1

(1/1 punti)

Sia n un intero positivo, qual è il valore di s al termine del seguente codice Python?

```
a = list(range(n))
b = a[::-1]
s = 0
for x in zip(a, b):
    s += x[0]+x[1]
```

- $n\,(n-1)$ ✓
- n^2
- $2n^3$
- $\frac{n(n-1)}{2}$

2

(1/1 punti)

a e b sono due liste concatenate contenenti interi; `d0`, `d1`e `d2` tre dizionari, inizialmente vuoti, implementati con liste di trabocco. Gli elementi dei dizionari sono coppie `(k, v)` dove la chiave k è di tipo intero e la chiave v è di tipo puntatore. Vengono eseguite le seguenti operazioni:

- per ogni elemento x di a, la coppia (x, `NULL`) viene inserita in `d0`;
- per ogni elemento x di b, la coppia (x, `NULL`) viene inserita in `d1`;
- per ogni chiave k in `d0` se questa non è in `d1`, la coppia (x, `NULL`) viene inserita in `d2`;

Se a contienene `n0` elementi e b ne contiene `n1`, qual è il costo computazionale (caso medio) per creare `d2`?

- (●) Lineare in max(n0, n1) ✓
- ( ) Quadratico in (n0+n1)
- ( ) Lineare nella dimensione di d2
- ( ) Lineare in n0

3

(0/1 punti)

Si consideri il problema di ordinare una sequenza contenente $2n + 3$ interi maggiori di $4n + 4$ e minori di $6n + 8$. Sia A un algoritmo efficiente che risolve il problema, qual è l'ordine di grandezza del suo costo computazionale?

- n ✓
- $n \log(n)$
- n^2
- Costante

4

(0/1 punti)

Sia n un intero maggiore di 1000 e $n0$ e $n1$ interi positivi minori di n, cosa viene stampato dal seguente codice Python?

```
k = ''
d0, d1 = {}, {}
for i in range(n):
    d0[k+'x'] = i
    d1[i] = k+'x'
    k += 'x'
print(d0[d1[n0]+d1[n1]])
```

- $n0 + n1 + 1$ ✓
- $n0 + n1$
- $n0 + n1 + 2$
- $n0 + n1 - 1$

5

(0/1 punti)

a è una lista concatenata implementata in C contenente almeno 1000 nodi. I nodi della lista sono definiti dalla seguente **struct**:

```
struct nodo {
    int valore;
    struct nodo *succ;
    struct nodo *prec;
};
typedef struct nodo nodo;
```

dove `succ` e `prec` sono, rispettivamente, gli indirizzi del nodo che segue e del nodo che precede quello in questione. Il campo `succ` dell'ultimo nodo ed il campo `prec` del primo nodo valgono `NULL`.

Si consideri la seguente funzione

```
nodo *ListaCross(nodo *x, nodo *y){
    while (x != y){
        x = x->succ;
        y = y->prec;
    }
    return y;
}
```

Sia b il puntatore all'ultimo nodo della lista a, cosa restituisce

`ListaCross(a, b)`

?

- ○ Qualche volta NULL ✓
- ○ Sempre NULL
- ◉ Sempre un puntatore non NULL
- ○ Qualche volta un errore in run-time

6

(0/1 punti)

Sia a una lista Python di $n > 1000$ elementi e k un intero positivo minore di $\sqrt{n}$, qual è la lunghezza di

a[k:-k]

- $n - 2k$ ✓
- $2k$
- k
- $2\,(n - k)$

7

(0/1 punti)

Qual è il costo computazionale della seguente funzione C in funzione della lughezza della stringa di input?

```c
#include <string.h>

void f(char *x){
    int i = 0;
    while ( i < strlen(x)){
        x[i] = '0';
        i++;
    }
}
```

- Quadratico ✓
- Lineare
- Costante
- Cubico

8

(1/1 punti)

Si consideri il problema di ordinare in base al tipo una lista Python contenente n elementi di k tipi diversi. Sia A un funzione efficiente che risolve il problema. Assumendo k molto più piccolo di n, qual è l'ordine di grandezza del costo di A?

- n ✓
- n^2
- $n \log(n)$
- Costante
- k

9

(1/1 punti)

Si consideri la seguente funzione nel linguaggio C:

```c
#include <stdlib.h>

int *f(int n){
    int i, *a;
    if (n < 35)
        n = 35;
    a = malloc(n);
    for (i = 0; i < n; i++){
        a[i] = i;
    }
    return a;
}
```

Quale tra le seguenti affermazioni è vera?

- (●) L'esecuzione di f può generare errori in run-time ✓
- ( ) L'esecuzione di f genera errori in runtime
- ( ) L'esecuzione di f non genera errori in runtime
- ( ) L'esecuzione di f può generare un loop infinito

10

(0/1 punti)

La lista a contiene $n > 1000$ elementi di cui $k < n$ sono interi. Gli interi in a sono tutti diversi e tra questi c'è 0. Che posizione occupa 0 in a dopo l'esecuzione del seguente codice Python?

```
def f(x):
    if type(x) == type(0):
        return x
    else:
        return -1

sorted(a, key=f)
```

- La posizione iniziale ✓
- $n - k$
- $n - k + 1$
- 0