

DOMANDE SCRITTO

NOTA INIZIALE: In generale, le classi di funzioni in termini di complessità temporale si ordinano come segue: $O(1) < O(\log n) < O(n) < O(n \log n) < O(n^c) < O(c^n)$.

1. Dopo aver effettuato i primi $n-1$ confronti dell'algoritmo insertion sort...

- a) Il minimo si trova già al suo posto
- b) il massimo si trova già al suo posto
- c) l'array risulta ordinato
- d) nessuna delle precedenti

2. Qual è l'ordine di grandezza (notazione O-grande) della funzione G ? $T_1 = 5n^2 + 2n - 10$, $T_2 = 5\sqrt{n} + 22$, $G = T_1 + T_2$.

- a) $5(n^2 + \sqrt{n}) + 2n + 12$
- b) $n^2\sqrt{n}$
- c) $n^2 + c$
- d) \sqrt{n}

3. Come è possibile migliorare l'efficienza dell'insertion sort?

- a) Scegliendo opportunamente il pivot
- b) Sfruttando la ricerca binaria
- c) Mediante una selezione random dell'elemento iniziale
- d) Nessuna delle precedenti

4. Qual è la complessità del seguente algoritmo?

```
for(int i = n; i > 0; i/= c){  
    cout << "Hello" << endl;  
}
```

Risposta: $O(\log n)$

5. Qual è l'ordine di grandezza (notazione O-grande) della funzione G ? $T_1 = 5\sqrt{n}^3$, $T_2 = 5n\sqrt{n}^5$. $G = T_1 * T_2$

- a) $n^{2.5}$
- b) $n\sqrt{n}$
- c) n^5
- d) $n^3\sqrt{n}$

6. Qual è l'ordine di grandezza (notazione O-grande) della funzione G? $T_1 = n^c$, $T_2 = n^k$. $G = \log(T_1/T_2)$

- a) $n^{(c-k)}$
- b) $\log n$**
- c) $\log(n^k) - \log(n^c)$
- d) $n^{(k-c)}$

7. Qual è la relazione tra T_1 e T_2 ? (solo n è variabile). $T_1 = O(n\log n\log n)$, $T_2 = O(1.5n\log n)$

- a) $T_1 > T_2$**
- b) $T_1 < T_2$
- c) $T_1 = T_2$
- d) non è possibile determinarlo

8. Diciamo che $f(n) = O(g(n))$ se e solo se esistono c , $n_0 > 0$ tali che:

Risposta: $0 \leq f(n) \leq cg(n)$ per ogni $n > n_0$

9. Qual è l'output del seguente algoritmo?

```
int main()
{
    char str1[] = {'C', 'i', 'a', 'o'};
    char str2[] = "Ciao";

    cout << sizeof(str1) << endl;
    cout << sizeof(str2) << endl;
}
```

Output: 4, 5

10. Quale istruzione serve inserire al posto di **???? per stampare il contenuto della matrice mt?**

```
int main()
{
    double mt[2][3] = {{1, 2, 3}, {4, 5, 6}};
```

```
for(int f = 0; f < 2; f++){
```

```
    for(int c = 0; c < 3; c++){
```

```
        cout << ?????? << " ";
```

```
    }
```

```
    cout << "\n";
```

```
}
```

a) mt(f+c)

b) *(mt+f)[c]

c) mt[f] + c

d) *(*(mt + f) +c)

11. Quale sarà l'output?

```
int boo(int &n){
```

```
    int a = 0;
```

```
    for(; n>0; n--)
```

```
        a +=n;
```

```
    return a;
```

```
}
```

```
int main()
```

```
{
```

```
    int n = 3;
```

```
    int m = boo(n);
```

```
    cout << n*m;
```

```
    return 0;
```

```
}
```

Output: 0

12. Una delle seguenti informazioni è vera per una funzione inline?

a) Viene eseguita più rapidamente poiché trattata come una macro internamente.

b) Viene eseguita più rapidamente perché gli viene attribuita una priorità più alta rispetto una funzione normale.

c) non viene eseguita più velocemente delle altre funzioni

d) nessuna delle precedenti.

13. Qual è l'output della seguente funzione?

```
#include <iostream>

using namespace std;

template <typename T>

void func(const T& x){

    static int count = 0;

    cout << "x = " << x << " count = " << count << endl;

    ++count;

    return;

}

int main(){

    func<int>(1);

    cout << endl;

    func<int>(1);

    cout << endl;

    func<double>(1.1);

    cout << endl;

    return 0;

}
```

RISPOSTA:

x = 1 count = 0

x = 1 count = 1

x = 1.1 count = 0

DOMANDE SULLA COMPLESSITA'

14. $T_1 = 5n^2 + 2n - 10 = T_2$ $5\sqrt{n} + 22 \Rightarrow G = T_1 + T_2$:

-già fatta ($n^2 + c$)

15. $T_1 = 5\sqrt{n}^3$ $T_2 = 5n\sqrt{n}^5 \Rightarrow G = T_1 * T_2$:

-già fatta (n^5)

16. $T_1 = O(n \log^2 n)$ $T_2 = O(1,5n \log n)$:

- già fatta ($T_1 > T_2$).

17. $T_1 = O((5^n)n \log n)$, $T_2 = O((4^n)n)$:

- $T_1 > T_2$

18. $T_1 = 5n^2 + 2n - 10$ $T_2 = 5\sqrt{n} + 22$ $T_1 * T_2$

- $n^{2.5}$

19. Quale delle seguenti relazioni contiene almeno un errore? (solo n è variabile, $c > 1$):

a) $O(c) < O(\log n) < O((\log^c)n)$

b) $O(\log n) < O(n \log n) < O((\log^c)n)$

c) $O(n) < O(\log n) < O(n^c)$

d) $O(\log n) < O((\log^c)n) < O(n^2)$

20. [47,3,32,21,56,92] dopo due confronti -> [3,21,47,32,56,92], qual è l'algoritmo di ordinamento utilizzato?

a) Insertion Sort

b) Selection Sort

c) Quick Sort

d) Nessuna

21. Qual è l'istruzione errata?

```
int x, y;
```

```
int *const p = &x;
```

```
const int *q = &y;
```

```
q = &x;
```

```
*q = 4;
```

22. Qual è una delle conseguenze dell'aggiunta di un nodo sentinella in una lista doppiamente linkata?

- a) la complessità dell'inserimento diventa $O(\log n)$
- b) il codice è più leggibile
- c) la lista diventa circolare
- d) la lista diventa una coda

23. Ogni algoritmo di ordinamento basato su confronti richiede almeno:

- a) $n \log n$ passi
- b) n^*n passi
- c) $\log n$ passi
- d) nessuna delle precedenti

24. scegliere l'uso corretto di delete per l'espressione `ptr = new int[100]`:

- a) `delete ptr`
- b) `delete ptr[]`
- c) `delete[] ptr`
- d) `[] delete ptr`

25. Qual è l'output del seguente programma?

```
#include <iostream>
```

```
using namespace std;
```

```
class abc{  
    void f();  
    void g();  
    int n;  
};
```

```
int main(){
```

```
cout << sizeof(abc) << endl;  
}
```

Risposta: 4 (ovvero 4 byte, dim dell'int perché i metodi non occupano spazio nell'istanza di un oggetto).

26. Considerando la seguente semplice implementazione di una lista concatenata cosa effettua la funzione boo()?

```
class Nodo{  
public:  
    Nodo* succ;  
    int valore;  
};
```

```
class ListaConcatenata{  
public:  
    Nodo* testa;  
};
```

```
void boo(Nodo* testa){  
    if(testa == nullptr) return;  
    cout << "testa->valore" << endl;  
    boo(testa->succ);  
}
```

- a) Se 'testa' è un puntatore nullo non fa niente, altrimenti stampa il valore che contiene
- b) **Se 'testa' corrisponde alla testa di una lista concatenata, stampa tutto il contenuto della lista dalla testa alla coda**
- c) Se 'testa' corrisponde alla testa di una lista concatenata, stampa tutto il contenuto della lista in ordine inverso.
- d) Il codice della funzione boo() contiene un errore e non può essere compilato.

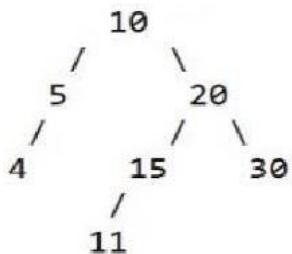
27.L'operatore usato per accedere ai membri di un ADT usando un oggetto è:

- a) **.**
- b) **->**

c) *

d) nessuna delle precedenti.

27. Indicare quale sequenza di chiavi genera il seguente BST:



a) 10 20 5 11 15 4 30

b) 10 5 20 30 11 15 4

c) 10 5 20 4 30 15 11

d) nessuna delle precedenti

28. Qual è l'output del seguente frammento di codice?

```
Nodo BST<T> *ptr = root;  
while(ptr->getParent()){  
    cout << ptr->getKey() << endl;  
}  
while(ptr){  
    cout << ptr->getKey() << endl;  
    ptr = ptr->getRight();  
}
```

- a) Stampa ricorsivamente tutti i figli destri a partire dalla radice
- b) Stampa ricorsivamente la chiave del parent fino alla radice, poi stampa tutte le chiavi dei figli destri
- c) Questo frammento non può essere compilato
- d) Stampa le chiavi della root all'infinito

29. L'operazione di enqueue nell'implementazione con una lista linkata con puntatori alla testa e alla coda avviene in tempo:

- a) Logaritmico – $O(\log n)$
- b) Lineare – $O(n)$
- c) Quadratico – $O(n^2)$
- d) Costante – $O(1)$

30. L'operazione di enqueue nell'implementazione con una lista linkata con il solo puntatore alla testa avviene in tempo:

- a) Logaritmico – $O(\log n)$
- b) Lineare – $O(n)$
- c) Quadratico – $O(n^2)$
- d) Costante – $O(1)$

31. Quali di questi algoritmi di ordinamento sono basati sul paradigma divide et impera?

-Quicksort e mergesort

32. Nel caso peggiore, qual è il numero di confronti necessari per la ricerca di un elemento all'interno di una lista concatenata semplice?

- a) $\log n$
- b) $n/2$
- c) $\log(n-1)$
- d) n

<pre>void func(int n) { int count = 0; for (int i=n/2; i<=n; i++) for (int j=1; j<=n; j = 2 * j) for (int k=1; k<=n; k = k * 2) count++; }</pre>	$O(n^2)$	$O(n \log^2 n)$	$O(n \log n)$	$O(\log n)$
<pre>void func(int n) { int count = 0; for (int i=n/2; i<=n; i++) for (int j=1; j+n/2<=n; j = j++) for (int k=1; k<=n; k = k * 2) count++; }</pre>	$O(n^2 \log n)$	$O(n \log^2 n)$	$O(n \log n)$	$O(\log n)$
<pre>void func(int n) { int count = 0; for (int i=0; i<n; i++) for (int j=i; j<i+1; j++) if (j%6 == 0) for (int k=0; k<j; k++) cout << "#"; }</pre>	$O(n^5)$	$O(n^4)$	$O(n^3)$	$O(n^2)$

33. $O(n \log^2 n) \rightarrow B$

34. $O(n^2 \log n) \rightarrow A$

35. $O(n^3) \rightarrow C$ (da attenzionare che forse è $O(n^5)$)

36.

class A:{

...

friend void foo(...)

}

- a) La funzione foo può accedere direttamente a tutti i membri di A pur non essendone membro.
- b) La funzione foo è un membro privato di A.
- c) La funzione foo può essere richiamata con la notazione A::foo
- d) La funzione foo può accedere solo ai membri public di A.

37. Dalla seguente classe, scegliere la definizione corretta per la funzione membro f

Template <class T>

Class abc{

 void f();

}

- a) Template <class T> void abc<T>::f() {}
- b) Template <class T> void abc::f() {}
- c) Template <T> void abc<class T>::f() {}
- d) Template <T> void abc<T>::f() {}

38. Qual è l'output del seguente programma?

```
#include <iostream>
using namespace std;
class abc{
public:
    static int x;
    int i;
    abc(){
        i= ++x;
    }
};
int abc::x;
int main(){
    abc m, n, p;
    cout << n.x << " " << n.i << endl;
}
```

- a) 3 1
- b) 3 3
- c) 3 2
- d) 1 3

39. si considerino le seguenti operazioni su una pila:

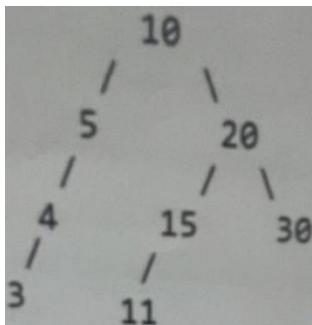
```
push(10)
push(12)
push(8)
push(4)
pop()
pop()
push(3)
```

quanti pop sono necessari per estrarre il numero 12?

- a) 1
- b) 2

- c) 3
- d) 4

40. Considerando il seguente BST indicare quale sarebbe la sequenza dei nodi da visitati da una visita post-order dopo aver eliminato il nodo con chiave 20:



- a) 3 4 5 10 11 15 30
- b) 10 5 4 3 30 15 11
- c) 3 4 5 11 15 30 10
- d) 3 11 4 15 5 30 10

41. Cosa fa la funzione boo()?

```

int boo(Lista<T> &testa, T& x, int i=20){

    Lista<T> q = testa;
    i = 0;

    while(q != NULL){

        if(x==q->getValore()) i++;

        q = q->succ;
    }

    return i;
}
  
```

- a) Conta il numero di volte che il valore 20 è presente nella lista puntata da "testa".
- b) Conta il numero di elementi presenti nella lista puntata da "testa".
- c) **Conta il numero di elementi uguali ad 'x' presenti nella lista puntata da "testa".**
- d) Scorre tutta la lista puntata da "testa" stampando le sue chiavi.

42. Qual è la relazione tra T1 e T2? (solo n è variabile). T1 = O(9nlog^3n), T2 = O((3^n)*n+(4^n)*nlogn:

- a) T1 > T2
- b) **T1 < T2**
- c) T1 = T2

d) non è possibile determinarlo.

43. Considerando la seguente sequenza di chiavi inserite in un BST inizialmente vuoto, quali sono i due figli del nodo con chiave 5?

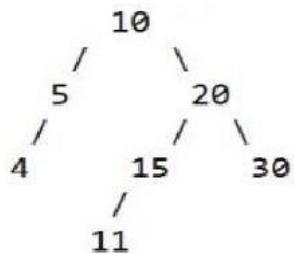
8 5 7 2 3 6 14 16

- a) 2 e 6
- b) 3 e 6
- c) 2 e 7
- d) 3 e 7

44. Il collegamento dinamico è lo strumento utilizzato per la realizzazione...

- a) ...della memoria dinamica.
- b) ...dell'ereditarietà
- c)...del polimorfismo
- d)...della composizione di classi.

45. Considerando il seguente BST. Se effettuassimo l'inserimento di un nuovo nodo con chiave 13, che valore avrebbe la chiave del suo nodo padre? Risposta: 11



46. Come è possibile migliorare l'efficienza del selection sort?

- a) Scegliendo opportunamente il pivot
- b) Sfruttando la ricerca binaria
- c) Mediante una selezione random dell'elemento iniziale
- d) Nessuna delle precedenti

47. Qual è l'output del seguente programma?

```
#include <iostream>

using namespace std;

class abc{

    void f();
    void g();

    long double X;

};

int main(){

    cout << sizeof(abc) << endl;
}
```

a) 16 (ovvero la dimensione in byte del long double)
b) 4
c) 2
d) Errore di compilazione

48. Cosa fa la funzione boo() ?

```
int Lista::boo(){

    Nodo* aus = testa;

    int c = 0, i = 0;

    while(aus){

        c++;

        if(aus->valore == 10){

            i++;

        }

        aus = aus->succ;

    }

    return c;
}

a) Restituisce il numero di volte che il valore 10 è presente nella lista puntata da "testa".  
b) Restituisce il numero di elementi presenti nella lista puntata da testa.  
c) Restituisce zero in ogni caso.  
d) Scorre tutta la lista puntata da "testa" stampando le sue chiavi solo se sono uguali a 10.
```

49. Considerando il seguente codice, quale delle chiamate della funzione max() genererà un errore di compilazione?

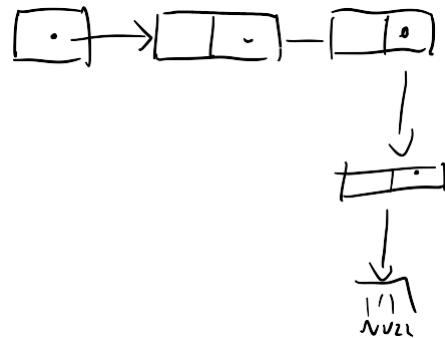
```
#include <iostream>
using namespace std;
template <typename T>
T max(T x, T y){
    return (x > y) ? x : y;
}

int main(){
    cout << max(3, 7) << endl;
    cout << max(3.0, 7.0) << endl;
    cout << max(3, 7.0) << endl;
    cout << max<float>(3, 7.0) << endl;
}
```

a) max (3, 7)
b) max(3.0, 7.0)
c) **max(3, 7.0)**
d) max<float>(3, 7.0)

50. Consideriamo la seguente semplice implementazione di una lista concatenata, cosa effettua la funzione boo() ?

```
class Nodo{  
public:  
    Nodo* succ;  
    int valore;  
};  
  
class ListaConcatenata{  
public:  
    Nodo* testa;  
};  
  
void boo(Nodo* testa){  
    if(testa==nullptr) return;  
    boo(testa->succ);  
    cout<<testa->valore<<endl;  
}
```



1. Se testa è un puntatore nullo non fa niente, altrimenti stampa il valore che contiene
2. Se testa corrisponde alla testa di una lista concatenata, stampa tutto il contenuto della lista dalla testa alla coda.
3. se testa corrisponde alla testa di una lista concatenata, stampa tutto il contenuto della lista in ordine inverso
4. Il codice contiene un errore e non può essere compilato.

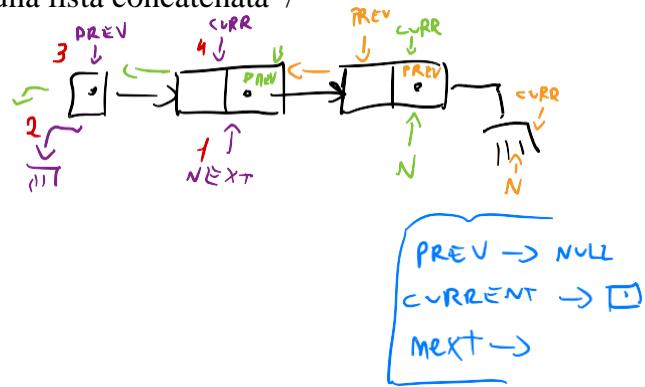
51. L'ultima riga della seguente funzione inverti() ha una istruzione mancante.

Quale di queste istruzioni dovrebbe essere inserita alla fine della

procedura per permettere alla funzione inverti() di invertire l'ordine degli elementi della lista concatenata la cui testa è data in input?

```
/*testa punta alla testa di una lista concatenata*/  
void inverti(Nodo* testa)
```

```
{  
    Nodo* prev = nullptr;  
    Nodo* current = testa;  
    Nodo* next;  
    while(current!=nullptr){  
        1 next = current->succ;  
        2 current->succ=prev;  
        3 prev=current;  
        4 current=next;  
    }  
    //AGGIUNGERE RIGA QUI  
}
```



- 8 1. testa = prev;
- 2. testa = current;
- 3. testa = next;
- 4. testa = nullptr;

52. Supponendo che il parametro p passato alla funzione func() sia la

testa di una lista concatenata, la funzione func() restituisce il valore 1

se e solo se

int func(teste)

Nodo{

public:

 int valore;

 Nodo *succ;

};

int func(Nodo* p){

 return(p==nullptr || p->succ==nullptr || ((p->valore <= p->succ->valore && func(p->succ))));

}

testa = null

Se è nata o se se ha un solo elemento
valore dell' test ≤ valore nodo succ

1. Gli elementi della lista sono tutti diversi
2. Gli elementi della lista sono ordinati in maniera non decrescente
3. Gli elementi della lista sono ordinati in maniera non crescente
4. Nessuna delle precedenti

53. Supponiamo di avere i puntatori al primo e all'ultimo elemento di una lista concatenata semplice.

Quale di queste operazioni ha un costo che dipende dalla lunghezza della lista?

1. Eliminare il primo elemento
2. Inserire un nuovo elemento in testa
3. Eliminare l'ultimo elemento della lista
4. Aggiungere un nuovo elemento alla fine della lista

54. Qual è il numero minimo di campi per ciascun nodo di una lista doppiamente linkata?

2. 2

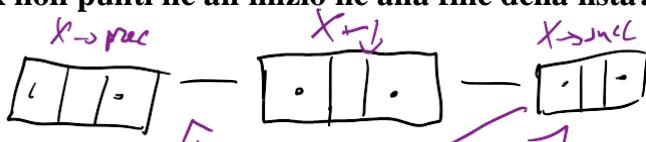
✗ 3. 3

4. 4

□

55. Una lista doppiamente linkata è definita usando la classe nodo seguente, dove prec e succ rappresentano i puntatori agli elementi adiacenti. Quale dei seguenti segmenti di codice deve essere eseguito per eliminare correttamente il nodo puntato da X, assumendo che X non punti né all'inizio né alla fine della lista?

```
Nodo {  
public:  
    int valore;  
    Nodo *succ;  
    Nodo *prec;  
};
```



- ✗ 1. $X \rightarrow \text{prec} \rightarrow \text{succ} = X \rightarrow \text{succ}$; $X \rightarrow \text{succ} \rightarrow \text{prec} = X \rightarrow \text{prec}$;
2. $X \rightarrow \text{prec}. \text{succ} = X \rightarrow \text{succ}$; $X. \text{succ} \rightarrow \text{prec} = X \rightarrow \text{prec}$; **No**
3. $X. \text{prec} \rightarrow \text{succ} = X. \text{prec}$; $X \rightarrow \text{succ}. \text{prec} = X. \text{prec}$; **No**
4. $X \rightarrow \text{prec} \rightarrow \text{succ} = X \rightarrow \text{prec}$; $X \rightarrow \text{succ} \rightarrow \text{prec} = X \rightarrow \text{succ}$;

56. Quale di queste strutture dati implementa una politica LIFO?

1. BST
2. Coda
- ✗ 3. Stack
4. Lista

57. L'implementazione di una pila con un array statico

1. Trasforma la pila in una coda
- X2.** Limita la dimensione massima della pila
3. Consente di effettuare tutte le operazioni in tempo costante
4. Non è possibile

58. Qual è la complessità di questa funzione?

```
func(int n)
{
    if(n==1) return;
    for(int i=1;i<=n;i++)
    {
        for(int j=1;j<=n;j++)
            cout<<"@ @ @";
        break;           C'è il break quindi O(n)
    }
}
```

□

- X1.** $O(N)$
- 2. $O(\sqrt{N})$
- 3. $O(N/2)$

4. O(logN)

59. Qual è la complessità di questa funzione?

```
void func(int n)
{
    int count=0;
    for(int i=n/2;i<=n;i++)
        for(int j=1;j<=n;j=2*j)
            for(int k=1;k<=n;k=k*2)
                count++;
}
```

- 1. O(n^2)
- 2. O($n \log^2 n$)**
- 3. O($n \log n$)
- 4. O($\log n$)

59. Come eredita i membri public e protected della classe A?

```
Class A
{
    Int a;
```

protected:
 Char b;
 ...

public:
 float c;
 ...

```
};
```

*La classe B li eredita
rendendoli entrambi privati.*

Class B : private A

```
{  
...  
};
```

Risposta: La classe B li eredita rendendoli entrambi privati

60. Una classe astratta è una classe che: ... *possiede almeno una funz virtuale pura*

Risposta: deve contenere almeno una funzione virtuale pura.

61. Qual è la complessità di questa funzione?

```
1 int i, j, k = 0;  
2  
3 for(i = n/2; i <= n; i++)  
4     for(j = 2; j <= n; j = j*2) a) O(n)  
5         k = k + n/2; b) O(n*log(n))
```

c) O(n/2)

d) O(log(n)/2)

62. Qual è la complessità di questa funzione?

```
1 int a = 0, i = N;  
2  
3 while(i > 0) a) O(n)  
4 { b) O(  
5     a += i; c) O(n/2)  
6     i /= 2; d) O(log(n))  
7 }
```

X

63. Qual è la complessità di questa funzione?

```
1 func(int n)  
2 {  
3     if(n == 1)  
4         return;  
5  
6     for(int i = 1; i <= n; i++) a. O(n)  
7     {  
8         for(int j = 1; j <= n; j++)  
9         {  
10             cout << " ###";  
11             break;  
12         }  
13     }
```

b. O(N)

c. O(n/2)

d. O(log(n))

64. Qual è la complessità di questa funzione?

- a) $O(n^2)$
b) $O(n * \log^2(n))$
c) $O(n * \log(n))$

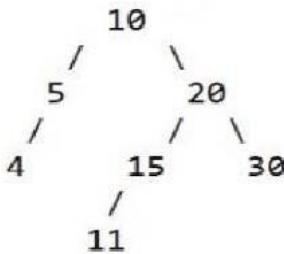
```
1 void func(int n)
2 {
3     int count = 0;
4
5     for(int i = n/2; i <= n; i++) O(n)
6         for(int j = 1; j <= n; j = 2*i) (O logn)
7             for(int k = 1; k <= n; k = k*2) O logn
8                 count++;
9 }
```

- d) $O(\log(n))$

65. Qual è l'ordine di grandezza(notazione O-grande) della funzione G? $T1 = 5n^2 + 2n - 10$, $T2 = 5\sqrt{n} + 2 \Rightarrow G = T1 * T2$:

- a) $n^3/2$
b) $n\sqrt{n}$
c) n^2+c
d) \sqrt{n}

66. Considerando la seguente sequenza di chiavi inserite in un BST inizialmente vuoto, Quali saranno i figli del nodo con chiave 10 dopo aver eliminato il nodo con chiave 5?



- La domanda non si riesce a leggere quindi supponiamo che questo sia l'albero. Se eliminassimo il nodo con chiave 5, dato che questo ha un solo figlio(il sinistro di chiave 4), dopo l'eliminazione del nodo 5 il nodo con chiave 10 avrà **4 e 20 come nuovi figli.**

67. Cosa intendiamo quando diciamo che un algoritmo X è asintoticamente più efficiente di un altro algoritmo Y?

- a) X è sempre migliore di Y per piccoli input.
- b) X è sempre migliore di Y per grandi input.
- c) Y è sempre migliore di X per piccoli input.
- d)Y è sempre migliore di X per grandi input.

68.Qual è la complessità asintotica della procedura di visita in profondità in un grafo contenente N nodi ed M vertici ed implementato attraverso le liste di adiacenza?

- Lineare in $M + N$.

69.Quale tra le funzioni è quella asintoticamente più grande?

- a) $8n^3 + 1/2^n$
- b) $(2+n)^3 + \log n + \log^2 n$
- c) 4^n
- d) n^3

70. Quale tra le seguenti funzioni è asintoticamente più piccole?

- a) $3^n + 5n + \log n(n)$
- b) $(10 + n)^3$
- c) 1^n
- d) 2^n

71. Quale fra le funzioni elencate di seguito è quella asintoticamente più grande?

- a) $n^n + 1/2^n$
- b) n^n
- c) $\log n + 2/n$
- d) $n^2 - \log n$

72. Data la funzione ricorsiva che calcola la moltiplicazione NxM mediante somme successive, qual è la sua complessità?

- $O(M)$.

73. Qual è la complessità asintotica nel caso ottimo, della procedura di inserimento in un albero binario di ricerca se eseguita su un albero contenente n chiavi?

- Logaritmica in n

74. Qual è la complessità asintotica della procedura per il calcolo della componente fortemente connessa di un grafo direzionale con V nodi ed E archi, rappresentato mediante matrice di adiacenza?

- Quadratico in V

75. Quale fra le funzioni elencate di seguito è quella asintoticamente più piccola?

- a) $3^4, 5n + \log n(n)$
- b) $3n^4 + 3n^2 + \log n + 4$
- c) $\log n + \log^2 n + 3n^5$
- d) $4n^4 + 2n^2 + \log n + 5$

76. Quale tra i seguenti algoritmi di ordinamento hanno una complessità asintotica lineare nel caso ottimo?

- Bubble sort con sentinella.

77. Qual è il numero minimo di archi da attraversare per effettuare una visita in ampiezza in un grafo con N nodi e M archi?

- M

1) Quale è la relazione fra T_1 e T_2 ?

$$T_1 = O(\log n)$$

$$T_2 = O((3^n) \cdot n) + (4^n) * n \log n$$

A) $T_2 > T_1$ Si vede a occhio che $T_2 > T_1$

B) $T_1 < T_2$ In T_2 vi sono gli esponenti.

C) $T_1 = T_2$

D) Non è possibile determinarlo

2) Quale delle seguenti relazioni contiene almeno un errore?

A) $O(1) < O(\log n) < O(\log^2 n)$

B) $O(\log n) < O(n \log n) < O(n \log^2 n)$

C) $O(1) < O(\log n) < O(n^2) \quad \text{X} \quad O(n) > O(\log n)$

D) $O(\log n) < O(\log^2 n) < O(n^2)$

3) Come è possibile migliorare l'efficienza dell'insertion sort?

A) Seguendo approssimativamente il pivot

B) Sfruttando la ricerca Binaria Migliora l'efficienza

C) Mediante una sezione casuale dell'elemento iniziale

D) Nessuna delle precedenti

4) Quale è l'ordine di grandezza (notazione O-grande) della funzione G ?

$$T_1 = 5n^2 + 2n - 10$$

$$T_2 = 5\sqrt{n} + 2$$

$$G = T_1 * T_2$$

$$5n^2 + 2n \quad 5\sqrt{n}$$

$$5(n^2 \cdot n^{1/2}) + 2n$$

$$5(n^{3/2}) + 2n$$

A) $n^{3/2}$ X

B) $n\sqrt{n}$

C) $n^2 + n$

D) \sqrt{n}

5) Quale è la relazione tra T_1 e T_2 (solo n è una variabile)

$$T_1 = O(n \log^2 n)$$

$$T_2 = O(1.5 n \log n)$$

A) $T_1 > T_2$

B) $T_1 < T_2$

C) $T_1 = T_2$

D) Non è possibile determinarlo

6) Quale è l'ordine di grandezza (notazione O-grande) della funzione G?

$$T_1 = 5n^2 + 2n - 10$$

$$T_2 = 5\sqrt{n} + 22$$

$$G = T_1 + T_2$$

$$(n^2) + 5\sqrt{n}$$

A) $5(n^2 + \sqrt{n}) + 2n + 12$

B) $n^2\sqrt{n}$

C) $n^2 + e$ n^2 pesa di più

D) \sqrt{n}

7) Quale è l'ordine di grandezza (notazione O-grande) della funzione G?

$$T_1 = 5\sqrt{n}^3$$

$$T_2 = 5n\sqrt{n}$$

$$G = T_1 * T_2$$

$$5(\sqrt{n}^3) \cdot 5n(\sqrt{n})$$

A) $n^{2.5}$

B) $n\sqrt{n}$

C) $n^{1.5}$

D) $n^3\sqrt{n}$

$$25n \left(n^{3/2} + n^{3/2} \right)$$

$$25n(n^4) = 25n^5$$

8) Diciamo che $f(n) = O(g(n))$ se e solo se esistono $c, n_0 > 0$:

- A) $f(n) \leq c g(n) \quad \forall n > n_0 \quad \text{X}$
- B) $c g(n) \leq f(n) \quad \forall n > n_0$
- C) $f(n) > c g(n) \quad \forall n > n_0$
- D) $c f(n) > g(n) \quad \forall n > n_0$

Dalla definizione del O (grande) si ha che

$$O \leq f(n) \leq c g(n)$$

9) Come è possibile migliorare la retezza dell' Selection Sort

- A) Selezionando il pivot
- B) Sfruttando la Ricerca Binaria Questo è per INSERTION SORT.
- C) //
- D) Nessuna delle precedenti X Ha Sempre Complessità $O(n^2)$

10) Quale è la complessità fra T_1 e T_2 (Solo n è variabile)

$$T_1 = O((5^e) * n \log n)$$

$$T_2 = O(14^e * n)$$

- A) $T_1 > T_2$ \otimes questa dato che l'esponenziale ha base maggiore
- B) $T_1 < T_2$
- C) $T_1 = T_2$
- D) Non è possibile Determinarla

11) Quale istruzione serve inserire al posto di ??? per stampare il contenuto della matrice?

```
double mat[2][3] = {{1,2,3,4,5,6}};  
for(int f=0; f<2; f++) {  
    cout << ??? << endl;  
    cout << endl;  
}
```

- A) $mat[f+e]$
- B) $*(\mathtt{mat} + f)[e]$
- C) $mat[f]+e$
- D) $(*(\mathtt{mat} + f) + e) \otimes$

12) Una delle seguenti affermazioni è vera per la funzione Inline

A) Vene eseguita più rapidamente poiché trattata come una riga zero ⊗ Vene sostituto il pezzo di codice.

B)

C)

D)

B) Quale è l'output del seguente Codice?

#include <iostream>

using namespace std;

```
Template <typename T>
void fune (const T& x) {
    static int count = 0;
    cout << "x = " << x << "count = " << count << endl;
    # count;
    return;
}
```

```
(# main() {
```

```
    fune<int>(1);
    cout << endl;
```

```
    fune<int>(2);
    cout << endl;
```

```
    fune<double>(1.1);
    cout << endl;
```

```
    return 0;
}
```

A) $x = 1 \quad \text{count} = 0$
 $x = 1 \quad \text{count} = 1$
 $x = 1.1 \quad \text{count} = 0$

⊗ Essendo la terza chiamata effettuata con un tipo
double diverso da quello prima il count ritorna a zero.

14) Quale è il seguente output del programma?

```
#include <iostream>
using namespace std;
class Abe {
public:
    void f()
    void g()
    long double x; // è un long double
};

int main() {
    cout << sizeof(Abe) << endl;
}
```

- A) 16 \neq long double = 16
- B) 4
- C) 2
- D) errore di compilazione

15) Cosa fa la funzione boo()

```
int Lista::boo() {
    Nodo * aux = testa;
    int c = 0, i = 0;
    while (aux)
        if (aux->valore == 10)
            i++;
    aux = aux->succ
}
```

return c; \rightarrow C è il numero di elementi trovati in Lista

(B) Restituisce il numero di elementi nella lista.

16) Non si esegue

17) Considerando il seguente codice, quale delle chiamate della funzione `max()` genera un errore di compilazione

#include <iostream>

using namespace std;

```
template <typename T>
T max (T x, T y) {
    return (x > y) ? x : y;
}
```

```
int main () {
    cout << max (3, 7) << endl;
    cout << max (3.0, 7.0) << endl;
    cout << max (3, 7.0) << endl;
    cout << max <float> (3, 7.0);
    return 0;
}
```

}

A) `max (3, 7);`

B) `max (3.0, 7.0);`

C) `max (3, 7.0);` \otimes due tipi diversi per uno stesso Template.

D) `max <float> (3, 7.0);`

18) Cosa fa la funzione `bool()`,

`int boo(Lista < T > & testa, T & x, int i = 20) {`

`Lista < T > q = testa, i = 0;`

`while (q != NULL) {`

`if (testa.getValore() == x) i++;`

`q = q->succ;`

`}`

`return i;`

`}`

c) Conta il numero di elementi uguali ad 'x' presenti nella lista puntata da `Testa`.

19)

`int x, y;`

`int * const p = &x;`

`const int * p = &y;`

`q = &x;`

`*y = 4;` → genera errore

20) Il collegamento dinamico è lo strumento utilizzato per la realizzazione.

A) della memoria dinamica,

B) dell'ereditarietà,

C) del polimorfismo, 

D) dalle scomposizioni di classi;

21) linea 108 nella funzione inverti()

```
void inverti(Nodo* testa) {  
    Nodo* prev = nullptr;  
    Nodo* current = testa;  
    Nodo* next;  
    while (current != nullptr) {  
        next = current->succ;  
        current->succ = prev;  
        prev = current;  
        current = next;  
    }  
}
```

// Abbiamo QVI

}

A) testa = prev;

B) testa = current;

C) testa = next;

D) testa = nullptr;

22) Nel caso peggiore, qual è il numero di confronti necessari per la ricerca di un elemento all'interno di una lista concatenata semplice?

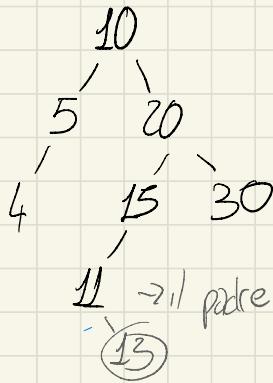
A) $\log n$

B) $\frac{n}{2}$

C) $\log(n-1)$

D) $O(n)$

23) Consideriamo il seguente BST. Se effettassimo l'inserimento di un nuovo nodo con chiave 13, che valore avrebbe la chiave del suo nodo padre?



B) 11 è il padre

24) Considerando la seguente sequenza di chiavi inserite in un BST inizialmente vuoto

Quali saranno i due figli del nodo con chiave 10 dopo aver eliminato il nodo con chiave 5?

NON SI POTESSE
A leggera

25) Quale è l'output del seguente frammento di codice?

```
NodeBST<T> *ptr = root;  
while(ptr->getParent() != NULL){  
    cout << ptr->getKey() << endl;  
    ptr = ptr->getRight();  
}
```

A) Stampa ricorsivamente tutti i figli destri a partire dalla radice. \textcircled{X}

B) Stampa ricorsivamente la chiave del parent fino alla radice, poi stampa tutte le chiavi dei figli destri

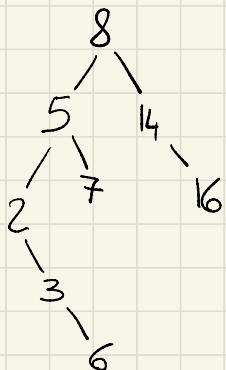
C) Questo frammento non può essere compilato.

D) Stampa la chiave della root all'infinito.

26) Considerando la seguente sequenza di chiavi inserite in un BST inizialmente vuoto.

Quali sono i due figli del nodo con chiave 5?

8 5 7 2 3 6 14 16



A) 2 e 6

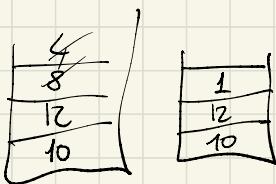
B) 3 e 6

C) 2 e 7 \textcircled{X}

D) 3 e 7

28) Si considerino le seguenti operazioni in pila. Quanti `pop()` sono necessari per estrarre il numero 12?

```
push(10);  
push(12);  
push(8);  
push(4);  
pop();  
pop();  
push(1);
```



B) 2 Θ due pop

29) L'operazione di enqueue nell'implementazione con una lista linkata con il solo puntatore alla testa avviene in tempo

A) Log

B) Lineare $\Theta(n)$

C) Quadratico

D) Costante

30) L'operazione di enqueue nell'implementazione con una lista linkata con puntatori alla testa e alla coda avviene in tempo

D) $\Theta(1)$ Costante

18. Qual è l'output del seguente programma?

```
#include <iostream>
using namespace std;
class abc {
    void f();
    void g();
    long double x;
};

int main() {
    cout << sizeof(abc) << endl;
}
```

- A) 16
- B) 4
- C) 2
- D) errore di compilazione

19. Qual è l'output del seguente codice?

```
int n = 100;
int a[n] = {0};
n = 200;
n += n;
cout << sizeof(a)/sizeof(int) << endl;
```

- A) 4
- B) 0
- C) 100
- D) 200

20. L'ultima riga delle seguenti funzione `Invert()` ha una istruzione mancante. Quale
è questa istruzione? Dovevrebbe essere inserita alla fine della procedura per permettere
che la lista sia data in input?

```
// questa funzione invierte la lista concatenata
void invert(Node* &testa) {
    Node* prev = NULL;
    Node* current = testa;
    Node* next;
    while (current != NULL) {
        next = current->succ;
        current->succ = prev;
        prev = current;
        current = next;
    }
}
```

//AGGIUNGERE UNA RIGA QUI

- A) testa = prev;
- B) testa = current;
- C) testa = next;
- D) testa=NULL;

20. Considerando la seguente sequenza di inserti inseriti in un BST malamente vuoto.

Quali sono i due figli del nodo con chiave 5?

8 5 7 2 3 6 14 16

- A) 2 e 6
- B) 2 e 7
- C) 3 e 7
- D) 3 e 8

30. Nel caso peggiore, quali è il numero di confronti necessari per la ricerca di un
elemento all'interno di una lista completamente esplorata?

- A) $\log n$
- B) 1
- C) $(\log n - 1)$
- D) 0

RISPOS

1 C 2,

11 B 12,

21 C 22,

$(\text{val}) < \text{Val} + 8$

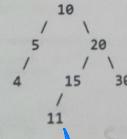
```
int func(Nodo *p) {
    return (p == nullptr) ||
           (p->succ == nullptr) ||
           (p->valore <> p->succ->valore
            && func(p->succ));
}
```

- A) Gli elementi della lista sono tutti diversi
- B) Gli elementi della lista sono ordinati in maniera non decrescita.
- C) Gli elementi della lista sono ordinati in maniera non crescente
- D) Nessuna delle precedenti

26. Quale struttura dati dinamica viene utilizzata dal gestore delle chiamate di funzioni nell'esecuzione di un programma?

- A) BST
- B) Lista concatenata
- C) Stack
- D) Coda

27. Considerando il seguente BST. Se effettuassimo l'inserimento di un nuovo nodo con chiave 13, che valore avrebbe la chiave del suo padre?



- A) 10
- B) 11
- C) 15
- D) 20

```

class Nodo {
    int valore;
    Nodo* succ;
    Nodo* prev;
};

* A)
    X->prec->succ = X->succ;
    X->succ->prec = X->prec;

```

* B)
 X->prec->succ = X->succ;
 X->succ->prec = X->prec; **NO**

* C)
 X->prec->succ = X->prec;
 X->succ->prec = X->prec; **NO**

* D)
 X->prec->succ = X->prec;
 X->succ->prec = X->succ; **NO**

24. L'operazione di enqueue nell'implementazione con una lista linkata con il solo puntatore alla testa avviene in tempo

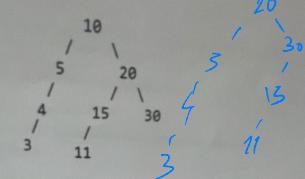
- A) Logaritmico - $O(\log n)$
- B) Lineare - $O(n)$
- C) Quadratico - $O(n^2)$
- D) Costante - $O(1)$

25. Supponendo che il parametro p passato alla funzione func() sia la testa di una lista concatenata, la funzione func() restituisce il valore 1 se e solo se ...

```

class Nodo {
    int valore;
    Nodo* succ;
};

```



- A) 3 4 5 10 11 15 30
- B) 10 5 4 30 15 11
- C) 3 4 5 11 15 30 10
- D) 3 1 4 5 5 30 10

3 - 4 - 5 - 11 - 15 - 30 - 10

22. Quale è l'output del seguente frammento di codice?

```

Nodo*BT1(*ptr = root;
while(!ptr->getLeft()) {
    cout << ptr->getKey() << endl;
}

while(ptr) {
    cout << ptr->getKey() << endl;
    ptr = ptr->getRight();
}

```

- B stampa ricorsivamente tutte i figli destri a partire dalla radice
- C stampa ricorsivamente la chiave dei parent fino alla radice, poi stampa tutte le chiavi dei figli destri
- D) Questo frammento non può essere compilato
- E) Stampa la chiave della root all'infinito

23. Una lista doppiamente linkata è definita usando la classe nodo seguente, dove prec e succ rappresentano i puntatori agli elementi adiacenti. Quale del seguenti segmenti di codice deve essere eseguito per eliminare correttamente il nodo puntato da X, assumendo che X non punti ne all'inizio ne alla fine della lista ?

20. Qual è l'output del seguente codice ?

```
char str1[] = {"C","i","s","o"};
char str2[] = {"Ciao"};

cout << sizeof(str1) << endl;
cout << sizeof(str2) << endl;
```

- A)

4

4



5 S'era /n/

- C)

4

5

- D)

5

4

21. Considerando il seguente BST. Indicare quale sarebbe la sequenza dei nodi visitati da una visita posteriore, dopo aver eliminato il nodo con chiave 20.

Vedi foto

Alta Sinistra
Pallina prima

A) Restituisce il numero di volte che il valore 10 è presente nella lista puntata da "lista".

B) Restituisce il numero di elementi presenti nella lista puntata da "lista".

C) Restituisce zero in ogni caso.

D) Scorre tutta la lista puntata da "test" stampando le sue chiavi solo se sono uguali a 10.

12. Dalla seguente classe, accorgersi la definizione corretta per la funzione membro f:

```
template <class T>
class abc {
    void f();
};
```

X

```
template <class T> void abc::f() {}
```

- B)

```
template <class T> void abc::f() {}
```

- C)

```
template <T> void abc::f() {}
```

- D)

```
template <T> void abc::f() {}
```

13. Una delle seguenti affermazioni è vera per una funzione inline:

X A) Viene eseguita più rapidamente poiché trattata come una macro internamente

B) Viene eseguita più rapidamente perché gli viene attribuita una priorità più alta rispetto

a una funzione normale

C) Non viene eseguita più velocemente delle altre funzioni

D) nessuna delle precedenti

14. Quale riga genera errore di compilazione?

```
int x,y;
int* const p = &x;
const int *q = &y;
q = &x;
*q = 4;
```

E' errore

Esercizi fisi:

Selezionare la corretta definizione di una funzione virtuale pura: `virtual void g() = 0;`

A) `virtual void f() = 0;`

B) `void virtual f() = 0;`

C) `virtual void f() = 0;`

D) Nessuna delle precedenti.

2) L'operazione di enqueue nell'implementazione con una lista finita con puntatori alla testa e alla coda avviene in tempo: $O(1)$ Costante.

A) $O(\log n)$

B) $O(n)$

C) $O(n^2)$

D) $O(1)$

3) Un header definito dall'utente viene incluso usando la sintassi....

A) `#include "file.h"`

B) `#include <file.h>`

C) `# include <file>`

D) `# include file.h`

4) Quali di questi Algoritmi è basato sul paradigma Divide et Impera? Merge - Quick

A) InsertionSort - Selection

B) MergeSort e QuickSort

C) MergeSort e InsertionSort

D) QuickSort e SelectionSort

5) Nel caso peggiore, qual è il numero di confronti necessari per la ricerca di un elemento all'interno di una lista concatenata semplice?

A) $O(\log n)$

B) $O(\frac{n}{2})$

C) $O(\log n-1)$

D) $O(n)$

6) Caso intendiamo dire quando diciamo che un algoritmo X è assolutamente più efficiente di un altro algoritmo Y ?

A) X è sempre migliore di Y per piccoli input.

B) X è sempre migliore di Y per grandi input.

C) Y è sempre migliore di X per piccoli input.

D) Y è sempre migliore di X per grandi input.

7) Una classe Astratta è una classe che:

- A) Deve contenere tutte funzioni virtuali pure.
- B) Deve contenere almeno una funzione virtuale pura. 
- C) Può non contenere funzioni virtuali pure
- D) Deve contenere funzioni virtuali pure definite fuori dalla classe.

8) class A {

...

friend void foo

}

A) La funzione foo può accedere direttamente a tutti i membri di A per non essere membro



B) La funzione foo è un membro privato di A.

C) La funzione foo può essere richiamata con la notazione A.foo

D) La funzione foo può accedere solo ai membri pubblici di A

Q) Class A {

int a;

...

protected:

char b;

...

public:

float c;

...

}

Class B: private A {

...

}

A) La classe B eredita tutti i membri di A.

B) B eredita tutti i membri private di A.

C) La classe B eredita i membri protected e private di A e li rende public.

D) La classe B eredita i membri protected e public di A e li rende public.
Private \otimes

10) L'implementazione di una pila con Array statico:

A) Trasforma la pila in una coda

B) Limita la dimensione Massima della Pila \otimes

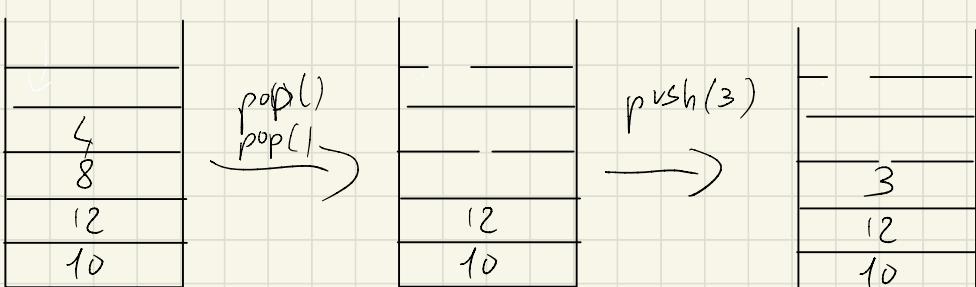
C) Consente di effettuare tutte le operazioni in tempo Costante.

D) Non è possibile.

Si considerino le seguenti operazioni su una pila:

```
push(10)
push(12)
push(8)
push(4)
pop()
pop()
push(3)
```

Quanti pop() sono necessari per estrarre il numero 12?



Per estrarre il 12 Servono 2 pop()

Quesito

Supponendo che il parametro p passato alla funzione func() sia la testa di una lista concatenata, la funzione func() restituisce il valore 1 se ...

```
class Nodo{
public:
    int valore;
    Nodo *succ;
};

int func(Nodo* p){
    return (p == nullptr || p->succ == nullptr ||
           ((p->valore >= p->succ->valore) && func(p->succ)));
}
```

Supponiamo di avere i puntatori al primo e all'ultimo elemento di

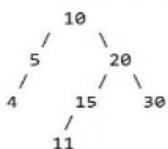
Gli elementi della lista sono in modo NON
Decrescente.

}

Supponiamo di avere i puntatori al primo e all'ultimo elemento di una lista concatenata semplice. Quale di queste operazioni ha un costo che dipende dalla lunghezza della lista?

Eliminazione ultimo elemento
della lista.

Indicare quale sequenza di input delle chiavi genera il seguente BST



- A) 10 20 5 11 15 4 30
- B) 10 5 20 20 11 15 4
- C) 10 5 20 4 30 15 11 X
- D) Nessuna delle precedenti

Una lista devono essere ridotte in definita maniera da chiavi node.

```

void func(int n)
{
    int count = 0;
    for (int i=n/2; i<=n; i++)
        for (int j=1; j<=i; j*=2)
            for (int k=1; k<=n; k = k * 2)
                count++;
}

```

$O(n \log n)$

```

void func(int n)
{
    int count = 0;
    for (int i=n/2; i<=n; i++)
        for (int j=1; j<=i; j++)
            for (int k=1; k<=n; k = k * 2)
                count++;
}

```

$O(n^2 \log n)$

```

void func(int n)
{
    int count = 0;
    for (int i=0; i<n; i++)
        for (int j=0; j<i; j++)
            if (j & 1 == 0)
                {
                    for (int k=0; k<j; k++)
                        count <<= 1;
                }
}

```

(?) $O(n^5)$ perché?

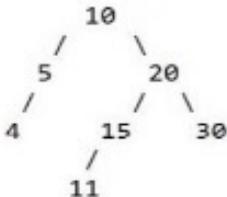
```

((p->valore >= p->succ->valore
  && func(p->succ))
);

```

Supponiamo di avere i puntatori al primo e all'ultimo elemento di una lista concatenata semplice. Quale di queste operazioni ha un costo che dipende dalla lunghezza della lista?

Indicare quale sequenza di input delle chiavi genera il seguente BST



Una lista doppiamente linkata è definita usando la classe nodo seguente, dove prec e succ rappresentano i puntatori agli elementi adiacenti. Quale dei seguenti segmenti di codice deve essere eseguito per eliminare correttamente il nodo puntato da X, assumendo che X non punti né all'inizio né alla fine della lista?

```

Nodo{
public:
    int valore;
    Nodo *succ;
    Nodo *prec;
};

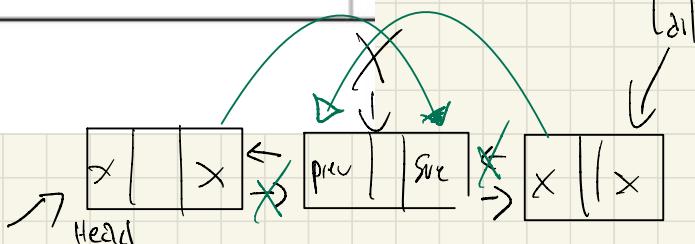
```

Gia fatto! Eliminazione ultimo elemento

Gia fatto

$X \rightarrow buee \rightarrow prev = X \rightarrow prev$

$X \rightarrow prev \rightarrow Suce = X \leftarrow Suce$



Ogni algoritmo di ordinamento basato su confronti richiede almeno

...

Scegliere l'utilizzo corretto di **delete** per l'espressione `ptr = new int[100]`

L'accesso di default ad un membro di classe è

Qual è l'output del seguente frammento di codice?

```
NodeBST<T> *ptr = root;  
while(ptr->getParent() != NULL) {  
    cout << ptr->getKey() << endl;  
}  
while(ptr) {  
    cout << ptr->getKey() << endl;  
    ptr = ptr->getRight();  
}
```

$O(\frac{n}{2})$

$\rightarrow \text{delete}[\] \text{ptr};$
 $\rightarrow \text{private}$

i figli destri del BST stampati ricorsivamente

Scegliere la corretta definizione di una funzione virtuale pura

$\rightarrow \text{virtual void g()=0;}$

L'operazione di enqueue nell'implementazione con una lista linkata con puntatori alla testa e alla coda avviene in tempo

$\rightarrow O(1)$

Un header definito dall'utente viene incluso usando la sintassi ...

$\rightarrow \#define "file.h"$

Quali di questi algoritmi di ordinamento sono basati sul paradigma divide et impera?

$\rightarrow \text{Merge - Quick}$

Nel caso peggiore, qual è il numero di confronti necessari per la ricerca di un elemento all'interno di una lista concatenata semplice?

$\rightarrow O(n)$

Cosa intendiamo dire quando diciamo che un algoritmo X è asintoticamente più efficiente di un altro algoritmo Y?

$\rightarrow X$ è migliore di Y anche per grandi Input

T1 = $5n^2 + 2n - 10$ T2 = $5\sqrt{n} + 22$ T1 > T2

$\rightarrow n^2 + c$

T1 = $5\sqrt{bn}^3$ T2 = $5\sqrt{cn}^5$ T1 > T2

$5\sqrt{b}n^3$ $5\sqrt{c}n^5$ $25n(n^3 \cdot n^5) = 25n(n^8) \rightarrow (n^8)$

T1 = $O(n \log^2 n)$ T2 = $O(1.5n \log n)$

$\rightarrow T_1 > T_2$

T1 = $O((5^n)n \log n)$ T2 = $O((4^n)n)$ $O(5^n) n \log n \mid O(4^n) n \rightarrow T_1 > T_2$

f(n)=O(g(n)) se c,n>0

T1 = $5n^2 + 2n - 10$ T2 = $5\sqrt{n} + 22$ T1 > T2 $5n^2 - 2n \mid 5\sqrt{n} \quad 23(n^2, n^{\frac{1}{2}}) - 2n = 23(n^{\frac{3}{2}}) - 2n = n^{\frac{3}{2}}$

T1 = n^e T2 = n^k G = $\log T_1/T_2$

$n^e \quad n^k \quad \log \frac{n^e}{n^k} = \log(n^{e-k}) (e-k)(\log(n)) = \log(n)$

n var c>1

$O(n) \leq O(\log n) < O(n^e)$

Selection Sort

int x, y; \rightarrow int* const p = &x; \rightarrow const int* q = &y; \rightarrow q = &x;
 \rightarrow *q = 4; trova l'errata

$*q = 4$ errore

```
#include <iostream>
using namespace std;
class abc {
public:
    static int x;
    int i;
    abc() { i = ++x; }
}
int abc::x;
main(){
    abc m, n, p;
    cout << m.x << " " << n.x;
}
```

(No
Indicazioni)

Dalla seguente classe, scegliere la definizione corretta per la funzione membro f

```
template <class T>
class abc {
    void f();
};
```

Trovare le complessità dei seguenti algoritmi

```
int i, j, k = 0;
for (i = n / 2; i <= n; i++) {
    for (j = 2; j <= n; j = j * 2) {
        k = k + n / 2;
    }
}
```

template <class T> void abc<T>::f()

$O(n \log n)$

```
int a = 0, i = N;
while (i > 0) {
    a += i;
    i /= 2;
}
```

$O(\log n)$

Quesito

Considerando la seguente semplice implementazione di una lista concatenata, cosa effettua la funzione boo()?

```
class Nodo{
public:
    int valore;
    Nodo *succ;};
class ListaConcatenata{
public:
    Nodo* testa;};
void boo(Nodo* testa){
if(testa == nullptr) return;
cout << testa->valore << endl;
boo(testa->succ);
}
```

Stampa ricorsivamente i valori della lista

L'ultima riga della seguente funzione inverti() ha una istruzione mancante. Quale di queste istruzioni dovrebbe essere inserita alla fine della procedura per permettere alla funzione inverti() di invertire l'ordine degli elementi della lista concatenata la cui testa è data in input?

```
/* 'testa' punta alla testa di
una lista concatenata */
void inverti(Nodo* testa)
{
    Nodo* prev = nullptr;
    Nodo* current = *testa;
    Nodo* next;
    while (current != nullptr){
        next = current->succ;
        current->succ = prev;
        prev = current;
        current = next;
    }
}
```

Head = prev;