Esercizio 1 (Sequenza lecita massima). *Data una sequenza di interi X, definiamo una sotto-sequenza di X come lecita se essa non contiene elementi di X consecutivi. Definiamo inoltre come valore della sotto-sequenza la somma dei suoi elementi.*
Dare lo pseudocodice di un algoritmo che, data la sequenza X in input, restituisca in $O(n)$ una sotto-sequenza lecita di valore massimo.

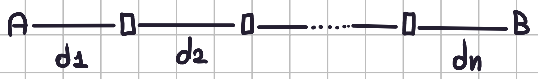
Oss: ogni soluzione non contiene elementi negativi.

```
SeqMax(X: array){
    n=x.length()
    L1:list
    L2:list
    v1=0
    v2=0
    for(i=0..., n-1){
        if(X[i]>0){
            if(i%2==0){
                L1.add(A[i])
                v1+=A[i]
            }else{
                L2.add(A[i])
                v2+=A[i]
            }
        }
    }
    if(v1>v2){return L1}
    return L2
}
```

Esercizio 2 (Minimo numero di rifornimenti). *Vogliamo effettuare un viaggio da una località A ad una località B con un'automobile avente un'autonomia massima di t kilometri. Lungo il percorso, a partire dalla località A sono presenti n distributori di benzina dove è possibile effettuare rifornimento completo del carburante, dunque ripristinando a pieno l'autonomia della macchina.*

Siano $d_1, \dots, d_n$ le distanze dei vari punti di sosta del tragitto, dove per $1 \leq i \leq n-1$ il valore d_i rappresenta la distanza dal distributore i al distributore $i+1$, mentre d_n rappresenta la distanza dal distributore n alla località B.

Assumendo che $d_1, \dots, d_n \leq t$ e che alla località A il serbatoio sia vuoto (ricordiamo che il primo distributore è situato sulla località A), progettare un algoritmo di complessità $O(n)$ che restituisca un'insieme minimo di distributori in cui è necessario effettuare la sosta per completare il viaggio. Dimostrare la correttezza della soluzione proposta.



Oss: il primo distributore e' nella soluzione.

Quando si fa rifornimento al K-esimo distributore, si potra' continuare per i prossimi l distributori senza benzina se $\sum_{i=K}^{l} d_i \leq t \wedge \sum_{i=K}^{l+1} d_i > t$.

```
Questa_e'_Benzina(t:reale, {d1, d2, ... dn}){
    Fuel = t
    dper = 0
    Sol = {1}
    for (i = 1 ..., n-1){
        Fuel -= di
        if(d(i+1) > Fuel){
            Fuel = t
            Sol.add(i)
        }
    }
    return Sol
}
```

Esercizio 3 (Asta da vendere). *Siano $p_1, \dots, p_k$ degli interi positivi, dove p_i rappresenta il prezzo di vendita di un'asta lunga i.*

Marco dispone di un'asta lunga n e vuole rivenderla intera o suddivisa in pezzi in modo da massimizzare il ricavo.

Questo problema e' correlato ai numeri di Fibonacci.

Conviene tagliare a K se $p_K > \sum_{i=1}^{K-1} p_i$

$P(K)$ = profitto max di un asta lunga K

$P(K+1) = \max(p_{K+1}, P(K)+p_1)$ $\quad P(1) = p_1 \quad P(2) = \max(p_2, p_1+p_1)$

```
PMax(K:int, P={p1,p2...,pn}){
    iF(K==1){return p1}
    m=0
    For(i=1...,K-1){
        m=max(pK,PMax(K-i,P)+pi)
    }
    return m
}
```

Esercizio 4 (Quote in borsa). *Le quotazioni in borsa di un'azienda nell'arco di n giorni sono immagazzinate in un array A di interi. Supponiamo che le quotazioni tra il primo e l'ultimo giorno siano salite, dunque che $A[1] < A[n]$.*

1. *Dimostrare che esiste almeno un indice $1 \leq i \leq n$ tale che $A[i] < A[i+1]$.*
2. *Progettare un algoritmo di complessità $O(\log n)$ che preso in input l'array A tale che $A[1] < A[n]$ restituisca un indice tale che $A[i] < A[i+1]$*

1: Supponiamo per assurdo che $\nexists i \mid A[i] < A[i+1]$, allora $A[n] \leq A[n-1] \leq A[n-2] \dots \leq A[1] \Rightarrow A[n] \leq A[1]$ si ha una contraddizione. ■

E' ovvio che $\sum_{i=1}^{n-1} A[i+1]-A[i] = A[n]-A[1] \Rightarrow$ se $A[n] > A[1]$ c'e' ALMENO un incremento positivo.

Raga la somma degli incrementi interni e' uguala all'incremento totale ! Teorema di Stokes Discreto

sono ridicolo

2:

```
In
    i=n/2
    prec=n
    while(true){
        iF(A[i]<A[i+1]){return i}
        iF(A[0]<A[i]){
            prec=i
            i=i/2
        }else if(A[i]<A[prec]){
            i=i+i/2
        }
    }
}
```