Esercizio 1 Si consideri il seguente problema. Date due sequenze $X = (x_1, x_2, \ldots, x_n)$ e $Y = (y_1, y_2, \ldots, y_m)$ sull'alfabeto $\{0, 1\}$, determinare la lunghezza della più lunga sottosequenza comune ad X e Y. Ad esempio, se $X = (1, 0, 1, 0, 1)$ e $Y = (0, 0, 1, 1, 1)$ allora una sottosequenza comune di lunghezza massima è $(0, 1, 1)$. Viene proposto il seguente algoritmo:

```
INPUT due sequenze X = (x_1, x_2, ..., x_n) e Y = (y_1, y_2, ..., y_m)
    NX ← 0
    FOR i ← 1 TO n DO
        IF x_i = 0 THEN NX ← NX + 1
    ENDFOR
    NY ← 0
    FOR i ← 1 TO m DO
        IF y_i = 0 THEN NY ← NY + 1
    ENDFOR
    L ← max{min{NX, NY}, min{n − NX, m − NY}}
OUTPUT L.
```

1. *(max 5 punti)* Provare che la lunghezza prodotta in output dall'algoritmo è la lunghezza di una sottosequenza comune alle due sequenze di input.
2. *(max 3 punti)* Far veder che l'algoritmo non risolve il problema.
3. *(max 8 punti)* Dimostrare che l'algoritmo garantisce un rapporto di approssimazione limitato da 2.
4. *(max 2 punti)* Esibire un'istanza per cui il rapporto di approssimazione è esattamente 2.

Soluzione Esercizio 1 Notiamo preliminarmente che l'algoritmo calcola il numero minimo di occorrenze del simbolo 0 nelle due sequenze, il numero minimo di occorrenze del simbolo 1 nelle due sequenze e restituisce il massimo tra questi due valori.

1. Per quanto detto deve esserci un simbolo in $\{0, 1\}$ che compare esattamente L volte in una delle due sequenze ed almeno L volte nell'altra. La sottosequenza che si ottiene concatenando L volte questo simbolo è una sottosequenza comune ad entrambe le sequenze ed ha ovviamente lunghezza L.
2. Si consideri ad esempio $X = (0, 1, 0, 1, 0, 1)$ e $Y = (1, 0, 1, 0, 1, 1)$. L'algoritmo produce la lunghezza 3 mentre la sottosequenza comune di lunghezza massima è $(1, 0, 1, 0, 1)$ ed ha lunghezza 5.
3. Sia L^* la lunghezza della più lunga sottosequenza comune alle due stringhe. Trattandosi di un problema di massimizzazione dobbiamo provare che il rapporto $\frac{L^*}{L}$ non può superare 2.
   Notiamo che nè il simbolo 0 nè il simbolo 1 possono comparire in ciascuna delle due sequenze più di L volte (in caso contrario si avrebbe $\max\{\min\{NX, NY\}, \min\{n - NX, m - NY\}\} > L$). Questo implica che le sottosequenze comuni ad X e ad Y possono contenere al più L occorrenze del simbolo 0 ed al più L occorrenze del simbolo 1. Questo significa che nessuna sottosequenza comune può avere lunghezza superiore a $2L$ e quanto detto deve valere chiaramente anche per la sottosequenza comune più lunga. Pertanto:
   $$\frac{L^*}{L} \leq \frac{2L}{L} = 2.$$
4. Si consideri ad esempio $X = (0, 1, 1, 1)$ e $Y = (0, 0, 1, 0)$. L'algoritmo restituisce $L = 1$ mentre la sottosequenza comune $(0, 1)$ ha lunghezza 2.

Esercizio 2 Si consideri il seguente problema. Date due sequenze $X = (x_1, x_2, \ldots, x_n)$ e $Y = (y_1, y_2, \ldots, y_m)$ sull'alfabeto $\{0, 1\}$, trovare una sequenza di lunghezza minima fra tutte quelle che contengono come sottosequenze sia X che Y. Ad esempio se $X = (0, 1, 0, 0, 1)$ e $Y = (1, 0, 1, 1, 0, 1)$ allora una sequenza minima che le contiene entrambe è $(1, 0, 1, 1, 0, 0, 1)$.

1. *(max 6 punti)* Proporre un algoritmo di programmazione dinamica che date in input due sequenze X ed Y calcola la lunghezza della più corta sequenza che le contiene come sottosequenze. (Suggerimento: si pensi all'algoritmo di programmazione dinamica per il problema della più lunga sottosequenza comune).
2. *(max 8 punti)* Spiegare perchè algoritmo proposto è corretto.
3. *(max 4 punti)* Modificare l'algoritmo proposto in modo che produca anche la sequenza, oltre alla lunghezza.

Soluzione Esercizio 2

1. Definiamo $D(i, j)$, con $0 \leq i \leq n$ e $0 \leq j \leq m$, come la lunghezza minima per una sequenza che contiene come sottosequenze $X' = (x_1, x_2, \ldots x_i)$ e $Y' = (y_1, y_2, \ldots y_j)$.
   Per calcolare efficientemente questi valori consideriamo una tabella D di dimensioni $(n+1) \times (m+1)$ i cui valori sono definiti ricorsivamente come segue:

   $D(i, 0) = i$ per $0 \leq i \leq n$ e $D(0, j) = j$ per $1 \leq j \leq m$.
   mentre per i e j entrambi maggiori di 0 si ha

   $$D(i,j) = \begin{cases} D(i-1, j-1) + 1 & \text{se } x_i = y_j \\ \min\{D(i, j-1), D(i-1, j)\} + 1 & \text{altrimenti.} \end{cases}$$

   La lunghezza della sequenza più corta che contiene come sottosequenze sia X che Y è $D(n, m)$, ecco dunque lo pseudo-codice dell'algoritmo che la calcola:

   ```
   INPUT due sequenze X = (x1, x2, ..., xn) e Y = (y1, y2, ..., ym)
       FOR i ← 0 TO n DO D[i,0] ← i
       FOR j ← 0 TO m DO D[0,j] ← j
       FOR i ← 1 TO n DO
           FOR j ← 1 TO m DO
               IF xi = yj THEN
                   D[i,j] ← D[i-1,j-1] + 1
               ELSE
                   D[i,j] ← min{D[i,j-1], D[i-1,j]} + 1
               ENDIF
           ENDFOR
       ENDFOR
   OUTPUT D[n,m]
   ```

2. Si tratta di spiegare perchè la relazione ricorsiva D è corretta.
   Sia $1 \leq i \leq n$ e $1 \leq j \leq m$. Distinguiamo due casi:
   - $x_i = y_j$: Sia S una più corta sequenza che contiene come $X' = (x_1, x_2, \ldots x_i)$ e $Y' = (y_1, y_2, \ldots y_j)$. Chiaramente l'ultimo simbolo di S è uguale a x_i $(= y_j)$. Per cui la sequenza S' che si ottiene da S eliminando l'ultimo simbolo è una più corta sequenza che contiene $(x_1, x_2, \ldots x_{i-1})$ e $(y_1, y_2, \ldots y_{j-1})$. Quindi $D(i, j) = D(i-1, j-1) + 1$.
   - $x_i \neq y_j$: L'ultimo simbolo di S è x_i o $= y_j$. Se l' ultimo simboloè x_i allora la sequenza S' che si ottiene da S togliendo l'ultimo simbolo è una più corta sequenza che contiene $(x_1, x_2, \ldots x_{i-1})$ e $(y_1, y_2, \ldots y_j)$. Per cui $D(i, j) = D(i-1, j) + 1$. Se l' ultimo simboloè y_j allora in modo analogo si vede che $D(i, j) = D(i, j-1) + 1$. Quindi risulta sempre $D(i, j) \geq \min\{D[i-1, j], D[i, j-1]\} + 1$.

D'altro canto da una sequenza che contiene $(x_1, x_2, \ldots x_{i-1})$ e $(y_1, y_2, \ldots y_j)$ è possibile ottenere, concatenando il simbolo x_i, una sequenza che contiene X' e Y' e lo stesso vale per $(x_1, x_2, \ldots x_i)$ e $(y_1, y_2, \ldots y_{j-1})$. Dunque $D(i, j) \leq \min\{D[i-1, j], D[i, j-1]\} + 1$.

3. Per ricostruire la sequenza possiamo ricorrere alla stessa tecnica vista a lezione per la ricerca della sottosequenza comune di lunghezza massima. Definiamo una seconda tabella P di dimensioni $n \times m$ il cui generico elemento $P[i, j]$ "punterà" alla posizione della tabella D che corrisponde alla scelta del sottoproblema per il calcolo di $D[i, j]$. Più precisamente modifichiamo l'algoritmo per il calcolo della sola lunghezza come segue:

```
SEQUENZA:  INPUT le due sequenze X = (x_1, ..., x_n) e Y = (y_1, ..., y_m)
    FOR i ← 0 TO n DO D[i,0] ← i
    FOR j ← 0 TO m DO D[0,j] ← j
    FOR i ← 1 TO n DO
        FOR j ← 1 TO m DO
            IF x_i = y_j THEN
                D[i,j] ← D[i-1,j-1] + 1
                P[i,j] ← "↖"
            ELSE
                IF D[i,j-1] ≤ D[i-1,j] THEN
                    D[i,j] ← D[i,j-1] + 1
                    P[i,j] ← "↑"
                ELSE
                    D[i,j] ← D[i-1,j] + 1
                    P[i,j] ← "←"
                ENDIF
            ENDIF
        ENDFOR
    ENDFOR
OUTPUT il valore D[n,m] e la tabella P
```

Ripercorrendo all'indietro le freccie della tabella P è possibile ricostruire la sequenza. La seguente procedura stampa la sequenza nel giusto ordine.
La procedura STAMPA-SEQUENZA prende in input la tabella P, le sequenze X e Y ed essendo ricorsiva i due indici i e j ad indicare in che posizione della tabella ci si trova. All'inizio sarà richiamata con STAMPA-SEQUENZA(P, X, Y, n, m).

```
STAMPA-SEQUENZA:  INPUT la tabella P, la sequenza X, la sequenza Y, indici i e j
    IF i = 0 THEN SOL ← (y_1, ..., y_j)
    ELSE IF j = 0 THEN SOL ← (x_1, ..., x_i)
    ELSE IF P[i,j] = "↖" THEN
        SOL ← STAMPA − SEQUENZA(P, X, Y, i-1, j-1)
        SOL ← (SOL, x_i)
    ELSE IF P[i,j] = "↑" THEN
        SOL ← STAMPA − SEQUENZA(P, X, Y, i, j-1)
        SOL ← (SOL, y_j)
    ELSE
        SOL ← STAMPA − SEQUENZA(P, X, Y, i-1, j)
        SOL ← (SOL, x_i)
    ENDIF
OUTPUT SOL
```