Esercizi Programming Contest

Alberto Montresor

22 maggio 2012

Alcuni degli esercizi che seguono sono associati alle rispettive soluzioni. Se il vostro lettore PDF lo consente, è possibile saltare alle rispettive soluzioni tramite collegamenti ipertestuali. Altrimenti, fate riferimento ai titoli degli esercizi. Ovviamente, si consiglia di provare a risolvere gli esercizi personalmente, prima di guardare la soluzione.
Per molti di questi esercizi l'ispirazione è stata presa dal web. In alcuni casi non è possibile risalire alla fonte originale. Gli autori originali possono richiedere la rimozione di un esercizio o l'aggiunta di una nota di riconoscimento scrivendo ad `alberto.montresor@unitn.it`.

1 Esercizi

2 Problemi

Problemi presi da:

- http://www.programming-challenges.com
- http://acm.uva.es

2.1 $3n + 1$ (ID: 100)

Consider the following algorithm to generate a sequence of numbers. Start with an integer n. If n is even, divide by 2. If n is odd, multiply by 3 and add 1. Repeat this process with the new value of n, terminating when $n = 1$. For example, the following sequence of numbers will be generated for $n = 22$:

22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1

It is conjectured (but not yet proven) that this algorithm will terminate at $n = 1$ for every integer n. Still, the conjecture holds for all integers up to at least 2^{50}. For an input n, the cycle-length of n is the number of numbers generated up to and including 1 and n. In the example above, the cycle length of 22 is 16. Given any two numbers i and j, you are to determine the maximum cycle length over all numbers between i and j, including both endpoints. Input The input will consist of a series of pairs of integers i and j, one pair of integers per line. All integers will be less than 1,000,000 and greater than 0. Output For each pair of input integers i and j , output i, j in the same order in which they appeared in the input and then the maximum cycle length for integers between and including i and j . These three numbers should be separated by one space, with all three numbers on one line and with one line of output for each line of input.

```
Sample Input
1 10
100 200
201 210
900 1000
```

```
Sample Output
1 10 20
100 200 125
201 210 89
900 1000 174
```

Soluzione Sezione 4.1

2.2 Cutting Sticks (ID: 10003)

You have to cut a wood stick into several pieces. The most affordable company, Analog Cutting Machinery (ACM), charges money according to the length of the stick being cut. Their cutting saw allows them to make only one cut at a time. It is easy to see that different cutting orders can lead to different prices. For example, consider a stick of length 10m that has to be cut at 2m, 4m, and 7m from one end. There are several choices. One can cut first at 2, then at 4, then at 7. This leads to a price of 10 + 8 + 6 = 24 because the first stick was of 10m, the resulting stick of 8m, and the last one of 6m. Another choice could cut at 4, then at 2, then at 7. This would lead to a price of 10 + 4 + 6 = 20, which is better for us. Your boss demands that you write a program to find the minimum possible cutting cost for any given stick.

Input The input will consist of several input cases. The first line of each test case will contain a positive number l that represents the length of the stick to be cut. You can assume $l < 1,000$. The next line will contain the number n $(n < 50)$ of cuts to be made. The next line consists of n positive numbers c_i $(0 < ci < l)$ representing the places where the cuts must be made, given in strictly increasing order. An input case with $l = 0$ represents the end of input.

Soluzione Sezione 4.2

2.3 Vito's Family (ID: 10041)

The famous gangster Vito Deadstone is moving to New York. He has a very big family there, all of them living on Lamafia Avenue. Since he will visit all his relatives very often, he wants to find a house close to them. Indeed, Vito wants to minimize the total distance to all of his relatives and has blackmailed you to write a program that solves his problem.

Input The input consists of several test cases. The first line contains the number of test cases. For each test case you will be given the integer number of relatives r $(0 < r < 500)$ and the street numbers (also integers) $s_1, s_2, ..., s_i, ..., s_r$ where they live $(0 < s_i < 30000)$. Note that several relatives might live at the same street number.

Output For each test case, your program must write the minimal sum of distances from the optimal Vito's house to each one of his relatives. The distance between two street numbers s_i and s_j is $d_{i,j} = |s_i - s_j|$.

```
Sample Input          Sample output
2                     2
2 2 4                 4
3 2 4 6
```

Soluzione Sezione 4.3

3 Soluzioni

4 Soluzioni

4.1 $3n + 1$

Il codice seguente utilizza un array di appoggio di dimensione S, dove memorizzare la lunghezza delle sequenze per tutti i numeri da 1 a S. Questo può essere utilizzato per velocizzare il calcolo per tutti gli indici compresi fra i e j; non possiamo garantire che un qualunque array sarà sufficiente per memorizzare tutti i possibili valori.

```
for i:= 2 to S do
  computed[i] := nil
computed[1]=0

length(n)
  if (n < S)
    if (computed[x] = 0)
      if ((n & 1) > 0)
        computed[x] := length(3*x+1)+1
      else
        computed[x] := length(n >> 1)+1
      return computed[x]
    else
      if ((n & 1) > 0) {
        return length(3*n+1)+1
      else
        return length(n >> 1)+1
```

4.2 Cutting Sticks

Siano $T[1] \dots T[n]$ i punti in cui tagliare un bastone di lunghezza L. Aggiungo $T[0] = 0$ e $T[n+1] = L$. Denoto con $T[i \dots j]$ un problema di taglio per un bastone di lunghezza $T[j] - T[i]$ (a partire da $T[i]$ fino a $T[j]$), da tagliare nei punti $T[i], \dots, T[j]$. Si noti che $T[0..n+1]$ corrisponde al mio problema iniziale.
Un taglio completo $C[i,j] = C[i..k]$ / $T[k]$ / $C[k+1..j]$ è dato da un taglio ad altezza $T[k]$ e due tagli completi $C[i..k]$ e $C[k..j]$ dei sottoproblemi $T[i..k]$ e $T[k..j]$.
Il costo di un taglio $C[i..j]$ è denotato $|C[i..j]|$.
Un taglio completo $C[i..j]$ per un problema $T[i..j]$ è ottimale se non esiste un altro taglio completo $C'[i..j]$ per $T[i..j]$, tale per cui $|C'[i..j]| < |C[i..j]|$.

Sottostruttura ottimale Se $C[i,j] = C[i..k]$ / $T[k]$ / $C[k+1..j]$ è un taglio completo ottimale di T[i..j], allora: - $C[i..k]$ è un taglio completo ottimale per $T[i..k]$ - $C[k..j]$ è un taglio completo ottimale per $T[k..j]$

Dimostrazione Per assurdo; se esistesse $C'[i..k]$ taglio ottimale completo per $T[i..k]$ tale che $|C'[i..k]| < |C[i..k]|$, allora

$$|C'[i..k] \text{ / } T[k] \text{ / } C[k..j]| < |C[i..j]|$$

assurdo. La dimostrazione per $C[k..j]$ è similare.

Definizione ricorsiva Per tutti i valori che vanno da 1 a k, il costo è

$$m[i,j] = 0 \qquad \text{per } i >= j-1$$
$$m[i,j] = \min_{i<k<j} \{m[i,k] + m[k,j]\} + (T[j] - T[i]) \qquad \text{altrimenti}$$

Esempio Consideriamo un bastone lungo 5, e due tagli a 1 e 2. Se taglio prima in corrispondenza di 1, pago 5 + 4 = 9. Se taglio prima in corrispondenza di 2, pago 5 + 2 = 7.
Usiamo il nostro algoritmo:

```
T[0]=0 T[1]=1 T[2]=2 T[3]=5
m[0,3] = min { m[0,1] + m[1,3] + (T[3]-T[0]),5+4=9
               m[0,2] + m[2,3] + (T[3]-T[0]) } 5+2=7
m[0,2] = m[0,1] + m[1,2] + (T[2]-T[0])  0+0+2=2
m[1,3] = m[1,2] + m[2,3] + (T[3]-T[1]) 0+0+4=4
```

Versione Memoized dell'algoritmo:

```
CutStick(T[1..n], L)
  T[0] := 0
  T[n+1] := L
  for i=0 to n+1 do
    for j=0 to n+1 do
      if (i>=j-1)
        m[i,j] := 0
      else
        m[i,j] := nil
  do-CutStick(T, 0, n+1)

do-CutStick(T[0..n+1], i, j)
  if m[i,j] = nil do
    m[i,j] := +infty
    for k := i+1 to j-1 do
      q := do-CutStick[i,k] + do-CutStick[k,j] + (T[j]-T[i])
      if (q < m[i,j])
        m[i,j] := q
  return m[i,j]
```

4.3 Vito's Family (ID: 10041)

In questo caso, possiamo evitare di utilizzare programmazione dinamica perché si tratta di un semplice problema di ricerca.
Assumiamo che stiamo considerando un caso alla volta e assumiamo che i valori in input siano collocati in un array $A[1..n]$.
Consideriamo l'elemento mediano. Ci sono due casi:

- Se n è dispari, si sceglie il mediano stesso (indice $(n+1)/2$);
- Se n è pari, ci sono due mediani: uno qualunque dei due mediani va bene, così come tutti i numeri civici compresi fra essi. I mediani hanno indici: $\lfloor (n+1)/2 \rfloor$ e $\lceil (n+1)/2 \rceil$.

Il costo di scegliere un numero civico pari a $A[k]$ è uguale a :

$$C(A[k]) = \sum_{i=1}^{n} |A[i] - A[k]|.$$

Caso n dispari Vogliamo dimostrare che qualunque scelta $A[k] + i$, per $i \geq 1$, ha un costo superiore (simmetricamente, per $A[k] - i$). Banalmente

- Tutti i nodi $A[1..k]$ a sinistra della mediana, mediana inclusa, vedono il loro costo incrementarsi di i.

- Ben che vada, tutti i nodi $A[k+1..n]$ vedono il loro costo decrementare di i. Per alcuni valori di i, tuttavia, il costo può anche aumentare.

Poiché gli indici compresi fra 1 e k sono in numero maggiore degli indici compresi fra $k+1$ e n, la distanza per la scelta $A[k]+i$ è maggiore della distanza per la scelta $A[k]$.

Caso n pari Qui abbiamo due mediani. Vogliamo dimostrare che qualunque scelta compresa fra $\lfloor (n+1)/2 \rfloor$ e $\lceil (n+1)/2 \rceil$ va bene.
La dimostrazione è simile al precedente: se scelgo un numero civico inferiore al mediano di destra, mi aumenta il costo; se trovo un numero civico superiore a quello di sinistra, mi aumenta il costo. Se i due numeri civici sono distinti, scegliere un qualunque valore compreso fra i due non cambia il costo.

Algoritmo finale Quindi il nostro algoritmo può lavorare in questo modo: utilizzo l'algoritmo della selezione della mediana in tempo atteso lineare, per poi utilizzare la formula sopra per calcolare il valore finale.