Esercizio 1 Si consideri il seguente problema. Dato un intero k ed una matrice di interi M di dimensione $n \times n$ in cui gli elementi in ogni riga sono ordinati in modo crescente ed anche gli elementi in ogni colonna sono ordinati in modo crescente, si vuole determinare se l'intero k compare tra gli elementi della matrice.

1. *(max 15 punti)* Descrivere in pesudo–codice un algoritmo che risolve il problema in tempo $O(n)$. (Suggerimento: sullo stile della ricerca binaria individuare nella matrice un elemento che, qualora risulti diverso dall'intero k, permetta, nella ricerca successiva, di cancellare una riga o una colonna della matrice)

2. *(max 3 punti)* Giustificare la complessità dell'algoritmo.

Soluzione Esercizio 1

1. L'algoritmo RICERCA comincia la ricerca di k a partire dalla posizione in alto a destra $M[1,n]$. Esclude dalla ricerca successiva la colonna n, se $M[1,n] > k$, oppure la riga 1, se invece $M[1,n] < k$. La ricerca viene poi riapplicata ricorsivamente alla matrice M privata dell'ultima colonna (vale a dire ad una sottomatrice rettangolare $n \times (n-1)$) o alla matrice M privata della prima riga (vale a dire ad una sottomatrice rettangolare $(n-1) \times n$). La chiamata iniziale è RICERCA$(M, 1, n, k, n)$.

```
RICERCA:  INPUT la matrice M e gli interi i, j, k ed n
    IF i > n OR j < 1 THEN STAMPA l'intero non è presente
    ELSE
        CASE
            M[i,j] < k:    RICERCA(M, i+1, j, k, n)
            M[i,j] = k:    STAMPA l'intero è presente
            M[i,j] > k:    RICERCA(M, i, j-1, k, n)
        ENDCASE
    ENDIF
END
```

2. Ad ogni chiamata si esclude o una riga o una colonna di M. Poiché M ha n righe ed n colonne, dopo al più $O(n)$ chiamate la procedura termina. Ogni chiamata richiede tempo $O(1)$. La complessità dell'algoritmo è quindi $O(n)$.

Esercizio 2

1. *(max 9 punti)* Scrivere in pseudo–codice una procedura che preso in input un intero positivo n, utilizzando le sole cifre $\{0, 1, 2\}$, stampi tutte le sequenze di lunghezza n la somma dei cui elementi dia n. Ad esempio se $n = 4$ allora la procedura deve stampare (non necessariamente in quest'ordine): 0022, 0112, 0121, 0202, 0211, 0220, 1012, 1021, 1102, 1111, 1120, 1201, 1210, 2002, 2011, 2020, 2101, 2110, 2200. La complessità della procedura **deve essere** $O(nD(n))$, dove $D(n)$ è il numero di sequenze da stampare.

2. *(max 3 punti)* Spiegare perché la procedura proposta ha la complessità richiesta.

Soluzione Esercizio 2

1. Un algoritmo che risolve il problema e che si basa sulla tecnica del backtracking è il seguente. L'algoritmo ABC, essendo ricorsivo, prende in input l'intero n, il vettore SOL in cui viene memorizzata la sequenza da stampare, il numero i di elementi inseriti nella sequenza fino a questo momento, il numero tot rappresentante la somma delle cifre che compaiono nella sequenza fino ad ora costruita. Quindi la prima chiamata iniziale sarà ABC(n, SOL, 0, 0).

```
ABC:  INPUT l'intero n, il vettore SOL e gli interi i e tot
    IF i = n THEN stampa la sequenza SOL[1], SOL[2], ..., SOL[n]
    ELSE
        IF tot + 2(n − i − 1) ≥ n THEN
            SOL[i + 1] ← 0
            ABC(n, SOL, i + 1, tot)
        ENDIF
        IF (tot < n) AND (tot + 2(n − i − 1) + 1 ≥ n) THEN
            SOL[i + 1] ← 1
            ABC(n, SOL, i + 1, tot + 1)
        ENDIF
        IF (tot < n) AND (tot + 2(n − i − 1) + 2 ≥ n)THEN
            SOL[i + 1] ← 2
            ABC(n, SOL, i + 1, tot + 2)
        ENDIF
    ENDIF
```

2. Onde evitare la generazione di sequenze inutili, si procede come segue: la cifra $x \in \{0, 1, 2\}$ viene accodata alla soluzione solo se non porta il totale a superare il valore n e fa si che con le cifre ancora da assegnare sia possibile poi ottenere n.

   Per quanto detto una chiamata ricorsiva sarà effettuata se e solo se la soluzione parziale fino a quel momento costruita può estendersi ad una sequenza da stampare. In questo modo la complessità risultante sarà $O(nD(n))$ dove $D(n)$ è il numero di sequenze da stampare.