Informatica Generale
Homework 1/2019: Primi, Composti, Razionali

Ivano Salvo – Sapienza Università di Roma – 7/4/2018

Esercizio 1. (Divisori Primi Distinti) Usare la logica del crivello di Eratostene (vedi *Dispensa* **D4**, sez. **2.4**) per scrivere una funzione (vedi riquadro) che carica il vettore d di lunghezza k, mettendo in $d[i]$ il numero di divisori primi distinti di i e torna come risultato, il *primo* numero $i < k$ che ha il massimo numero di divisori primi distinti (questa è la sequenza A001221 dell'*On-line Encyclopedia of Integer Sequecens*). Assumere il vettore d inizializzato a tutti 1, tranne in posizione 1, dove c'è uno zero.

```
int divisoriPrimi(int d[], int k)
  /* PREC: d vettore di k>0 elementi, riempito di 1 (ma d[1]=0)
   * POST: modifica d, dimodoche' d[i]=numero divisori di i > 0
   *     ritorna il numero i per cui questo numero e' massimo
   */
```

Esempio Se k fosse 31, il vettore risultante deve essere:

$$\#, 0, 1, 1, 1, 1, 2, 1, 1, 1, 2, 1, 2, 1, 2, 2, 1, 1, 2, 1, 2, 2, 2, 1, 2, 1, 2, 1, 2, 1, 3$$

e la funzione tornerebbe $30 = 2 \cdot 3 \cdot 5$ che è il primo numero con 3 divisori distinti. (# significa che non è rilevante il valore del vettore in posizione 0 ☺) Se k fosse compreso tra 7 e 30, la funzione deve tornare 6 (il primo che ha 2 divisori primi distinti). In altre parole, la funzione torna il più grande primoriale minore di k.

Esercizio 2. (Altamente Composti) Usare la logica del crivello di Eratostene (vedi *Dispensa* **D4**, sez. **2.4**) per scrivere una funzione (vedi riquadro) che *alloca* un vettore di k elementi e lo restituisce in output caricato mettendo

in $d[i]$ il numero di divisori (stavolta non necessariamente primi) di i (questa è la sequenza A000005 dell'*On-line Encyclopedia of Integer Sequecens*). La funzione inoltre carica in *ac* il primo indice del massimo del vettore, cioè il più grande numero altamente composto minore di k.

```
int* maxAltamenteComposto(int k, int* ac)
   /* PREC: k>0, POST: torna un vettore d lungo k
    * tale che d[i]=numero divisori di i>0 e carica in *ac
    * il maggiore numero altamente composto < k.
    */
```

Esempio Se k fosse 31, il vettore risultante sarebbe:

$$\#, 1, 2, 2, 3, 2, 4, 2, 4, 3, 4, 2, 6, 2, 4, 4, 5, 2, 6, 2, 6, 4, 4, 2, 8, 3, 4, 4, 6, 2, 8$$

e la funzione caricherebbe in `*ac` il numero 24 che è il primo numero con 8 divisori (# significa che non è rilevante il valore del vettore in posizione 0 ☺).

Esercizio 3 (Enumerare i Razionali) Probabilmente, tutti sapete che i numeri razionali possono essere messi in corrispondenza biunivoca con i naturali. Usualmente, ciò viene insegnato mostrando che i naturali possono essere messi in corrispondenza biunivoca con le coppie di numeri naturali (la cosiddetta codifica *diagonale* o a *coda di rondine*, vedi ad esempio *Homework* **P, 2016**), e poi facendo implicito appello al fascinoso (e altamente *non costruttivo*) Teorema di Berstein-Cantor-Schröder.

Tuttavia, qui dovrete agire da informatici (e quindi costruttivisti!) e dovrete programmare due funzioni una inversa dell'altra (vedi riquadro) che codificano/decodificano i numeri razionali positivi *direttamente*, mappando nello stesso intero le coppie (m, n) e (m', n') ogni qualvolta ${}^{m}\!/_{n} = {}^{m'}\!/_{n'}$.

L'idea si basa sul seguente fatto: la funzione mcd che calcola il massimo comun divisore nella versione *sottrattiva*:

$$\begin{array}{lrcll} (*) & \mathsf{mcd}(n,n) & = & n & \\ (0) & \mathsf{mcd}(m,n) & = & \mathsf{mcd}(m,n-m) & (m<n) \\ (1) & \mathsf{mcd}(m,n) & = & \mathsf{mcd}(m-n,n) & (n<m) \end{array}$$

segue la stessa sequenza di calcolo per due qualsiasi coppie di numeri (m, n) e (m', n') ogni qualvolta ${}^{m}\!/_{n} = {}^{m'}\!/_{n'}$. Quindi codificheremo le computazioni di mcd come sequenze binarie e le sequenze binarie in numeri naturali.

Banalmente, se $m = n$ si applica la clausola $(*)$ e non viene attivata nessuna chiamata ricorsiva. Tutte le coppie del tipo (m, m), che rappresentano tutte il numero razionale 1, verranno rappresentate dalla sequenza vuota.

Sulle coppie nella forma $(n, 2n)$ si applica la clausola (0) e poi la computazione si arresta sui valori (n, n). Simmetricamente, sulle coppie nella forma $(2n, n)$ si applica la clausola (1) e ci arrestiamo come sopra. Quindi tutte le coppie nella forma $(n, 2n)$, che rappresentano tutte il numero razionale $1/2$, saranno rappresentate dalla sequenza 0, mentre quelle nella forma $(2n, n)$, che rappresentano tutte il razionale $2/1 = 2$, dalla sequenza 1. Facciamo un ultimo esempio più complesso: l'input di mcd è una coppia che, come razionale, rappresenta $5/2$. La computazione di mcd sarà:

$$\mathsf{mcd}(5n, 2n) \xrightarrow{1} \mathsf{mcd}(3n, 2n) \xrightarrow{1} \mathsf{mcd}(n, 2n) \xrightarrow{0} \mathsf{mcd}(n, n)$$

e rappresenteremo $5/2$ con la sequenza 110. Altri esempi in Fig. 1.

A questo punto, occorre codificare le sequenze di 0 e 1. Codifichiamo le 2^n sequenze lunghe n con i numeri naturali compresi tra 2^n e $2^{n+1} - 1$: una sequenza lunga n che in binario rappresenta il numero b, sarà rappresentata dal numero intero $2^n + b$. La sequenza vuota $*$ sarà rappresentata dal numero naturale 1. Quindi $1 = *, 2 = 0, 3 = 1, 4 = 00, 5 = 01, 6 = 10, 7 = 11, 8 = 000, 9 = 001, 10 = 010, 11 = 011, 12 = 100, 13 = 101, 14 = 110, 15 = 111, 16 = 0000$, e così via. Assumete che la lunghezza delle sequenze sia al più 31.

Ovviamente la funzione di decodifica dato un intero, dovrà ritornare il corrispondente razionale rappresentato nella *forma canonica* ridotta ai minimi termini: dato l'intero occorre calcolare la sequenza di 0 e 1 associata ed usarla per eseguire "all'indietro" mcd, partendo dalla coppia (1,1).

```
int codificaRat(int num, int den)
  /* PREC: num>=0,den>0.
   * POST: torna la codifica di num/den. Torna 0 se num=0
   */

void decodificaRat(int n, int* num, int* den)
   /* PREC: n>=0.
    * POST: carica in *num e *den due numeri coprimi,
    * tali che n = codificaRat(*num, *den)
    */
```

Chi si appassionerà al problema, potrà leggere il breve scritto di Calkin e Wilf, *"Recounting the Rationals"*, reperibile in rete all'indirizzo:
`https://www.math.upenn.edu/~wilf/website/recounting.pdf`

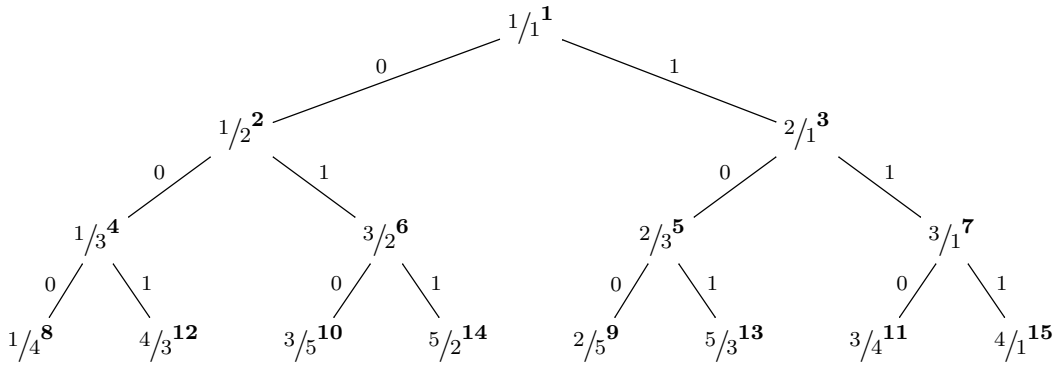

Figura 1: L'albero dei razionali di Calkin-Wilf. In apice e in neretto il naturale corrispondente alla frazione nella numerazione proposta. I cammini (dal basso verso l'alto) che congiungono $^m/_n$ a $^1/_1$ sono le corrispondenti sequenze binarie che codificano le computazioni di $\mathsf{mcd}(m, n)$. Per avere i naturali in neretto ordinati da sinistra a destra in ogni livello, dovremmo codificare le sequenze rovesciate di computazione di $\mathsf{mcd}(m, n)$, cioè lette dall'alto verso il basso.

Note Pratiche. Dovete consegnare semplicemente la funzione richiesta, **rispettando il prototipo**. Il vostro file può contenere eventuali altre funzioni ausiliarie. Non dovete lasciare **nessuna** istruzione di input/output nel codice. Potete verificare se il vostro programma funziona, utilizzando i files `main-1-1.c`, `main-1-2.c`, e `main-1-3.c` forniti nella pagina degli Homework.

Potete chiamare come volete il file `.c` contenenti la funzione richiesta (ed eventuali funzioni ausiliarie): il sistema li rinominerà automaticamente usando il vostro twiki name. Se il vostro twiki name fosse `LilyEvans`, essi verranno chiamati `LilyEvans.1.c`, `LilyEvans.2.c`, e `LilyEvans.3.c`. Il comando utilizzato per la compilazione sarà:

```
gcc -std=c99 main-1-1.c LilyEvans.1.c
gcc -std=c99 main-1-2.c LilyEvans.2.c
gcc -std=c99 main-1-3.c LilyEvans.3.c
```

Osservate che i files dovrebbero poter essere compilati separatamente con l'opzione `-c` che traduce in linguaggio macchina, ma senza generare un eseguibile (`gcc -c -std=c99 LilyEvans.1.c` e `gcc -c -std=c99 main-1-1.c` dovrebbero generare con successo un file chiamato rispettivamente `LilyEvans.1.o` e `main-P-1.o`.