LIS

Sunday, 19 November 2023 09:49

- Determinare una tra le più lunghe sottosequenze crescenti di X
- Esempio pratico:
  $X = < 2, 4, 7, 6, 11, 13, 21, 14, 1 >$
  $S = < 2, 4, 6, 11, 13, 21 >$
- Spiegazione esaustiva:
  Possiamo comprendere facilmente il sottoproblema è definito da i, siccome dobbiamo solamente iterare attraverso le sottostringe di X.
  Detto questo, per trovare la sottosequenza crescemente maggiore si deve fare così:

  $\max\big(Lis(X[0:0]), Lis(X[0:1]), Lis(X[0:2]), \ldots, Lis(X[0:-1])\big)$

  Questo è scritto in codice python, per chi non sapesse il python:

  X[0:0] -> Array che inizia da elemento 0 e prende 0 elementi, aka array vuoto
  X[0:1] -> Array che inizia da elemento 0 e prende 1 elemento, quindi solamente elemento 0
  X[0:2] -> Parte da 0 e prende 2 elementi, quindi 2 elementi: <X[0], X[1]>
  X[0:-1] -> Parte da 0 e prende -1 elementi, in python -1 inserito li rappresenta la fine dell'array

  Quindi prende tutti gli elementi di X

  Quindi, nel nostro caso specifico:

  $Max(Lis(\epsilon), Lis(< 2 >), Lis(< 2, 4 >), Lis(2, 4, 7), Lis(2, 4, 7, 6), \ldots)$

  Che ci darà il seguente risultato:

  $Max(|\epsilon|, | < 2 > |, | < 2, 4 > |, |< 2, 4, 7 >|, |< 2, 4, 6 >|, \ldots)$

  Noi, come possiamo ottenere il seguente risultato con 1 sola funzione?
  E' estremamente comprensibile che ci serve:
  1) Funzione per calcolare il LIS di X con i dimensione: X_i
  2) Funzione che calcola il massimo di tutte le lunghezze di X

  Chiameremo la funzione N^1 funzione ausiliare.
- Algoritmo
  - LIS:
    - Caso base: i=0
      Quando i=0, X è vuoto, quindi torneremo nulla
      $c_i = 0$
    - Passo ricorsivo: i>0

Qui noi dobbiamo iterare per la lunghezza di X e prendere il massimo chiamando la funzione ausiliare

$c_i = MAX(LIS_{AUX}(X, j), j = 0\ to\ i)$

Riscriviamolo meglio:

$$c_i = \begin{cases} 1 & i \leq 1 \\ MAX\left(c_j^{aux}, where\ j < i\right) & else \end{cases}$$

Un attimo di spiegazione:

Noi qui iteriamo per tutti i valori di j < i

E per tutti questi valori chiamiamo la funzione ausiliare con parametro j

Che è la lunghezza di X, e poi facciamo il massimo

- Problema ausiliario:
  - i=0

    Quando i=0, abbiamo X vuoto

    $c_i = \epsilon$
  - i>0

    Qui noi dobbiamo iterare per tutte le sottostringe di i dove

    $x_j < x_i$

    $c_i = MAX\left(LIS_{auc}(X, j)|x_i, where\ x_j < x_i, j < i\right)$

    E poi dobbiamo aggiungere xi

Riscriviamolo meglio:

$$c_i^{aux} = \begin{cases} \epsilon & i = 0 \\ MAX\left(c_j^{aux}|x_i, where\ x_j < x_i, j < i\right) & else \end{cases}$$

Un attimo di spiegazione:

Quando i non è 0, noi proviamo tutte le possibili combinazioni di j < i

Dove $x_j < x_i$

Ed appena ne troviamo una richiamiamo la funzione ausiliare

E poi la confrontiamo con gli altri valori $x_0 \leq x_j$

- Pseudocodice iterativo:
  - LIS(X, n):

```
Max = 0
For i=2 to n:
    Temp = LIS-AUX(X, i)
    If LIS-AUX(X, i) > max:
        Max = LIS-AUX(X, i)
Return max
```

  - LISAUX(X, n):

```
C[] = [n] // Creo array di n valori
```

```
B[] = [n]
M = 0
C[1] = 1

For i=2 to n:
    Max = 0
    H = 0
    For j=1 to i-1:
        If c[j] > max:
            Max = c[j]
            H = j
            B[i] = h
    C[i] = c[j] + 1
Return c[n], m
```

- PRINT
  - LIS(X, b, i)

    ```
    If i=0:
        Printina(0)
    If i=1:
        Printina(X[i])
    LIS(X, b, b[i])
    Printina(X[i])
    ```

  - LIS-AUX(X, b, i)

    ```
    If i=0:
        Printina(0)
    If i=1:
        Printina(X[i])
    LIS(X, b, b[i])
    Printina(X[i])
    ```