Dimostrazione

1) banale

2) a) banale; se ci fosse un altro carattere prima di x_i questo contraddirebbe l'ipotesi che $\overline{LIS}(X_i)$ è una sequenza crescente

b) la sequenza $\langle x_j, x_i \rangle$ è $\overline{IS}(X_i)$, e quindi $\overline{LIS}(X_i) \geq 2$

$\Downarrow$

$$Z = \overline{LIS}(X_i) = \langle \underbrace{Z_{k-1}}_{\neq \varepsilon}, x_i \rangle$$

$$Z = \langle x_{i_1}, x_{i_2}, \dots, x_{i_k} \rangle \qquad i_k = i$$

$$Z_{k-1} = \langle x_{i_1}, x_{i_2}, \dots, x_{i_{k-1}} \rangle \qquad x_{i_{k-1}} < x_i$$

$Z_{k-1} = \overline{LIS}(X_{i_{k-1}})$: se non fosse la più lunga ne esiste un'altra: gli aggiungo $x_{i_k} = x_i$ e ottengo una $\overline{LIS}(X_i)$ di lunghezza $> |Z| = k$: assurdo. Ora dimostro che tra tutte le $\overline{LIS}$ dell'insieme è quella più lunga: se così non fosse potrei aumentare $|Z| = k$: assurdo.

Ricorrenza sui costi:

notazione: $\ell(i) = |\overline{LIS}(X_i)|$

$$\ell(i) = \begin{cases} 1 & i = 1 \\ 1 + \max\{\ell(j) : 1 \leq j < i,\ x_j < x_i\} & i > 1 \end{cases}$$

convenzione: $\max \emptyset = 0$ (così ho uniformato i 2 casi 2a e 2b)

Per ricostruire la sequenza:

$$\overline{LIS}(X_i) \begin{cases} \langle x_i \rangle \\ \langle \overline{LIS}(X_{\bar{j}}), x_i \rangle \end{cases}$$

info addizionale è $\bar{j}$

$$prev(i) = \begin{cases} 0 & 1^\circ \text{ caso} \\ \bar{j} & 2^\circ \text{ caso} \end{cases}$$

↳ "previous"

1 valore $\forall$ intero $\rightarrow$ vettore di n posizioni

Accumulare il max degli $\ell(i)$:

$$len = \max_{1 \le i \le n} \{\ell(i)\}$$

Altra informazione addizionale:

$end = i$ se $\overline{LIS}(X_i) = LIS(X)$

mantiene cioè l'indice dell'ultimo carattere della LIS

Com'è fatta la LIS

$$\langle \ldots, x_{prev(prev(i))}, x_{prev(i)}, x_i \rangle$$

Algoritmo ricorsivo? No, ci sono problemi ripetuti $\rightarrow$ complessità esponenziale

Codice bottom-up

cioè a partire da $\overline{LIS}(X_1)$

```
LIS (X)
  n ← length (X)
  l [1] ← 1
  len ← 1
  end ← 1
  prev [1] ← 0
  for i ← 2 to n do
      l [i] ← 1 ; prev [i] ← 0
      for j ← 1 to i-1 do
          if (X_j < X_i) then
              if l [i] < 1 + l [j] then
                  l [i] ← 1 + l [j] ; prev [i] ← j
      if len < l [i] then
          len ← l [i]
          end ← i
  return len, prev, end
```

Complessità: (conto i confronti tra caratteri)

$$T(n) = \sum_{i=2}^{n} \sum_{j=1}^{i-1} 1 = \sum_{i=2}^{n} (i-1) = \frac{n(n-1)}{2} = \Theta(n^2)$$

n sottoproblemi solo, ma tempo lineare per ognuno

Stampare la stringa:

```
R_PRINT (X, prev, i)
    if (prev(i) ≠ 0) then  R_PRINT (X, prev, prev[i])
    print (X_i)
    return
```

i → indice da cui inizia

invocato con P_PRINT (X, prev, end)