ESERCIZIO 1.

Data una stringa binaria S vogliamo contare il numero di diverse coppie $(0,1)$ presenti nella sequenza.

Ad esempio:
- per $S = 1110100$ la risposte deve essere 1 (abbiamo infatti ___01__).
- per $S = 010010$ la risposta deve essere 8 (abbiamo infatti 01____, 0___1_, __0_1_ , *e* ___01_).

Progettare un algoritmo che risolve il problema in tempo $O(n)$.

Motivare BENE la correttezza e la complessità dell'algoritmo proposto.

Utilizziamo una tabella di dimensioni $n \times 2$.

$T[i][0] =$ **il numero di** 0 **presenti nella stringa** $S_0, S_1, \ldots S_i$
$T[i][1] =$ **il numero di coppie** $(0,1)$ **presenti nella stringa** $S_0, S_1, \ldots S_i$

La soluzione al problema sarà $T[n-1][1]$.

Resta solo da definire la ricorrenza per il calcolo delle $\Theta(n)$ celle della tabella .

La formula ricorsiva che permette di ricavare $T[i][j]$ dalle celle precedentemente calcolate è la seguente:

$$T[i][j] = \begin{cases} 1 & \text{se } i = j = 0 \text{ AND } S_0 = 0 \\ 0 & \text{se } i = 0 \\ T[i-1][0] & \text{se } j = 0 \text{ e } S_i \neq 0 \\ T[i-1][0] + 1 & \text{se } j = 0 \\ T[i-1][0] & \text{se } j = 1 \text{ e } S_i \neq 1 \\ T[i-1][1] + T[i-1][1] & \text{altrimenti} \end{cases}$$

La ricorrenza è corretta per i seguenti motivi:
- per $i = 0$ si ha $T[0][j] \neq 0$ solo se $j = S_0 = 0$

consideriamo ora i casi $i > 0$:
- se $S_i \neq 0$ allora il simbolo S_i non contribuisce in alcun modo al numero di zeri presenti fino a quel momento (vale dunque $T[i][0] = T[i-1][0]$) in caso contrario questo numero aumenta di 1 (vale dunque $T[i][0] = T[i-1][0] + 1$)
- se $S_i \neq 1$ allora il simbolo S_i non contribuisce ad alcuna coppia $(0,1)$ in $S_0, S_1, \ldots S_i$ (si ha dunque $T[i][1] = T[i-1][1]$) in caso contrario il simbolo $S_i = 1$ contribuisce a $T[i-1][0]$ coppie (può infatti formare coppia $(0,1)$ con tutti gli zeri che lo precedono), si ha dunque $T[i][1] = T[i-1][1] + T[i-1][0]$.

Implementazione:

```
def es1(S):
    n=len(S)
    T=[[0,0]  for _ in range(n)]
    if S[0]=='0': T[0][0]=1
    for i in range(1,n):
        if S[i]=='0':
            T[i][0] =T[i-1][0]+1
            T[i][1] =T[i-1][1]
        else:
            T[i][0] =T[i-1][0]
            T[i][1] =T[i-1][1]+T[i-1][0]
    return T[n-1][1]
```

Complessità $\Theta(n)$

ESERCIZIO 2

Progettare un algoritmo che, dati due interi n e k, stampa tutte le stringhe binarie di lunghezza n in cui siano presenti almeno k zeri consecutivi.

Ad esempio per $n = 4$ e $k = 2$ l'algoritmo deve stampare, non necessariamente nello stesso ordine, le seguenti 8 stringhe:
0000 0001 0010 0011 0100 1000 1001 1100

L'algoritmo proposto deve avere complessità $O(nS(n,k))$ dove $S(n,k)$ è il numero di stringhe da stampare.

Motivare BENE la correttezza e la complessità dell'algoritmo proposto

Algoritmo:

- Utilizziamo un algoritmo di backtracking per la stampa di tutte le stringhe binarie sol di lunghezza n con l'aggiunta di funzioni di taglio.
- Utiliziamo:
  1. una variabile booleana $inserito$ che vale $True$ se la stringa fin'ora costruita contiene la sequenza di k zeri, $False$ altrimenti.
  2. una variabile t che contiene il numero di cifre 0 consecutive con cui termina la stringa fin'ora costruita.
- al passo i:
  - nella cella $sol[i][j]$ inserisco 0 e incremento t inoltre, se t supera k, $inserito$ diventa $True$
  - nella cella $sol[i][j]$ inserisco 1 se $inserito$ vale $True$ oppure resta sufficiente spazio per poter inserire k zeri consecutivi (vale a dire $n - 1 - i \geq k$).

Implementazione

```
def es2(n,k):
    sol=[0]*n
    es2b(0, n, k, 0, False, sol)


def es2b(i, n, k, t, inserito, sol):
    if i==n:
        print(''.join(sol))
        return
    sol[i]='0'
    if t+1==k:
        es2b(i+1, n, k, t+1, True, sol)
    else:
        es2b(i+1, n, k, t+1, inserito, sol)
    if inserito or n-1-i>=k:
        sol[i]='1'
        es2b(i+1, n, k, 0, inserito, sol)
```

- Nota che nell'albero di ricorsione prodotto dall'esecuzione di $es2$ un nodo viene generato solo se porta ad una foglia da stampare.
- Possiamo quindi dire che la complessità dell'esecuzione di $es2(n,k)$ richiederà tempo

$$O(S(n,k) \cdot h \cdot f(n) + S(n,k) \cdot g(n))$$

dove:

- $h = n$ è l'altezza dell'albero.
- $f(n) = O(1)$ è il lavoro di un nodo interno.
- $g(n) = O(n)$ è il lavoro di una foglia

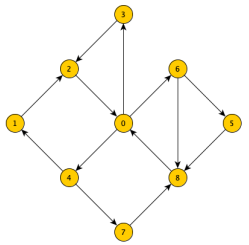
- Quindi il costo di $es2(n,k)$ è $O(nS(n))$.

ESERCIZIO 3

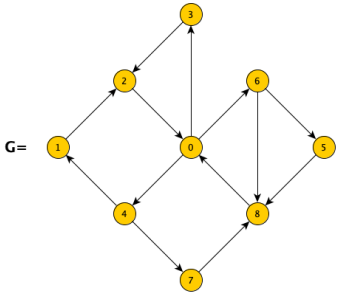
Considerate il grafo G in figura.

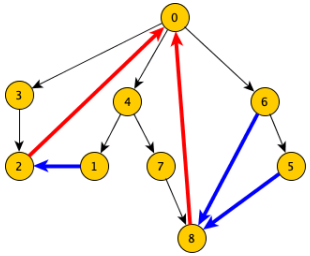
1. Riportate l'albero dfs che si ottiene eseguendo una visita di G a partire dal nodo . Nel corso della visita ogni qualvolta un nodo ha più vicini non visitati scegliete sempre quello di indice minimo (in altre parole assumete il grafo rappresentato tramite liste di adiacenza dove i nodi di ciascuna lista sono ordinati per indice crescente).
2. dire quali sono gli archi di attraversamento, quali gli archi in avanti e quali gli archi all'indietro che si incontrano nel corso della visita.

G=



G=





Gli archi in rosso sono archi all'indietro
Gli archi blu sono archi di attraversamento
Non ci sono archi in avanti.