Cognome: **Nome**: # **Matricola**: **Riga**: **Col**:

Algoritmi e Strutture Dati – 31/08/2021 – Parte A

Esercizio -1 Iscriversi allo scritto entro la scadenza. In caso di inadempienza, -1 al voto finale.

Esercizio 0 Scrivere correttamente nome, cognome, numero di matricola, riga e colonna su tutti i fogli consegnati. Consegnare foglio A4 e foglio protocollo di bella. In caso di inadempienza, -1 al voto finale.

A1 – Complessità – Punti ≥ 8

Trovare i limiti superiore e inferiore più stretti possibili per la seguente equazione di ricorrenza:

$$T(n) = \begin{cases} 1 & n < 6 \\ 3T(\lfloor n/2 \rfloor) + 8T(\lfloor n/6 \rfloor) + n^2 & n \geq 6 \end{cases}$$

A2 – ContainsZero – Punti ≥ 10

Scrivere un algoritmo

boolean containsZero(**int**[ ] A, **int** n)

che prenda in input un vettore A ordinato contenente n numeri interi e restituisca **true** se A contiene un sottovettore (contiguo) non vuoto la cui somma è pari a zero.

Ovvero, se esistono due indici i, j, $1 \leq i \leq j \leq n$ tali che $\sum_{k=i}^{j} A[k] = 0$.

Discutere informalmente la correttezza dell'algoritmo e la sua complessità computazionale. Esistono soluzioni $O(n^3)$ (punteggio 30%), soluzioni $O(n^2)$ (punteggio 60%), soluzioni $O(n \log n)$ (punteggio 80%) e soluzioni $O(n)$ (punteggio 100%).

Per esempio,

- `A = [-5,-4,-1, 2, 3, 4, 7]` contiene `[-4,-1, 2, 3]` la cui somma è zero;
- `A = [-2, 0, 1]` contiene [0] la cui somma è zero;
- `A = [-6,-3, 2, 3, 7]` non contiene sottovettori (contigui) non vuoti la cui somma sia zero.

A3 – Radice quadrata – Punti ≥ 10

Supponete di non avere a disposizione una funzione per il calcolo della radice quadrata. Scrivere un algoritmo

boolean isSquare(**int** n)

che prenda in input un valore intero positivo n e restituisca **true** se è un quadrato perfetto, ovvero se esiste un intero k tale che $n = k^2$, **false** altrimenti.

Discutere informalmente la correttezza dell'algoritmo e la sua complessità computazionale utilizzando il criterio di costo uniforme. Soluzioni di complessità $O(n)$ non verranno prese in considerazione.

Cognome: **Nome**: **# Matricola**: **Riga**: **Col**:

Algoritmi e Strutture Dati – 31/08/2021 – Parte B

B1 – PrintBitsK – Punti ≥ 8

Scrivere un algoritmo

$$\mathsf{printBits}(\textbf{int}\ n,\ \textbf{int}\ k)$$

che dati due interi n e k, stampi tutte le stringhe binarie di n bit che contengono al più k bit 1 consecutivi.

Discutere informalmente la correttezza dell'algoritmo e la sua complessità computazionale.

Ad esempio, per $n = 5$ e $k = 2$ le stringhe da stampare sono le seguenti:

```
00000 10000 01000 11000 00100 10100 01100 00010
10010 01010 11010 00110 10110 00001 10001 01001
11001 00101 10101 01101 00011 10011 01011 11011
```

B2 – Sequenza k-limitata – Punti ≥ 10

Una sequenza *k-limitata* è una sequenza di valori tali per cui due valori consecutivi differiscono al più di k. Ad esempio, $[1, 3, 2, 4]$ è 2-limitata, ma non è 1-limitata (perché $|1 - 3| = 2 > 1$, $|2 - 4| = 2 > 1$).

Data una sequenza A, una sottosequenza k-limitata di A è una sottosequenza di A che è anche k-limitata. Per esempio, in $A = [\mathbf{4}, 6, \mathbf{3}, 1, \mathbf{4}]$ è presente la sottosequenza $[4, 3, 4]$, che è 1-limitata.

Scrivere un algoritmo

$$\mathsf{printKSequence}(\textbf{int}[\,]\ A,\ \textbf{int}\ n,\ \textbf{int}\ k)$$

che prenda in input un vettore A contenente n interi e un valore k, e stampi tutte le sequenze k-limitate contenute in A.

Discutere informalmente la correttezza dell'algoritmo e la sua complessità computazionale.

Per esempio,con $A = [4, 6, 3, 1, 4]$ e $k = 1$, le sottosequenze 1-limitate da stampare sono: $[4, 3, 4]$, $[4, 3]$, $[4, 4]$, $[4]$, $[1]$, $[3, 4]$, $[3]$, $[6]$, $[4]$, $[]$.

Discutere informalmente la correttezza dell'algoritmo e la sua complessità computazionale.

B3 – Somma Palindroma – Punti ≥ 12

Dato un intero positivo n, una *somma palindroma* di n è una sequenza palindroma di k valori interi positivi $a_1, \ldots, a_k$ tale che $\sum_{i=1}^{k} a_i = n$ e $a_1 = a_k$, $a_2 = a_{k-1}$, ... ovvero $a_i = a_{k+1-i}$. Una somma palindroma n ha *lunghezza pari* se k è pari (e di conseguenza, n deve essere pari).

Ad esempio, il numero 6 può essere scritto come somma palindroma con lunghezza pari esattamente in 4 modi diversi:

$$(1{+}1{+}1{+}1{+}1{+}1), (1{+}2{+}2{+}1), (2{+}1{+}1{+}2), (3{+}3)$$

Scrivere un algoritmo

$$\textbf{int}\ \mathsf{countSumEven}(\textbf{int}\ n)$$

che prenda in input un valore n *pari* e conti tutte le somme palindrome di n con lunghezza pari.

Discutere informalmente la correttezza dell'algoritmo e la sua complessità computazionale.