Con questo algoritmo distinguiamo i 3 casi
Sostituire $s[i]$ e $t[j]$ che costa $1 + dist(i-1, j-1)$ dove corrisponde $dist(s[1..i-1], t[1..j-1])$, nel caso in cui $s[i]$ e $t[j]$ sono uguali non impiegano il tempo 1
Cancellare $s[i]$ e trasformare $s[1..i-1]$ in $t[1..j]$ e usano $1 + dist(s[1..i-1], t[1..j])$ operazioni
Inserire $t[j]$ alla fine di $s[1..i]$ che ha totale richiedono $1 + dist(s[1..i], t[1..j-1])$

L'algoritmo ha tempo $O(mn)$

ESERCIZIO n 7 di 6.5

L'algoritmo Greedy non produce soluzioni ottime per il problema dello zaino 0-1, perché Greedy costruisce la soluzione per passi, aggiungendo elementi tenendo conto di due condizioni:
che l'elemento rispetti i vincoli del problema, che il nuovo elemento aggiunto provochi il maggiore incremento possibile al valore della funzione obiettivo.

Con la tecnica Greedy possiamo risolvere una variante del problema dello zaino 0-1, lo zaino frazionario.
Per il problema dello zaino frazionario, facciamo le stesse assunzioni del problema dello zaino normale, solo che ora di ogni elemento possiamo prendere una frazione $x_i < 1$. L'algoritmo dovrà scegliere di ogni elemento quella frazione prendere, ovviamente detto la condizione che $\sum_{i=1}^{n} v[i] x_i > w$ ottenuto il problema può essere risolto banalmente prendendo tutte le frazioni degli elementi.
Per il funzionamento dell'algoritmo supponiamo che:

$$\frac{v[1]}{w[1]} > \frac{v[2]}{w[2]} > \dots > \frac{v[n]}{w[n]}$$