Informatica Generale

Homework di Prova: Codifiche, ovvero

Tre Passi con Cantor & Gödel

docente: Ivano Salvo – Sapienza Università di Roma

Nelle prime lezioni abbiamo lasciato intendere che frammenti minuscoli del C (il famigerato TinyC o l'altrettanto odiato TinyReC) siano sufficienti per calcolare (magari con una complessità computazionale molto più alta) qualsiasi funzione calcolabile in C. Lo studente attento avrà notato che questi frammenti non hanno strutture dati, ma solo il tipo `int` che più volte ho invitato a pensare non tanto come il misero tipo `int` del C, ma come un tipo di dato che permetta memorizzare numeri naturali *arbitrariamente* grandi.

Questo homework, vi farà riflettere sul fatto che qualunque informazione possa essere *codificata* con un numero naturale (almeno in linea di principio), sia essa la vostra canzone preferita, un film, o il risultato reale(!?) di un complesso calcolo scientifico. Questa scoperta risale forse alla tecnica dell'*aritmetizzazione* di Gödel o forse, più embrionalmente è contenuta nelle scoperte di Cantor sulla cardinalità. Più prosaicamente (o più "ingegneristicamente") potete pensare il desolato deserto di 0 e 1 della memoria del vostro calcolatore come un'unica lunga sequenza che codifica (in binario) un unico numero naturale (compreso tra 0 e $2^{2^{37}} - 1$ nel mio calcolatore con 16GB di memoria RAM).

Il resto del corso, sarà dedicato a far vedere, viceversa, come occorra scegliere opportunamente le struttura dati dei vostri programmi, evitando codifiche troppo criptiche!

Esercizio 1 (Coppie). Possiamo codificare le coppie di naturali come segue. Prima codifichiamo l'unica coppia di somma 0 (0,0) con il numero 0, poi le coppie di somma 1, codificando con 1 la coppia (0,1) e con 2 la coppia (1,0), poi tutte le coppie di somma 2, assegnando 3 a (0,2), 4 a (1,1) e 5 a (2,0) e così via (vedi Figura 1). Il vostro programma dovrà leggere una coppia di interi e produrre in output la corrispondente codifica. Questo esercizio vi faccia riflettere sul fatto che l'insieme dei $\mathbb{N}$ dei naturali è equipotente all'insieme $\mathbb{Q}$ dei razionali [**Sugg:** riflettete sulla codifica delle coppie nella forma $(0, k)$].

Input: Due numeri interi, da leggersi con due `scanf("%d",&m)`.

Output: Un numero intero che rappresenta relativa codifica.

Esercizio 2 (Sequenze Finite) Consideriamo l'insieme delle sequenze lunghe k con possibili valori nell'insieme $\{0, 1, \ldots, n-1\}$. Questo tipo di dato assomiglia ai vettori del C. Ovviamente le possibili sequenze sono n^k (sono le disposizioni con ripetizione di n oggetti presi k ad k). Possono quindi essere codificate con un numero naturale compreso tra 0 ed $n^k - 1$ come segue. Le sequenze che cominciano con 0 sono numerate da 0 a

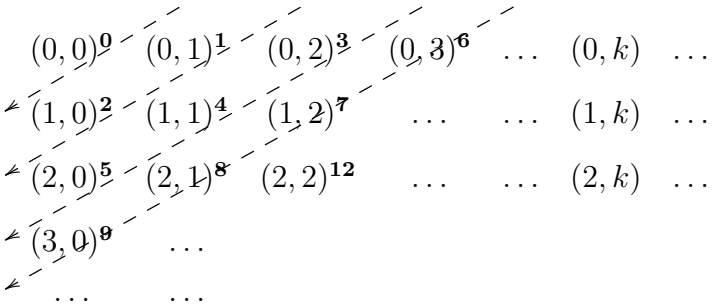

Figura 1: codifica *a coda di rondine* di Cantor dei "razionali"

$n^{k-1}-1$, quelle che cominciano con 1 da n^{k-1} a $2n^{k-1}-1$, fino a quelle che cominciano per $n-1$ che sono numerate da $(n-1)n^{k-1}$ fino a n^k-1. Per trovare il numero esatto occorre applicare lo stesso ragionamento alla coda della sequenza e codificarla con un numero da 0 a $n^{k-1}-1$.

Il vostro programma dovrà leggere gli interi n, k e poi una sequenza di k interi compresi tra 0 e $n-1$ e produrre in output la codifica della sequenza.

ESEMPI: Presi in input qualsiasi n e qualsiasi k ed una sequenza di k zeri, il programma dovrà produrre in output 0.

Presi in input 4 e 5 e poi la sequenza di 5 numeri $2,1,0,3,1$ il programma dovrà restituire $2\cdot 4^4+1\cdot 4^3+0\cdot 4^2+3\cdot 4+1$ e quindi il numero 589 (vi ricorda qualcosa? Se n fosse 10?).

⋆Esercizio 3 (SOTTOINSIEMI DI CARDINALITÀ FISSATA) Immaginiamo di voler codificare le combinazioni (senza ripetizione) di n oggetti presi a k a k, o se preferite i sottoinsiemi di cardinalità k presi dall'insieme $\{1,\ldots,n\}$. Anche se di fatto stiamo numerando insiemi, consideriamo ogni k-upla $\{a_0,\ldots,a_{k-1}\}$ sempre scritta in forma *canonica* come una sequenza crescente di interi $\langle a_0,\ldots,a_{k-1}\rangle$ tali che $a_0<a_1<\ldots<a_{k-1}$ e immaginiamole numerate seguendo l'*ordine lessicografico* (cioè quello del dizionario).

Per esempio, nel gioco del Lotto, ogni estrazione è una cinquina formate con i numeri da 1 a 90. La cinquina $\langle 1,2,3,4,5\rangle$ avrà il numero 0, la cinquina $\langle 1,2,3,4,6\rangle$ il numero 1, la cinquina $\langle 1,3,7,70,75\rangle$ il numero 123.644 fino alla cinquina $\langle 86,87,88,89,90\rangle$ che avrà il numero 43.949.267, che correttamente corrisponde a $\binom{90}{5}-1$

Può essere utile osservare che le sequenze che contengono 1 sono numerate da 0 a $\binom{n-1}{k-1}-1$ (infatti mi rimane la libertà di avere tutte le combinazioni di $n-1$ oggetti sui $k-1$ posti rimanenti), quelle che cominciano per 2 da $\binom{n-1}{k-1}$ a $\binom{n-1}{k-1}+\binom{n-2}{k-1}-1$. Quelle che cominciano per 1,2 invece sono numerate da 0 a $\binom{n-2}{k-2}-1$ e così via.

Dovete scrivere un programma che legge tre interi: n, k e un numero m compreso tra 0 e $\binom{n}{k}-1$ e produce in output k interi (in ordine crescente) corrispondenti alla k-upla che ha codifica m.

ESEMPI: Presi in input la terna 6, 3 e 11 dovrà produrre in output: 2 3 5.

Presi in input la terna 90, 5 e 123644 dovrà produrre in output: 1 3 7 70 75.