Note

1. La leggibilità è un prerequisito: parti difficili da leggere potranno essere ignorate.
2. Quando si presenta un algoritmo è fondamentale spiegare l'idea soggiacente e motivarne la correttezza.
3. L'efficienza è un criterio di valutazione delle soluzioni proposte.
4. Si consegnano tutti i fogli, con nome, cognome, matricola e l'indicazione *bella copia* o *brutta copia*.

Domande

Domanda A (5 punti) Data la ricorrenza $T(n) = 5T(n/3) + (n-2)^2$, trovare la soluzione asintotica.

Domanda B (4 punti) Si consideri una tabella hash di dimensione $m = 8$, gestita mediante chaining (liste di trabocco) con funzione di hash $h(k) = k \bmod m$. Si descriva in dettaglio come avviene l'inserimento della sequenza di chiavi: 16, 4, 22, 14, 28.

Domanda C (5 punti) Scrivere una funzione ricorsiva `subseq(X,Y,m,n)` che date due sequenze $X[1..m]$ e $Y[1..n]$, di lunghezza m e n rispettivamente, verifica se X è una sottosequenza di Y e restituisce un valore booleano conseguente. Valutarne la complessità.

Esercizi

Esercizio 1 (7 punti) Scrivere una funzione `perm(m,n)` che dati due numeri interi m e n, maggiori o uguali di 0, verifica se uno dei due numeri può essere ottenuto permutando le cifre dell'altro. Ad esempio 915 e 159 sono uno la permutazione dell'altro mentre 911 e 19 no. Attenzione al ruolo degli zeri, ad es. 150 è la permutazione di 51, dato che $51 = 051$. Valutarne la complessità.
(*Suggerimento*: Il counting sort può fornire una ispirazione.)

Esercizio 2 (9 punti) Dare un algoritmo per individuare, all'interno di una stringa $a_1 \dots a_n$ una sottosequenza (quindi una sequenza di caratteri possibilmente non consecutivi) palindroma di lunghezza massima. Ad esempio, nella stringa "*corollario*" la sottosequenza palindroma di lunghezza massima è "*orllro*". Più precisamente:

i. dare una caratterizzazione ricorsiva della lunghezza massima $l_{i,j}$ di una sottosequenza palindroma di $a_i \dots a_j$;

ii. tradurre tale definizione in un algoritmo (bottom up o top down con memoization) che determina la lunghezza massima;

iii. trasformare l'algoritmo in modo che fornisca anche la sottosequenza, non solo la sua lunghezza;

iv. valutare la complessità dell'algoritmo.

Nota: Correzione, risultati e visione dei compiti: *Lunedì 17 Settembre, ore 14:00, 1BC/45*

1) $T(n) = \underset{a}{5}\ T(\underset{b}{n/3}) + (n-2)^2$ CORREZIONE APPELLO

$f(n)$

$n^{\log_3 5}$ $\quad 1 < \log_3 5 < 2$

$f(n) = \Omega(n^{\log_3 5 + \varepsilon})$ $\quad 0 < \varepsilon < \underbrace{2 - \log_3 5}_{>0}$

$\lim_{n\to\infty} \frac{(n-2)^2}{n^{\log_3 5 + \varepsilon}} = \infty$ $\rightsquigarrow T(n) = \Theta(f(n))$

$\Theta(n^2)$

a manca la regolarità

$\exists\ 0 < k < 1$ t.c. n_0 $\forall n \geq n_0$

$a f(\frac{n}{b}) \leq k f(n)$

$5(\frac{n}{3} - 2)^2 \leq k(n-2)^2$

osservo: $2 > \frac{2}{3}$, la posso maggiorare

$5(\frac{n}{3} - 2)^2 < 5(\frac{n}{3} - \frac{2}{3})^2$

$= 5(\frac{n-2}{3})^2$

$= \frac{5}{9}(n-2)^2$ $\quad k = \frac{5}{9}$

2) tabella hash

3) Algoritmo ricorsivo, X subsequenza di Y

X $x_1 \dots x_m$

Y $y_1 \dots y_n$

```
subseq ( X, Y, m, n )
  if m=0
    return true
  else if m>n
    return false
  else
    if X[m] = Y[n]
      return subseq(X, Y, m-1, n-1)
    else
      return subseq(X, Y, m, n-1)
```

$\Theta(n)$

4) perm(m,n)

uno è permutazione dell'altro

Es 150 e 015 ok

```
for i=1 to 9
  D[i]=0
while (m>0)
  d = m % 10
  if d>0
    D[d]++
  m = m div 10
while n>0
  d = n % 10
  if d>0
    D[d]--
  n = n div 10
```

D 


```
i=1
while (i<=9 and D[i]=0)
  i++
return (i>9)
```

$\Theta(\log_2 m + \log n)$

↓

conto il numero di cifre di n e m.

5) $a_1 \dots a_n$

massima sottosequenza palindroma

$l_{i,j}$ = lunghezza della più lunga sottoseq. pal. di $a_i \dots a_j$

1.
$$l_{i,j} = \begin{cases} 0 & \text{se } i > j \\ 1 & \text{se } i = j \\ * & \text{se } i < j \text{ e } a_i \neq a_j \quad * \max\{l_{i+1,j}, l_{i,j-1}\} \\ * & \text{se } i < j \text{ e } a_i = a_j \quad * l_{i+1,j-1} + 2 \end{cases}$$

```
palSeq(A, n)
  for i = 1 to n
      L[i, i-1] = 0
      L[i, i] = 1
  for i = 1 to n
    for len = 2 to n-i+1
      j = i + len - 1
      if A[i] = A[j]
          L[i,j] = L[i+1, j-1] + 2
      else
          L[i,j] = max{L[i+1, j], L[i, j-1]}
  return L[1, n]
```

```
palSeq(A,n)
    for i=1 to n
        for j=i-1 to n
            L[i,j] = -1
    palSeqTD(A,1,n,L)
```

TD = top down

```
palSeqTD(A,i,j,L)
    if L[i,j]<0
        if i > j
            L[i,j]=0
        else if i = j
            L[i,j]=1
        else if i<j
            if A[i]=A[j]
                L[i,j] = palSeqTD(A,i+1,j-1,L)+2
            else
                L[i,j] = max{palSeqTD(A,i,j-1), palSeqTD(A,i+1,j,L)}
```

```
getPal (A, i, j, L, P)
  if i ≤ j
    if i = j
      return A[i]
    else if P[i,j] = c
      return A[i] + getPal (A, i+1, j-1, L, P) + A[j]
    else if P[i,j] = c
      return getPal (A, i, j-1, L, P)
    else
      return getPal (A, i+1, j, L, P)
  else
    return " ";
```