CORREZIONE SECONDO APPELLO

A $T(n) = 5T\left(\frac{n}{3}\right) + 2n^2$

con metodo dell' esperto

$n^{\log_b a} = n^{\log_3 5} \rightarrow \log_3 5$ certamente < 2

$f(n) = 2n^2$

$f(n) = \Omega(n^{\log_3 5 + \varepsilon}) \qquad 0 < \varepsilon < 2 - \log_3 5$

verifichiamo la regolarità

$a f\left(\frac{n}{b}\right) \leq K f(n) \qquad (K < 1)$

$5 \cdot \cancel{2}\left(\frac{n}{3}\right)^2 \leq K \cancel{2} n^2$

vale ad esempio per questa K

$\frac{5}{9} n^2 \leq K n^2 \rightarrow \boxed{K = \frac{5}{9}}$

quindi $T(n) = \Theta(f(n)) = \Theta(n^2)$

B $h_1(k) = K \bmod m \rightarrow K \bmod 9$

$h_2(K) = 1 + K \bmod 7$

doppio hash $\rightarrow h(K, i) = (h_1(K) + i\, h_2(K)) \bmod m$

$h(12, 0) = [12 \bmod 9 + \underline{0 \cdot h_2(12)}] \bmod 9$

$= 3 \bmod 9 = \boxed{3}$

$h(3, 0) = (3 \bmod 9) \bmod 9$

$= 3 \bmod 9 = 3$ <u>COLLISIONE</u>

$h(3, 1) = [3 + 1 \cdot 1 + (3 \bmod 7)] \bmod 9$

$= 7$

(c)

x (key)

x.l   x.r

MaxPath(x) = max $\sum$ chiavi nei cammini x → foglia

```
if x = nil
   return 0          → altrimenti -∞
else if x.l = nil
   return x.key + MaxPath(x.r)
else if x.r = nil
   return x.key + MaxPath(x.l)
else
   return x.key + max(MaxPath(x.l),
                      MaxPath(x.r))
```

Devo trattare questi casi perché i nodi potrebbero avere key < 0

Es 1

Ordstack = stack

```
OEmpty(S)
   return Empty(S)      → funzione della stack "normale"

Opop(S)
   return Pop(S)

Opush(x, S)
   while (not Empty(S) and Top(S) > x)
         Pop(S)
   Push(S, x)
```

$\Phi(S) = |S|$ funzione potenziale

	c	$\Delta\Phi$	$\hat{c}$
OEmpty	1	0	1
OPop	1	-1	0
OTop	1	0	1
OPush	k + 1	-k + 1	2

k → k estrazioni (numeri più grandi di x)

1 → inserimento nuovo elemento

ES 2 LIS

$x_1 \; x_2 \; x_3 \; \ldots \; x_n$ $\quad$ l_i = lunghezza di una LIS che inizia da x_i

$x_{i_1} \quad x_{i_2} \quad x_{i_3} \quad \ldots$

$$l_i = \begin{cases} 0 & i > n \\ 1 & i = n \\ 1 + \max\{l_J \mid J = i+1 \ldots n\} \text{ con } x_i \le x_J & i < n \end{cases}$$

carattere iniziale (x_i)

LIS (X, n) $\quad$ $X[1 \ldots n]$ $\quad$ uso $L[1 \ldots n]$ per memorizzare le sottosequenze $L[i]$ = lung LIS da $X[i]$

```
X[0] = -∞   // così tutte le altre saranno più lunghe
for i=0 to n
    L[i] = -1   // inizializzo l'array
LIS-REC(X, n, L, 0)   carattere iniziale sottosequenza
return L[0] - 1   carattere fasullo
```

```
LIS-REC (X, n, L, i)
  if L[i] = -1
      max = 0
      for J = i+1 to n
          if X[i] ≤ X[J]
              if max < LIS-REC(X, n, L, J)
                  max = LIS-REC(X, n, L, J)
      L[i] = max + 1
  return L[i]
```

$O(n)$