Laboratorio di Algoritmi e Strutture Dati

Esercitazioni del 13 Novembre 2012

Esercizio 1: gcd, lcm e frazioni egizie

Parte 1: gcd e lcm

Un modo efficiente per calcolare il *massimo comun divisore* `gcd(a,b)` di due interi `a` e `b` nonnegativi è dato dall'*Algoritmo di Euclide*.
Assumiamo `b > 0`. Si tratta di determinara la coppia di interi `q,r` univocamente determinati da:

```
a = b * q + r,     r < b
```

allora

```
gcd(a,b) = gcd(b,r)
```

In particolare, `gcd(a,0)`= a.
Esempio:
Si voglia calcolare `gcd(1071,1029):` Allora:

```
1071 = 1029 * 1 + 42
1029 = 42 * 24 + 21
42 = 21 * 2 + 0
```

Avendo ottenuto resto `0`, concludiamo che `gcd(1071,1029) = 21`.

Il *minimo comune multiplo* `lcm(a,b)` si può agevolmente calcolare a partire da `gcd(a,b)`. Infatti:

```
lcm(a,b) = a * b / gcd(a,b)
```

(basta considerare la fattorizzazione in primi di `a` e `b` per convincersene).

Si richiede di scrivere un programma che:

1. Chieda all'utente di immettere due interi `a` e `b` (con `a >= 0, b > 0`, non si richiede il controllo della correttezza dell'input).
2. Calcoli e stampi `gcd(a,b)` e `lcm(a,b)`.

Parte 2: Frazioni Egizie

Sia data una frazione `0 <= a/b <= 1` e sia `h/k` la sua espressione in forma irriducibile (vale a dire: `h/k = a/b` con `h` e `k` primi fra loro: `gcd(h,k) = 1`. Allora

```
h/k = 1/d1 - 1/(d1 * d2) + 1/(d1 * d2 * d3) - 1/(d1 * d2 * d3 * d4) ...
```

per un numero finito di interi `d1, d2, ...`, ognuno dei quali `>= 1`, calcolati attraverso le relazioni seguenti:

```
d = floor(k/h)
```

```
h/k = 1/d - 1/d * (h'/k')
```

dove `floor(x)` è la parte intera di `x`.
Dunque, `h'` e `k'` sono determinati dalla seguente equazione:

```
h'/k' = (k - h * d) / k
```

dove `h'/k'` è in forma irriducibile.
Esempio:

```
5/28 = 1/d -1/d * h'/k'
d = floor(28/5) = 5
h'/k' = (28 - 25)/28 = 3/28

3/28 = 1/d -1/d * h'/k'
d = floor(28/3) = 9
h'/k' = (28 - 27)/28 = 1/28
```

Dunque

```
5/28 = 1/5 - 1/(5*9) + 1/(5*9*28) = 1/5 - 1/45 + 1/1260
```

Si richiede di scrivere un programma che:

1. Chieda all'utente di immettere due interi `a` e `b` (con `0 < a <= b`, non si richiede il controllo della correttezza dell'input).
2. Calcoli e stampi lo sviluppo di `a/b` come

   ```
   a/b = 1/d1 - 1/(d1 * d2) + 1/(d1 * d2 * d3) - 1/(d1 * d2 * d3 * d4) ...
   ```

Per memorizzare una frazione potete usare una di queste opzioni:

- un array di due interi:

  ```
  int fraz[2];
  ```

- una struttura con due campi:

  ```
  struct frazione {
      int numeratore;
      unsigned denominatore;
  };

  struct frazione fraz;
  ```

Parte 3: Frazioni Egizie. Esercizio inverso

Realizzate un programma che avendo in input un intero `n > 0` e `n` interi `d1 d2 ... dn` , ognuno dei quali `> 1` , produca in output la frazione `a/b` data da:

```
a/b = 1/d1 - 1/(d1*d2) + 1/(d1*d2*d3) - 1/(d1*d2*d3*d4) ... 1/(d1*d2* ... *dn)
```

La frazione `a/b` deve essere in forma irriducibile.

Esempio.
Supponiamo che l'input sia:

```
5
4 2 6 3 11
```

Allora l'output deve essere:

```
221/1584
```

dato che:

$$\frac{1}{4} - \frac{1}{4*2} + \frac{1}{4*2*6} - \frac{1}{4*2*6*3} + \frac{1}{4*2*6*3*11} = \frac{221}{1584}\,.$$

Note

L'espansione *egizia* di una frazione h/k come è descritta nella Parte 2 non è univocamente determinata: ci possono essere più successioni della forma descritta che esprimono h/k. L'espansione fornita dall'algoritmo descritto è la più corta possibile. Ad esempio, l'espressione fornita nella Parte 3 per 221/1584 non è la più corta, dato che l'algoritmo descritto nella Parte 2 ne fornisce una con solo quattro termini.