di Andrea de Prisco

Quarta puntata

Teoria della computabilità:

2 lettori... Evviva!

Bene, sono proprio contento. L'Appuntamento di questo mese sarà interamente dedicato alle lettere di due lettori di questa rubrica i quali, molto intelligentemente, hanno provveduto a inviare i loro contributi. Personalmente non pensavo che un simile evento sarebbe mai successo, anche perché non ho mai osato illudermi di avere lettori...

Le due lettere

A scrivere sono due lettori medio giovani (18 e 21 anni) il primo dalla provincia di Gorizia, Antonio Cunei, il secondo da Roma, Roberto Ugolini.

Cominciamo dal più giovane che, oltre ad aver scoperto il bug citato nel riquadro «Non funziona!» pubblicato a p. 152 di MCmicrocomputer n. 58, sostiene che anche la versione corretta suggerita di seguito continua a non essere valida.

Prima di risolvere l'enigma passiamo dunque la parola al lettore.

Salve a tutti!

Innanzitutto mi presento: sono uno sfigatissimo studente diciottenne, accanito lettore di MC, la cui unica ragione di vita sono computer ed affini. Salto i preamboli e passo al punto: Dio, che goduria trovare su un giornale già ottimamente fatto una rubrica come «appunti di informatica»! Ed è proprio riguardo a questa rubrica che scrivo: leggendo avidamente il # 58 di MC ho assorbito con interesse la trattazione riguardante cardinalità finite e transfinite, con connessa dimostrazione di # Fs < # F.

Visto il tutto, però, ho iniziato a fare qualche considerazione col preciso scopo di gettare un po' di caos nel bell'ordine della dimostrazione. E ho trovato un punto debole. Non nella dimostrazione, a dir la verità, ma «intorno». Cercherò di spiegarmi meglio, ma dovrete avere pazienza, dato che ho partorito le suddette considerazioni all'una passata di ieri notte, e che adesso scrivo dopo un'abbondante mangiata (cosa che non aiuta certo a concentrarsi).

Dunque: prima di tutto una contraddizione nell'articolo: nel riquadro «Cardinalità finite e transfinite» di pag. 153 si afferma: «Nel corso dell'articolo abbiamo ad esempio dimostrato che l'insieme delle sequenze finite di numeri ha la stessa cardinalità dei naturali»; e poche righe più sopra: «Tra insiemi infiniti, per decidere se un insieme è o non è della stessa cardinalità dei naturali occorre dimostrarlo: se lo è basta fornire la regola di corrispondenza BIunivoca tra i due [...]». Dolente dirlo, ma vi siete pestati i piedi: il criterio di trasformazione da sequenza finita di numeri (d'ora in avanti li chiamerò «pacchetti», o mi verranno dei crampi alle mani) a numero interno non è affatto biunivoco! Come già esposto nel riquadro «Non funziona!» di p. 152, il criterio esposto nell'articolo non è iniettivo, ma anche corretto come suggerito continua a non essere biunivoco! (o per lo meno, lo è, ma non in N ma in un sottoinsieme di N). Tale criterio manca infatti della proprietà surgettiva; emmò arriva l'esempio: secondo il criterio del riquadro, la sequenza A1, A2... An, viene trasformata nel numero:

$$B1^n * B2^{A1} * B3^{A2} * .. * (B_{n+1})^{AN}$$

Dunque: prendiamo la sequenza 3,2: questa diventa

$$3,2 => 2^2 * 3^3 * 5^2 = 900$$

Se noi abbiamo però come funzione corrispondente all'algoritmo una innocua divisione per due, il risultato sarà:

$$450 = 2^1 * 3^3 * 5^2 => ???$$

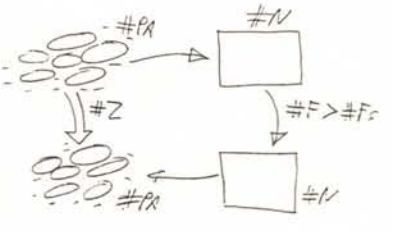
(Yuk, yuk). Orbene, non avendo dimostrato una corrispondenza biunivoca fra i «pacchetti» (sequenze finite di numeri) e i naturali, non avete dimostrato che la cardinalità dell'insieme dei «pacchetti» e quella dei naturali è uguale.

E adesso mi spingo più in là nella cattivissima opera di demolizione: ho notato che, nel corso dell'articolo, dove in una corrispondenza si dimostrano esserci più elementi da un lato del confronto avete usato il <= fra le rispettive cardinalità (fra funzioni binarie e funzioni si ha che # Fb <= # F, come anche fra funzioni calcolate dal formalismo S e programmi dello stesso si ha che # Fs <= # Ps). Ora, io non avevo mai visto niente sulle cardinalità prima di aprire MC # 58, ma a intuito la cosa combina bene, e la userò nello stesso modo (d'ora in avanti il discorso si farà quasi certamente caotico).

Dunque, abbiamo appena dimostrato che la cardinalità dell'insieme dei «pacchetti» non è necessariamente uguale a quella dei naturali, ma che anzi si ha (rifacendosi al discorso di prima) che

$$\#PA <= \#N$$

dove # PA è la cardinalità dell'insieme dei pacchetti finiti di numeri. Ora, nell'articolo si dimostra che # Fs < # F, cioè che «la cardinalità delle funzioni calcolate dal formalismo S è minore strettamente della cardinalità di tutte le funzioni dai naturali ai naturali»; ma, per quanto appena detto, il problema si sposta dal calcolo delle funzioni da N in N, alle funzioni da PA in PA!!! (problema che è chiaramente diverso, perdendosi l'isomorfismo fra le due cose). Per completare la faccenda, chiamiamo Z l'insieme di tutte le funzioni (pessima questa) da PA in PA. Il tutto si può schematizzare nel seguente modo:



Uhmmm, dunque... dov'ero rimasto? Ah, si! Bene; ora dovremmo trovare un legame fra # Z e # PA; a naso direi che # Z > # PA, ma purtroppo sarebbe una dimostrazione poco rigorosa. Mi basta una relazione più rilassata per distruggere completamente quello che resta; a dire la verità la considerazione che seguirà mi sembra un po' ingenua, ma spero che vada bene ugualmente: se noi abbiamo un insieme infinito di elementi K e vogliamo confrontare # K con la cardinalità dell'insieme delle funzioni da K in K, sarà sufficiente che consideriamo, per ogni elemento x di K, la funzione che associa a tutti gli elementi l'elemento x (è una funzione, no? Fa un po' pena, però è una funzione). Chiaramente esisteranno anche altre funzioni oltre a quelle considerate (non è che sia chissà che soddisfatto di questa considerazione, mi manca di rigore). Se ora il nostro insieme K è proprio PA, avremo come conseguenza che:

$$\#Z >= \#PA$$

Incollando insieme il tutto avremo che:

$$\#Z >= \#PA <= \#N < \#F$$

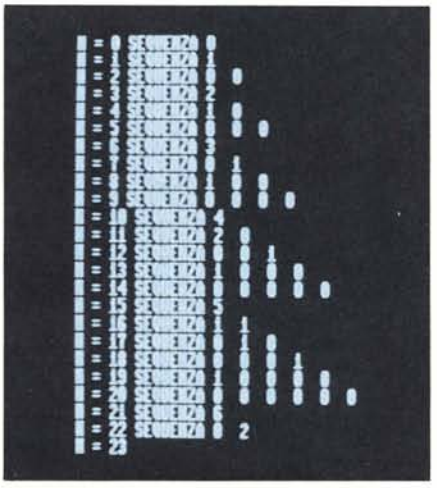
In ogni caso, anche senza l'ultima traballante dimostrazione, si ha comunque che:

$$? \quad \#Z <=> \#PA < \#F$$

per cui la dimostrazione che # Fs < # F non dimostra necessariamente che sia anche # Z < # F (e il fatto che poi questa disequazione sia o non sia verificata va verificato in altra maniera).

Chiaramente c'è qualcosa che puzza in tutta questa sporca faccenda! Ben lungi dall'essere sicuro di quanto affermato, mi piacerebbe comunque verificarlo da qualche parte; dove posso trovare un po' di letteratura che parli dell'argomento?

E adesso, dulcis in fundo, (ma ci sarà ancora qualcuno che legge? o questa lettera starà giacendo in qualche comodo cestino della redazione?) un piccolo, tremendo, terrificante quesito post-natalizio per farvi impazzire tutti quanti: trovare la legge, la regola o comunque il modo in cui questi «numeroni» («AAAAGGGHHH!!!!») vengono successivamente generati (non c'entra con gli automi a stati finiti ed è tremendamente semplice):



1
11
21
1211
111221
312211
13112221
1113213211
31131211131221
...

Per mandarvi ancora di più in casino vi dirò che la cosa va avanti all'infinito, che dopo il primo il numero di cifre è sempre pari e che non compare mai il 4 (ma questa indicazione vi porta inevitabilmente fuori strada). Spero che non ve l'abbiano già fatto!

In conclusione tanti auguri di buona pasqua (non ho speranza che con le poste la lettera arrivi prima) e i miei complimenti per la rivista, che giudico la migliore e più professionale in assoluto in Italia: ogni volta che arriva in edicola, in stazione si vedono girare studenti ipnotizzati con l'ultimo MC aperto davanti (in stazione perché vado a scuola a Udine, pur risiedendo a Monfalcone; d'altra parte a scuola siamo quasi tutti di fuori).

Antonio Cunei
34074 Monfalcone (GO)

```
100 DIM S(100)
110 PRINT
120 PRINT "----------------"
130 PRINT "C - CODIFICA  "
140 PRINT "D - DECODIFICA"
150 PRINT "E - ENUMERAZIONE"
160 PRINT "----------------"
170 PRINT
180 GET A$
190 IF A$="C" THEN GOSUB 280:GOTO 110
200 IF A$="D" THEN GOSUB 370:GOTO 110
210 IF A$="E" THEN GOSUB 510:GOTO 110
220 GOTO 180
230 REM *********************************
240 REM * CODIFICA  E DECODIFICA DELLE *
250 REM * SEQUENZE FINITE  DI NATURALI *
260 REM * ENUMERAZIONE  DELLE SEQUENZE *
270 REM *********************************
280 I=1:P=1
290 I$="":PRINT "ELEMENTO ";I;:INPUT I$
300 IF I$="" THEN 320
310 S(I)=VAL(I$):I=I+1:GOTO290
320 S(I)=I-2
330 II=S(P):JJ=S(P+1)
340 GOSUB590:S(P+1)=FF
350 P=P+1:IFP<>ITHEN330
360 PRINT "CODIFICA = ";S(P):RETURN
370 INPUT"NUMERO = ";NN
380 GOSUB640:I=BB+1:IF BB=0 THEN 420
390 FOR P=I TO 2 STEP -1
400 NN=AA:GOSUB 640
410 S(P)=BB:NEXT
420 S(1)=AA
430 PRINT"SEQUENZA":
440 FOR P=1 TO I
450 PRINTS(P);:NEXT
460 PRINT
470 RETURN
480 REM *************************
490 REM * ENUMERAZIONE SEQUENZE *
500 REM *************************
510 GS=0
520 PRINT"N =";GS;
530 NN=GS:GOSUB380
540 GETA$:IF A$<>"" THEN RETURN
550 GS=GS+1:GOTO520
560 REM *************************
570 REM * FUNZIONE DI CODIFICA *
580 REM *************************
590 FF=(II+JJ)*(II+JJ+1)/2+JJ
600 RETURN
610 REM *************************
620 REM * FUNZIONI DI DECODIFICA *
630 REM *************************
640 PP=0:NP=0
650 KK=NP*(NP+1)/2
660 IF KK<=NN THEN PP=NP:NP=NP+1:GOTO650
670 BB=NN-PP*(PP+1)/2
680 AA=PP-BB
690 RETURN
```

Per quanto riguarda il quesito finale, non voglio buttare li la soluzione soprattutto per non togliere la soddisfazione agli altri lettori (che abbiamo visto sono almeno 2) di risolverlo. Per *dimostrare* però che non voglio raggirare l'ostacolo così vigliaccamente aggiungerò il successivo numerone generato dalla misteriosa regola:

1321131112311311221 1

Più interessanti mi sembrano le affermazioni circa il numero pari di cifre e il fatto che il 4 non venga mai generato: se qualcuno vuole provare a dimostrarlo o a dimostrare il contrario, lo faccia pure. Sarà sicuramente molto interessante.

Veniamo al «tosto»

Desidererei raccontare innanzitutto la storia di Appunti di Informatica. Era il lontano agosto 1985 quando passeggiando (da intruso) nel parco di un noto residence di una ridente località balneare calabra (come il sottoscritto, ridente, balneare e **calabro**) ho avuto la sfortunata (per entrambi) occasione di incontrare il nostro illustrissimo venerato megadirettore galattico On. Marinacci Marco in veste di turista sbomballato in costume da bagno, magliettina, occhiali da sole.

In quella sede, divorati da api, *calabr*..oni e formiche, sotto le fresche fronde di un lì presente albero (calabrese anch'esso) «ci siamo inventati» questa nuova rubrica che avrebbe visto la luce per la prima volta sul numero di novembre di quell'anno.

I patti erano chiari: la rubrica doveva essere «facile», diretta soprattutto ai non addetti ai lavori, ma essenzialmente non doveva avere aspetto tondeggiante (trad.: «pallosa»). Certo, certo... almeno speriamo...

Dal canto mio, gli articoli pubblicati in quelle pagine, avendo pattuito tra l'altro che non doveva trattarsi di una serie a puntate ma chiunque poteva leggere qualsiasi articolo e in qualsiasi ordine, dicevo dal canto mio sapevo perfettamente che l'intera opera non avrebbe mai potuto avere una veste troppo professionale ma doveva rappresentare solo un'infarinatura del tutto generale circa l'informatica che non salta fuori giocando con 64 e il suo joystick. Chi voleva saperne di più circa le argomentazioni trattate poteva comodamente iscriversi al corso di laurea in Scienze dell'Informazione presso una delle 6 o 7 sedi universitarie che lo offrono.

Da qui veri e propri miracoli come 5 pagine di articolo dedicate al progetto a grandi linee di una CPU (MC n. 55) oppure le 4 pagine del n. 52 dove si parla di multitasking a basso livello o gli altri argomenti che trattati seriamente richiederebbero spazi redazionali centinaia di volte maggiore o, meglio, un anno di lezioni presso il suddetto corso di laurea.

Arrivati al tema Computabilità fatta la prima scaletta è risultato evidente che un solo articolo era improponibile e quindi è stata momentaneamente infranta la non consequenzialità degli articoli. Anche occupando però diversi numeri, per evitare di andare avanti tre o quattrocento puntate, e perlomeno esaurendo un sotto-argomento per numero, i colpi di accetta semplificatrice sono stati sempre più pesanti sino al punto, come ha notato il lettore di Gorizia, di fornire addirittura dimostrazioni inconsistenti, nel contesto in cui venivano proposte.

Nell'articolo incriminato il sottoscritto per il solito motivo della farina, non voleva nemmeno inserire il riquadro «Non funziona!» tanto... nessuno lo leggerà mai. Inseritolo lo sbaglio non è stato duplice, come afferma il lettore, ma diciamo 1.3 volte maggiore. Chiarito l'inghippo, argomento di questo numero, ci accorgeremo che tutto calza alla perfezione. Esiste infatti un teorema algebrico che afferma che non esistono infiniti di ordine inferiore all'infinitezza dei naturali. Per essere più precisi, questo teorema dice che qualunque sottoinsieme infinito dei naturali prendiamo esso può essere messo in corrispondenza biunivoca con i naturali stessi. Ad esempio, l'insieme dei pari, è un sottoinsieme proprio dei naturali eppure può essere messo in corrispondenza biunivoca coi naturali. Si dimostra che tale corrispondenza è possibile con qualsiasi sottoinsieme infinito.

Nel nostro caso

Il procedimento di trasformazione da «pacchetti» a naturali mostrato nell'articolo (non nel riquadro) è *totalmente* sbagliato essendo non iniettivo: due sequenze diverse possono generare lo stesso naturale quindi facendo il passo inverso non riusciremo a ricostruire la sequenza di partenza. Ciò è stato ufficialmente dichiarato nel riquadro «Non funziona!», quindi siamo a posto.

Di contro, il lettore fa notare che per quanto iniettivo, il procedimento di seguito mostrato come corretto non essendo surgettivo non è nemmeno biunivoco quindi il sottoscritto non avrebbe dimostrato l'equipotenza coi naturali e tutta la dimostrazione del fatto che $\#Fs < \#F$ sarebbe andata a farsi benedire. Idem dicasi per il terzo procedimento, anche se non c'è stata segnalazione dal lettore.

Ovvero esistono dei naturali (ad esempio 450) che non generano alcuna sequenza. Attenendosi perfettamente a quanto indicato in tutto l'articolo, sembrerebbe che la frase «Per attuare una codifica a tutti gli effetti biunivoca...» sia fuori posto e che effettivamente i due procedimenti mostrati non siano biunivoci. Capisco perfettamente che c'è da mettersi le mani nei capelli, ma non è colpa mia se nella teoria delle cardinalità transfinite (Cantor) ho ragione.

Comprendiamo meglio

In altre parole, i due procedimenti forniti nel riquadro non mettono in corrispondenza biunivoca i «pacchetti» con i naturali ma con un suo sottoinsieme ricorsivo (nel senso di calcolabile). Tanto per fare un esempio, 900 appartiene a tale insieme, 450 no. Giusto per chiarire la ricorsività di questo, per sapere se un particolare numero appartiene o no a tale sottoinsieme dei naturali basta fare la scomposizione in fattori primi del numero (operazione sempre possibile): se otteniamo una sequenza lecita (primo esponente maggiore o uguale del numero di elementi rimanenti della sequenza così ottenuta) allora il numero da cui siamo partiti appartiene all'insieme ricorsivo, se tale sequenza è illecita, come quella generata da 450 nella lettera del lettore, tale elemento non vi appartiene.

Dal momento che ogni sottoinsieme infinito dei naturali può essere messo in corrispondenza biunivoca con i naturali, otteniamo per la ben nota proprietà transitiva (come per magia, tanto per cambiare) che i pacchetti possono essere messi in corrispondenza biunivoca con i naturali, quindi la dimostrazione del numero 58 regge perfettamente.

Per maggiore correttezza

Dal momento che in questo momento sicuramente almeno il lettore di Gorizia sta leggendo queste righe, e che effettivamente semplificare troppo le cose a lungo andare può creare qualche problema, prima di passare al secondo lettore (risposta brevissima), per questioni di correttezza o di maggiore professionalità (forse spesso poco usata in Appunti) innanzitutto daremo una dimostrazione del fatto che qualsiasi sottoinsieme infinito dei naturali è equipotente (ha la stessa cardinalità) ai naturali e poi, per accontentare Gorizia, forniremo un metodo iniettivo e surgettivo quindi biunivoco senza trucchetti transfiniti (e questa volta vado sul sicuro: è il metodo «ufficiale» anche se un po' più incasinato) per mettere in corrispondenza i naturali con i «pacchetti».

Dimostrazione 1

La dimostrazione rigorosa che un qualsiasi sottoinsieme infinito B di un insieme numerabile A è numerabile è presente su qualsiasi testo di teoria degli insiemi (in appendice abbiamo indicato un testo usato per l'esame di Algebra del corso di laurea in Scienze dell'Informazione) e fa naturalmente riferimento a relazioni di ordinamento, di equivalenza, cardinalità, partizioni, definizioni per induzioni e a tutto un bagaglio di conoscenze in tali campi per cui è meglio rimandare i lettori più interessati agli appositi testi disponibili nelle librerie scientifiche.

In questa sede daremo una dimostrazione meno rigorosa ma più intuitiva e tangibile di come sia possibile realizzare tale corrispondenza biunivoca.

Immaginiamo di avere un qualsiasi sottoinsieme infinito dei naturali. Dal momento che su tutti i numeri naturali è definita la relazione di ordinamento $<=$ (minore-uguale), ovvero presi due numeri qualsiasi è sempre possibile stabilire quale dei due è minore-uguale dell'altro, altrettanto matematicamente è possibile, preso qualsiasi sottoinsieme dei naturali (finito o infinito) stabilire qual è il minimo, ossia quale elemento è minore-uguale di tutti gli altri.

Dal nostro insieme qualsiasi da cui siamo partiti estraiamo il minimo (togliendolo) e a questo associamo il primo naturale, lo zero. Riapplichiamo il procedimento sui rimanenti elementi estraendo l'elemento da associare al secondo naturale e così via. Si noti che (uno) il procedimento non termina mai essendo il sottoinsieme dal quale siamo partiti infinito; (due) non finendo mai l'applicazione è surgettiva ovvero tutti i naturali saranno prima o poi associati con elementi dell'insieme B; (tre) per il tipo di costruzione attuata non possiamo applicare due elementi diversi allo stesso naturale (ricordo che negli insiemi ogni elemento è presente al più una volta) quindi l'applicazione è banalmente iniettiva; (quattro) come lo stesso lettore fa notare, se un'applicazione da un insieme ad un altro è iniettiva e surgettiva allora tale applicazione è biunivoca ed esiste anche l'inversa: sono in grado di passare da un elemento dell'insieme B ad un naturale così come di fare il contrario.

Dimostrazione 2

La seconda dimostrazione riguarda la codifica BIunivoca delle sequenze finite di naturali, in naturali. Essa si rifà alla codifica delle coppie di naturali in un naturale. Ovvero a un procedimento biunivoco per passare da una qualsiasi coppia di numeri a un naturale che lo identifica univocamente. L'estensione al caso in cui non abbiamo a che fare con coppie ma con sequenze comunque grandi (finite) di naturali è assai semplice: dapprima aggiungo alla sequenza lunga n il numero di elementi (decrementato di uno) come ultimo elemento ottenendo una sequenza lunga n+1 dopodiché codifico i primi due elementi con un naturale, poi prelevo il terzo elemento della nuova sequenza e lo codifico col naturale testé ottenuto... continuando fino a quando non termino la sequenza ottenendo così un unico numero che la identifica univocamente. Per tornare indietro (dal numero alla sequenza) è sufficiente applicare il procedimento inverso tante volte quante indicate al primo numero ottenuto con la decodifica più uno.

Attenzione: se il primo valore decodificato è zero vuol dire che la sequenza è lunga 1 (non stupitevi, una sequenza può anche essere fatta di un solo elemento così come di zero elementi, sequenza vuota, anche se quest'ultima a noi non interessa) e il valore cercato è il secondo componente della coppia generata da quel numero.

Prima di mostrare qualche esempio di codifica e decodifica di «pacchetti», dimostriamo il primo passo: la biunivocità della codifica delle coppie in un naturale. Forniremo sia la funzione che dalla coppia passa al naturale corrispondente sia le due funzioni (inverse) che preso un naturale restituiscono il primo e il secondo elemento della coppia corrispondente.

Per attuare una codifica delle coppie immaginiamo di costruire la tabella di figura 2. La coppia (i,j) si trova nella i-esima riga alla j-esima colonna. Seguendo le freccie diamo così una numerazione alle coppie: la coppia (0,0) sarà la zero, la coppia (1,0) la uno, la coppia (0,1) la due, la coppia (2,0) la terza è così via. La formula che permette di ricavare direttamente la posizione in funzione di i e j della coppia (considerato che i+j di ogni freccia è costante) è molto semplice ed è mostrata in figura 3a.

Per il tipo di costruzione attuato, essa è iniettiva (seguendo il percorso delle freccie associamo ad ogni coppia un naturale diverso, sempre crescente) ed è anche banalmente surgettiva in quanto nella costruzione non saltiamo alcun naturale ed essendo le coppie infinite, li accoppieremo «tutti».

Le due funzioni inverse che, preso un naturale n restituiscono il primo e il secondo elemento della coppia che lo identifica biunivocamente, sono chiamate in figura 3b e 3c «A» e «B». Esse fanno riferimento ad un numero K (funzione di n, mostrato in figura 3d) che è uguale al più grande intero P tale che P moltiplicato P+1 diviso 2 è minore-uguale di n.

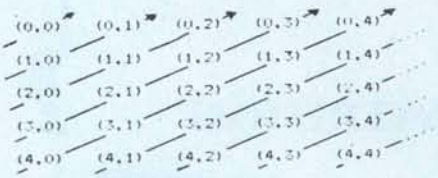


Figura 2: codifica delle coppie di naturali.

(a) F(i,j) = (i+j)(i+j+1)/2 + j

(b) A(n) = K - B(n)

(c) B(n) = n - K(K+1)/2

(d) K = max p [p(p+1)/2 <= n]

Figura 3: funzioni di codifica e decodifica delle coppie.

Facciamo un esempio

Proviamo a codificare la sequenza di naturali:

{0,1,0,1}

Abbiamo scelto numeri piccoli per non ottenere risultati troppo grandi (anche qualche migliaio di cifre!) e far andare MC in overflow. Come detto, la prima operazione da compiere è di arricchire la sequenza di partenza col numero di elementi meno uno di cui questa è formata. Quindi codificheremo:

{0,1,0,1,3}

Applichiamo la formula di figura 3a ai primi due elementi ottenendo il naturale 2 (calcolare per credere). Il 2 così ottenuto lo associamo col terzo elemento, lo zero, ottenendo sempre dalla stessa formula il naturale 3. Combinato questo col quarto elemento, l'uno, la funzione restituirà 11 che, abbiamo quasi finito, combinato col 3 (ultimo elemento) dà come risultato 108. Questo valore identifica univocamente la sequenza «arricchita» di cui sopra.

Proviamo a tonare indietro, partendo dunque da 108. Applichiamo la funzione B a tale valore ottenendo 3, ultimo elemento della sequenza non-

ché, aumentato di 1, ci indica quante volte ancora dobbiamo applicare de-odifiche. Bene, applicata la funzione A al valore 108 otteniamo 11. Adoperando B su 11 otteniamo 1 (quarto elemento della sequenza) applicando la A questa ci restituisce 3. Ancora, applichiamo la B ed otteniamo zero (terzo elemento della sequenza) applicando A, sempre su 3, otteniamo 2. Per finire basterà applicare le due funzioni a tale numero per ottenere primo e secondo elemento della sequenza, 0 e 1.

Semplice? no?

Il listato Basic mostrato serve per fare qualche esperimento di questo tipo. Voi fornite un input ed esso vi restituirà la sequenza corrispondente. Oppure potete effettuare una codifica inserendo i vari elementi rispondendo return quando avete finito. Non adoperate numeri troppo grandi o sequenze molto lunghe per non far andare la macchina, qualunque essa sia (purtroppo... vedi MC n. 59, stessa rubrica) in overflow. Buon divertimento.

Conclusioni

Per concludere vorrei complimentarmi col lettore di Gorizia che studiando attentamente, e nei minimi particolari, il primo articolo sulla computabilità è riuscito a tirarne fuori abbastanza per provocare l'uscita di quest'articolo, praticamente tutto dedicato a lui. Per quanto riguarda il resto della sua lettera, una volta chiarito che i «pacchetti» possono effettivamente essere messi in corrispondenza biunivoca con i naturali, la sua quasi-dimostrazione, terminante con quel giusto punto interrogativo sull'ultima disequazione, trova risposta nell'uguale e che quindi la cardinalità delle funzioni da «pacchetti» a «pacchetti» è uguale alla cardinalità delle funzioni da naturali a naturali nonché maggiore della cardinalità dei «pacchetti» che abbiamo detto essere pari a quella dei naturali. In definitiva la sua ultima disequazione la scriveremo così:

$$\#N = \#PA < \#Z = \#F$$

Da notare, infine, che per ogni insieme finito o infinito che sia si dimostra che la cardinalità dell'insieme è sempre strettamente minore delle funzioni dall'insieme in se stesso ed è possibile anche calcolare quanto vale: due elevato alla cardinalità dell'insieme.

Se ad esempio il nostro insieme è finito ed è formato da quattro elementi (cardinalità 4) l'insieme delle funzioni da tale insieme in se stesso è 2 alla 4 dunque 16. Analogamente per gli insiemi infiniti: la cardinalità dei naturali è una quantità transfinita alla quale è stato dato anche un nome «Alef-zero» dalla prima lettera dell'alfabeto ebraico con deponente 0. Le funzioni dai naturali ai naturali hanno cardinalità due alla Alef-zero, altra quantità transfinita al quale è stato dato il nome «C». Essa è strettamente maggiore di Alef-zero e si dimostra essere pari alla cardinalità dei numeri reali nonché alla cardinalità di parti di N.

Altre quantità transfinite sono le funzioni da reali a reali (due alla C) o parti di R e così via (parti di parti di R, parti di parti di parti di R ecc.) tutti strettamente l'uno più grande dell'altro.

Come disse nel 1914 Hausdorff, nella prefazione ad una sua opera sulla teoria degli insiemi: «...un campo in cui nulla è di per se stesso evidente, in cui enunciati veri sono spesso paradossali ed enunciati plausibili sono falsi».

```
Program Fibonacci (input,output);
Var n:integer;
Function Fib (x:integer):integer;
Begin
  If x <= 2 Then Fib:=1
            Else Fib:=Fib(x-1)+Fib(x-2)
End;
Begin
  Read(n);
  Write(Fib(n))
End.
```

Figura 4: programma Fibonacci in versione ricorsiva inviata dal lettore Roberto Ugolini di Roma.

```
Program Fibonacci (input,output);
Var n,fib,fib1,fib2,i: integer;
Begin
  Read(n);
  If n <= 2 then Fib:=1
     Else Begin
            Fib1:=1;
            Fib2:=1;
            For i:=1 to n-2 Do
            Begin
              Fib:=Fib1+Fib2;
              Fib2:=Fib1;
              Fib1:=Fib
            End
          End;
  Write(Fib)
End.
```

Figura 5: stesso programma, dello stesso autore, in versione iterativa.

Seconda lettera

Il secondo lettore, con intenzioni ben più pacifiche, ci scrive da Roma riguardo alle funzioni ricorsive, argomento del numero 57. A lui la parola:

Caro Andrea,

mi chiamo Roberto, ho 21 anni e seguo con molto interesse la tua rubrica «Appunti di Informatica». Nel numero 57 mi ha particolarmente interessato la questione del calcolo del K-esimo numero di Fibonacci.

Ho provato a scrivere un programma non ricorsivo e poi ho svolto alcune considerazioni sulla questione dell'efficienza. Innanzitutto ti scrivo le mie due soluzioni al problema (in Pascal su uno Spectrum 48 K, figure 4 e 5).

Penso sia abbastanza chiara la maggiore leggibilità della soluzione ricorsiva, ma quanto all'efficienza?

Con la soluzione ricorsiva il max numero di Fibonacci ottenibile è 28657 con un tempo di calcolo di 11.18 secondi. Con la versione iterativa il medesimo calcolo avviene in pochi attimi. Inoltre con la prima soluzione per ogni chiamata della funzione provoca l'allocazione di spazio in memoria e il tempo di esecuzione cresce esponenzialmente contro la linearità dell'algoritmo iterativo. Quindi, al momento di progettare l'algoritmo conviene considerare che se il programma viene eseguito su un input di piccole dimensioni e il problema è ricorsivo allora conviene un algoritmo ricorsivo, in tutti gli altri casi conviene usare un algoritmo iterativo.

Roberto Ugolini - Roma

Caro Roberto,

quanto tu dici nella tua lettera è vero, se parliamo in termini di efficienza e il problema da ricorsivo può essere facilmente trasformato in iterativo. Esistono però dei problemi che difficilmente possono essere trasformati in iterativi ovviamente scartando l'ipotesi di implementare un ambiente pseudo-ricorsivo su linguaggi di programmazione che non dispongono di tale possibilità. Nell'articolo del n. 57 abbiamo parlato di Fattoriale e di Fibonacci, due procedimenti ricorsivi assai famosi. Se però ci spostiamo un po' più sul difficile, come una partita a scacchi, la risoluzione del cubo di Rubik, un programma di riempimento figure in pagina grafica, la ricerca, l'inserimento o il bilanciamento di una struttura ad albero, ti accorgeresti facilmente di come comincia ad essere impensabile un'applicazione non ricorsiva. Nel tuo esempio basta notare che nel primo caso è necessaria una sola variabile e un misero IF, nel secondo ben 5 variabili il solito IF e un FOR. Quindi come vedi la complessità è più che doppia, quindi l'aumento di velocità in qualche modo è pur costato. Come dire: «Nulla si crea... nulla si distrugge...».

Bibliografia

Aiello, Albano, Attardi, Montanari:
TEORIA DELLA COMPUTABILITÀ LOGICA, TEORIA DEI LINGUAGGI FORMALI
Editrice ETS Pisa, 1976

Antonelli, Manca, Salibra:
LOGICA
Editrice ETS Pisa, 1980

Cecconi, Stampacchia:
ANALISI MATEMATICA, VOLUME 1, CAP. 1
Liguori Editore, 1974