Esercitazione12_Latex

October 25, 2023

1 Esercitazione 10

1.1 Es 1. Sequenze numero dispari a/b

Testo

Dare lo pseudo-codice di un algoritmo che dato l'intero n, stampa tutte le stringhe lunghe n con simboli in {a, b} che contengono un numero dispari di simboli a ed un numero dispari di simboli b. Ad esempio per n = 3 non viene stampato nulla mentre per n = 4, le stringhe da stampare sono: abbb babb bbab bbba baaa abaa aaba aaab
L'algoritmo deve avere complessita' O(nS(n)) dove S(n) e' il numero di stringhe da stampare. Motivare la complessit'a del vostro algoritmo.

Idea

Soluzione

[ ]:
```
def stampaDispariAux(x, i, n, na, nb):
        if i==n:
                if na % 2 != 0:
                        print(''.join(x))
        else:
                x[i]='a'
                stampaDispariAux(x, i+1, n, na+1, nb)
                x[i]='b'
                stampaDispariAux(x, i+1, n, na, nb+1)

def stampaDispari(n):
        x = [None for _ in range(n)]
        if n % 2 == 0:
                stampaDispariAux(x, 0, n, 0, 0)

# L'algoritmo precedente arriva all'ultimo carattere e decide se stampare o meno
# la sequenza: il che significa che le genera tutte e 2^n.
# E' possibile fare meglio e generare solo quelle "giuste" al prezzo di␣
 ↪complicare
# un pochino la soluzione

def allC(x, i, n, c):
        for j in range(i,n):
```

```
                x[j] = c
        print(''.join(x))

def stampaKa(x, i, n, k):
        if i==n:
                print(''.join(x))
        elif k == n-i:
                # mi mancano k 'a' da mettere e ho k posizioni da sistemare
                allC(x, i, n, 'a')
        elif k==0:
                # ho messo tutte le 'a' e devo completare con tutte 'b'
                allC(x, i, n, 'b')
        else:
                x[i]='a'
                stampaKa(x, i+1, n, k-1)
                x[i]='b'
                stampaKa(x, i+1, n, k)

def stampaDispariOpt(n):
        x = [None for _ in range(n)]
        if n % 2 == 0:
                for k in range(1,n,2):
                        stampaKa(x, 0, n, k)
```

Esecuzione

[ ]:
```
stampaDispariOpt(2)
stampaDispariOpt(4)
```

```
ab
ba
abbb
babb
bbab
bbba
aaab
aaba
abaa
baaa
```

1.2 Es 2. m,n,k

Testo

Dare lo pseudo-codice di un algoritmo che presi i tre interi n, m e k, stampa tutte le sequenze di n interi positivi con interi di valore al piu' m e nelle quali nessun intero compare pi'u di k volte. Ad esempio per n = 3, m = 2 e k = 2 le sequenze da stampare sono:
1,1,2 - 1,2,1 - 2,1,1 - 1,2,2 - 2,1,2 - 1,2,2

L'algoritmo deve avere complessita' O(nS(n)) dove S(n) e' il numero di sequenza da stampare. Motivare la complessita' del vostro algoritmo.

Idea

tenere un vettore indicizzato su [1,..,m] che tiene quanti numeri j sono stati messi nella sequenza che si sta generando (ovviamente questa e' informazione ridondante, ma evita di riguardarsi x ogni volta)

Soluzione

[ ]:
```
def generaSeq(n, m, k):
        T = [0 for _ in range(m+1)]  # occorrenze dell'intero i usate in x
        x = [None for _ in range(n)] # sequenza in costruzione
        generaAux(n, m, k, x, 0, T)

def generaAux(n, m, k, x, i, T):
        if i==n:
                print(x)
                return
        for j in range(1, m+1):
                if T[j]<k:
                        T[j] += 1
                        x[i] = j
                        generaAux(n, m, k, x, i+1, T)
                        T[j] -= 1
```

Esecuzione

[ ]:
```
generaSeq(3, 2, 2)
```

```
[1, 1, 2]
[1, 2, 1]
[1, 2, 2]
[2, 1, 1]
[2, 1, 2]
[2, 2, 1]
```

1.3 Es 3. Matrici binarie senza 1 adiacenti

Testo

Dare lo pseudo-codice di un algoritmo che dato l'intero n, stampa tutte le matrici binarie n x n con la proprieta' che nella matrice non compaiono mai due uni adiacenti in orizzontale, in verticale o in diagonale.
L'algoritmo deve avere complessita' O(n2S(n)) dove S(n) e' il numero di matrici da stampare. Motivare la complessita' del vostro algoritmo.

Idea

Soluzione

[ ]:
```
def stampaMatriciAux(M, i, j):
        n = len(M)
        m = len(M[0])
        if i == n: #and j == m-1:
                print(M)
                return
        if j==m-1:
                ii = i+1
                jj = 0
        else:
                ii = i
                jj = j+1
        M[i][j] = 0
        stampaMatriciAux(M, ii, jj)
        if (i == 0 and j == 0) or (i == 0 and  M[i][j-1] == 0) or (j == 0 and␣
 ↪M[i-1][j] == 0 and M[i-1][j+1]) or (M[i-1][j] == 0 and M[i][j-1] == 0 and␣
 ↪M[i-1][j-1] == 0 and (j+1>n-1 or M[i-1][j+1]==0)):
                        M[i][j] = 1
                        stampaMatriciAux(M, ii, jj)

def stampaMatrici(n):
        M = [[None for _ in range(n)] for _ in range(n)]
        stampaMatriciAux(M, 0, 0)
```

Esecuzione

[ ]:
```
stampaMatrici(2)
```

```
[[0, 0], [0, 0]]
[[0, 0], [0, 1]]
[[0, 1], [0, 0]]
[[0, 1], [1, 0]]
[[1, 0], [0, 0]]
```