottenerlo bisogna partire dal polinomio $p(x) = x^3 - 6x^2 + 4x + 12$, chiamare $a$, $b$ e $c$ le sue radici reali, e considerare come polinomio $q(x)$ quello di terzo grado avente come radici $ab + a + b$, $bc + b + c$, $ca + c + a$ e tale che $q(2015) = 2016^{2017}$. Qual è il numero cercato dall'Impero?

```
sage: x = PolynomialRing(RR, 'x').gen()
sage: p = x^3-6*x^2+4*x+12
sage: (a,b,c) = [z for (z, mult) in p.roots()]
sage: cc = a*b+a+b; bb = a*c+a+c; aa = b*c+b+c
sage: q = (x-aa)*(x-bb)*(x-cc)
sage: q(x=-2)/q(x=-4/3)
```

## ⋆Il robot depresso

"Fai questo, fai quello... Su questa nave mi danno solo compiti ridicoli come dimostrare che 2243 è un numero primo" pensava tra sé e sé Artin, il robot della Sezione D'Oro. "Un cervello bionico come il mio può calcolare immediatamente anche quali sono i cinque numeri interi positivi $\alpha_1, \alpha_2, \ldots, \alpha_5$ minori di 2243 tali che $\alpha_i^5 + 2016\alpha_i + 2016$ è multiplo di 2243 per ogni $i$. E so anche dire quanto fa il resto di $\alpha_1^4 + \alpha_2^4 + \cdots + \alpha_5^4$ nella divisione per 2243. Ma nessuno me lo chiede mai!" Sapreste dire anche voi quanto vale questo resto?

```
sage: x = PolynomialRing(GF(2243),'x').gen()
sage: (x^5+2016*x+2016).roots()
[(2102, 1), (2049, 1), (1536, 1), (1008, 1), (34, 1)]
sage: sum(mult*a^4 for (a,mult) in (x^5+2016*x+2016).roots())
908
```

## ⋆La natural Borela

"V'è un sol modo di uscire dall'inferno—disse la mia guida Cartesio—attraverso questa rete di cunicoli, che sbuca dall'altro lato della Terra", e mi mostrò una mappa su un foglio di pergamena. Su di essa era tracciato un triangolo $ABC$. Sul lato $BC$ stavansi nell'ordine 21 punti $A_1, A_2, \ldots, A_{21}$ tali che $BA_1 = A_1A_2 = \cdots = A_{20}A_{21} = A_{21}C$. Analogamente, sul lato $CA$ v'erano in quest'ordine 21 punti $B_1, \ldots, B_{21}$, di nuovo con $CB_1 = B_1B_2 = \cdots = B_{20}B_{21} = B_{21}A$, e poscia sul lato $AB$ in quest'ordine 10 punti $C_1, \ldots, C_{10}$, con $AC_1 = C_1C_2 = \cdots = C_9C_{10} = C_{10}B$. Eran poi tracciate tutte le rette del tipo $AA_i$, $BB_j$ e $CC_k$ per $i = 1, 2, \ldots, 21$, $j = 1, 2, \ldots, 21$ e $k = 1, 2, \ldots, 10$. Finalmente, con l'aiuto di Cartesio, riuscimmo a scoprire in quante regioni era diviso il triangolo, risolvendo un problema con la stella.

```
na = 11; nb = 22; nc = 22;
# Ceva!
intersections = [(a,b,c) for a in range(1,na)
    for b in range(1, nb)
    for c in range(1, nc)
    if a/(na-a)*b/(nb-b)*c/(nc-c) == 1]
print(intersections)
```

# . . . e la geometria?

La geometria va "algebrizzata".

- Calcolo vettoriale (algebra lineare).
- Coordinate cartesiane: c'è qualcosa di implementato, ma non in Sage.
- Altri sistemi di coordinate, ad es. coordinate baricentriche: comode per i triangoli; c'è proprio una libreria Python scritta da un nostro ex-olimpiadista.

## ⋆Degno di un ninja

Con un movimento netto delle dita, Mario deve affettare in due parti uno strano frutto a forma di icosaedro regolare. È necessario che il taglio sia fatto lungo un piano che passa per almeno tre vertici dell'icosaedro. Quanti sono questi piani?

```
def four_points_coplanar(a,b,c,d):
  "Return true if four 3D points are coplanar"
  return matrix([a,b,c,d]).augment(vector([1,1,1,1]))
    .det() == 0

def are_planes_same(a,b):
  "Return true if two lists of 3 3D points represent the same
  return all([four_points_coplanar(a[0],a[1],a[2],p)
    for p in b])
ico = polytopes.icosahedron()
planes = Combinations(ico.vertices_list(),k=3)
unique_planes = []
for p in planes:
  if not any([are_planes_same(p,q) for q
      in unique_planes]):
    unique_planes.append(p)
print len(unique_planes)
```

## Occhio al cappello $\star$

Nel selvaggio West, gli indiani hanno infilzato di frecce il cappello da cowboy di $\widehat{DOC}$. Le frecce hanno tutte la punta nello stesso punto al centro del cappello, e hanno la forma di segmenti tutti della stessa lunghezza. Matryx nota che esistono tre frecce $a, b, c$ tali che l'angolo tra $a$ e $b$ è di $60°$, quello tra $b$ e $c$ è di $60°$, quello $c$ e $a$ è di $90°$. Invece $\widehat{DOC}$, matematico più abile, nota che per ogni coppia di frecce $s$ e $t$ c'è anche una freccia che è la simmetrica di $t$ rispetto al piano perpendicolare a $s$. Quante frecce ci sono nel cappello, al minimo?

```
a = vector((1,0,0)) ; a.set_immutable()
b = vector((0,1,0)) ; b.set_immutable()
c = vector((1/2,1/2,1/sqrt(2))) ; c.set_immutable()
S = set((a,b,c))

def pairwise_symmetries(S):
        SS = set()
        for s in S:
                for t in S:
                        v = s - 2 * (s*t)*t
                        v.set_immutable()
                        SS.add(v)
        return SS

for k in range(10):
        S = S.union(pairwise_symmetries(S))
        print len(S)
```

## La teoria dei futuri astronauti ⋆⋆

Matryx e $\widehat{DOC}$ sono finiti nell'antico Egitto, dove per nascondere la DeuLerean costruiscono un'enorme piramide retta con base un quadrato $ABCD$ e vertice $V$. Sugli spigoli $VB$ e $VD$ prendono rispettivamente due punti $P$ e $Q$ con $\frac{BP}{PV} = \frac{DQ}{QV}$ tali che il piano $APQ$ divide la piramide in due stanze di egual volume. Quanto vale $\frac{BP}{PV}$? *Si risponda indicando* $a + b + 2c$, *dove* $a, b, c$ *sono interi tali che* $\frac{BP}{PV} = \frac{a+\sqrt{b}}{c}$, *e* $b$ *non ha quadrati perfetti tra i suoi divisori.*

```
from sympy import symbols, simplify, solve
from sympy.geometry import Point, Plane, Line

A = Point(1,1,0); B = Point(1,-1,0); C = Point(-1,-1,0);
D = Point(-1,1,0); V = Point(0,0,1)
x = symbols('x', positive=True)
P = B*1/(x+1) + V*x/(x+1)
Q = D*1/(x+1) + V*x/(x+1)
R = Plane(A, P, Q).intersection(Line(V, C))[0]

def volume_of_tetrahedron(a,b,c,d):
        return a.distance(b) * Line(a,b).distance(c) *
            Plane(a,b,c).distance(d) / 6

vol1 = 2*volume_of_tetrahedron(V,A,R,P)
vol1 = simplify(vol1)
vol2 = volume_of_tetrahedron(A,B,C,V) #half pyramid
print(solve(vol1 - vol2, x))
```

## Cerchi nel grano

Nell'anno 3018, è ormai conoscenza comune il fatto che i cerchi nel grano sono messaggi degli alieni del pianeta Otto Persei, grossi esperti della cultura terrestre. In un messaggio particolarmente elaborato, l'alieno Mrrr tracciò un triangolo $ABC$ con $AB = 2017$ e $BC = 2076$. Il cerchio inscritto nel triangolo incontrava $AC$ e $AB$ in $B_1$ e $C_1$ rispettivamente. Il cerchio tangente al lato $AB$ e ai prolungamenti dei lati $CA$ e $CB$ (dai lati di $A$ e $B$ rispettivamente) incontrava la retta $AC$ in $B_2$. Il cerchio tangente al lato $AC$ e ai prolungamenti dei lati $BA$ e $BC$ (dai lati di $A$ e $C$ rispettivamente) incontrava la retta $AB$ in $C_2$. Sapendo che i quattro punti $B_1 C_1 B_2 C_2$ stavano su una stessa circonferenza, trovare il minimo valore possibile per $AC$.

```
# needs Python 3 and https://github.com/Delfad0r/python-bary/
from bary import *
from bary.default import *

B1 = intersect(incircle, AC)[0]
C1 = intersect(incircle, AB)[0]
B2 = intersect(excircle_C, AC)[0]
C2 = intersect(excircle_B, AB)[0]
Gamma = circle_through_three_points(B1, C1, B2)
condition = belongs_to(Gamma, C2)
condition_numeric = condition.subs({a:2076, c:2017})
print(sympy.solve(condition_numeric, b))
```

# Limitazioni

- Velocità ("two-language problem").
- Calcoli simbolici e risolutori funzionano solo quando vogliono.
- Contiene solo le funzioni e i concetti che qualcuno ci ha già scritto dentro.
- A un certo livello di complicazione e dimensione dei conti, anche 'distruggerlo di conti con il computer' smette di funzionare.

# Conclusioni

- A cosa serve la matematica? A risparmiare conti! Un teorema è più bello/utile/interessante quanti più conti fa risparmiare.
- Computer $\gg$ calcolatrice: più vicino alla notazione matematica, più potente, più utile in generale da usare e da insegnare.
- Sarebbe interessante (ma uno sport totalmente diverso!) una gara a squadre umano + computer.