Esercitazione10_Latex

October 25, 2023

1 Esercitazione 10

1.1 Es 1. M-lista di valore minimo

Testo

Data una matrice n x m di interi M = [mi,j], una M-lista e' una sequenza (m1,j1 , m2,j2 …mn,jm) tale che 1 <= j1 <= … jm <= m. Il valore di una M-lista e' la somma degli elementi che la compongono. Progettare un algoritmo che trova un M-lista di valore minimo in O(nm)

Idea

tengo una tabella T della stessa dimensione di M T[i][j] sara' il valore minimo di una m-lista di lunghezza j T[i][j] viene computata per colonne. Ad esempio:
T[i][0] e' il min_{k in [0,i]} M[k][0]
T[i][j+1] e' il min_{k<l in [0,i]} T[k][j]+T[l][j+1]

Soluzione

[ ]:
```
def mLista(M):
        n = len(M)
        m = len(M[0])
        T = [[None for _ in range(m)] for _ in range(n)]
        T[0][0] = M[0][0]
        for i in range(1,m):
                T[0][i] = T[0][i-1]+M[0][i]
        for i in range(1,n):
                T[i][0] = min(T[i-1][0],M[i][0])
        for i in range(1,n):
                for j in range(1,m):
                        T[i][j] = min(T[i-1][j],T[i][j-1]+M[i][j])
        return T

def ricostruisci(T):
        n = len(T)
        m = len(T[0])
        i = n-1
        res = []
        for j in reversed(range(0,m)):
                while (i>0 and T[i-1][j]==T[i][j]):
```

```
                        i -= 1
                res = [i] + res
        return res
```

Esecuzione

[ ]:
```
M = [[3,4,7,-5],[1,5,1,1],[-1,3,4,5]]
T = mLista(M)
seq = ricostruisci(T)
print(T)
print(seq)
M2 = [[3,2,7,-3],[1,5,1,1],[-1,3,4,5]]
T2 = mLista(M2)
seq2 = ricostruisci(T2)
print(T2)
print(seq2)
```

```
[[3, 7, 14, 9], [1, 6, 7, 8], [-1, 2, 6, 8]]
[1, 1, 1, 1]
[[3, 5, 12, 9], [1, 5, 6, 7], [-1, 2, 6, 7]]
[0, 0, 1, 1]
```

1.2 Es 2. Sequenze/Multi-insiemi di somma n

Testo

Vengono dati in input tre interi positivi x1, x2 e x3, con x1 $<$ x2 $<$ x3, ed un intero positivo n. - Scrivere lo pseudocodice di un algoritmo che in tempo O(n) restituisce il numero di sequenze sull'alfabeto {x1, x2, x3} la somma dei cui elementi e' n. Ad esempio per x1 = 2, x2 = 4, x3 = 8 e n = 10 la risposta deve essere 10, le uniche sequenze possibili sono infatti: 2,8 - 8,2 - 2,4,4 - 4,2,4 - 4,4,2 - 2,2,2,4 - 2,2,4,2 - 2,4,2,2 - 4,2,2,2 - 2,2,2,2 - Scrivere lo pseudocodice di un algoritmo che in tempo O(n) restituisce il numero di multiinsiemi sull'alfabeto {x1, x2, x3} la somma dei cui elementi e' n. Ad esempio per x1 = 2, x2 = 4, x3 = 8 e n = 10 la risposta deve essere 4, gli unici multiinsiemi possibili sono infatti: 2,8 - 2,4,4 - 2,2,2,4 - 2,2,2,2

Idea

Allocare una matrice T con i percorsi piu' lunghi gia' computati inizialmente -1 (non computato) poi si computano tutti con la programmazione dinamica

Soluzione

[ ]:
```
def camminoMaxAux(M,i,j,T,v):
        n = len(M)
        # mi fermo se l'elemento corrente non e' il successivo del precedente
        if M[i][j] != v and M[i][j]!=v+1 :
                return 0
        # calcolo il massimo tra tutti i percorsi (alto, sinistra, basso,␣
 ↪destra)
        # se il valore non e' gia' disponibile in T[i][j]
```

```
        if T[i][j]<0:
                m1, m2, m3, m4 = 0,0,0,0
                if i-1 >= 0:
                        m1 = camminoMaxAux(M, i-1, j, T, M[i][j])
                if j-1 >=0:
                        m2 = camminoMaxAux(M, i, j-1, T, M[i][j])
                if i+1 < n:
                        m3 = camminoMaxAux(M, i+1, j, T, M[i][j])
                if j+1 < n:
                        m4 = camminoMaxAux(M, i, j+1, T, M[i][j])
                # aggiorno T[i][j]
                T[i][j]         = 1 + max(m1,m2,m3,m4)
        return T[i][j]

def camminoMax(M):
        n = len(M)
        T = [[-1 for _ in range(n)] for _ in range(n)]
        for i in range(n):
                for j in range(n):
                # calcolo quelli non ancora computati dalla funzione ausiliaria␣
 ↪ricorsiva
                        if T[i][j]<0:
                                T[i][j] = camminoMaxAux(M, i, j, T, M[i][j])
        return T
```

Esecuzione

[ ]:
```
A =[[3,6,2],[7,1,9],[4,8,5]]
B =[[9,7,6],[8,2,5],[1,3,4]]
T = camminoMax(A)
print(T)
print(max(max(T)))
T = camminoMax(B)
print(T)
print(max(max(T)))
```

```
[[1, 1, 1], [1, 1, 1], [1, 1, 1]]
1
[[1, 1, 2], [2, 6, 3], [1, 5, 4]]
6
```