Ultima revisione: giovedì 15 luglio 2021

ESAME DI ALGORITMI E STRUTTURE DATI

*Documento creato da **Alessio Meini**, studente al primo anno di Ingegneria Informatica AA. 2020/2021.*

Si ringraziano tutti quelli che hanno sostenuto e collaborato al progetto!

*In particolare si ringraziano **Andrea Il Migliore** per la soluzione di "void printInevidence()" del primo appello; **Lorenzo Grassi** e **Cristina Maria Rita Lombardo** rispettivamente per la fornitura delle domande del quiz e dell'orale nel secondo appello e **Mariachiara Orrù** per le soluzioni dell'orale sempre del secondo appello.*

*Un ringraziamento speciale va a **Federico Nardi** che ha reso possibile la realizzazione di questo progetto!*

NON ci assumiamo la responsabilità riguardo la completezza e/o correttezza delle informazioni qui riportate.

Mi auguro che questo documento possa essere di aiuto a chi si sta preparando a tenere l'esame di Algoritmi e Strutture Dati.

In bocca al lupo!

Parte delle domande proposte al quiz e all'esame orale tenuto il 9 giugno 2021 (primo appello estivo)

LA RISPOSTA ALLE DOMANDE DEI QUIZ È EVIDENZIATA

QUIZ

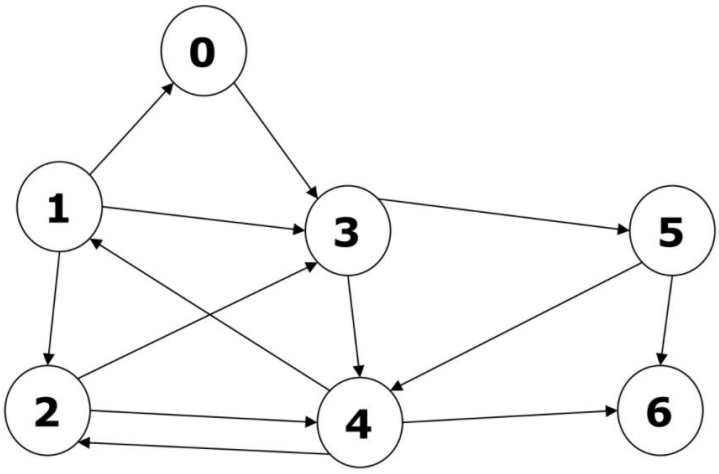
1. Date le seguenti stringhe (xyzzy – xxyzx) quanto è lunga la PLSC?

   ✓ 3(xyz)

2. Mostrare l'output al secondo step del radix sort (a, b, c, d, e) sulla lista [ace ceb bec abc eba bba]

   ✓ eba bba abc ace ceb bec

3. Visita in profondità di un grafico salvato in lista di adiacenza



   ✓ 0,3,4,1,2,6,5

4. Qual è la complessità dell'algoritmo di Dijkstra?

   ✓ O(nlogn+mlogn)

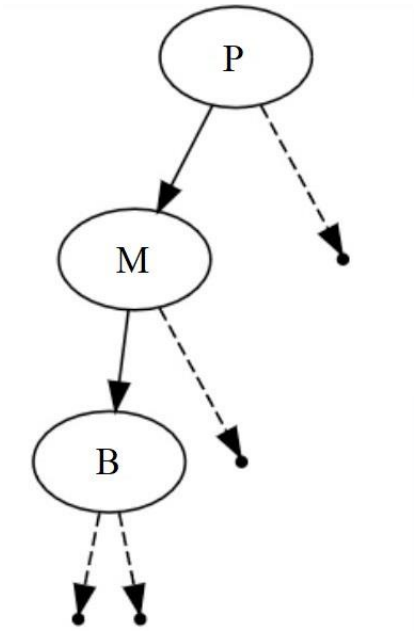
5. A quale categoria di problemi (P,NP, NP-Completi) appartiene SAT-I?

   ✓ SAT nella logica del I ordine non è risolvibile con un algoritmo

6. Un problema si definisce appartenente a NP:

   ✓ Se esiste un algoritmo che verifica una sua soluzione in tempo polinomiale

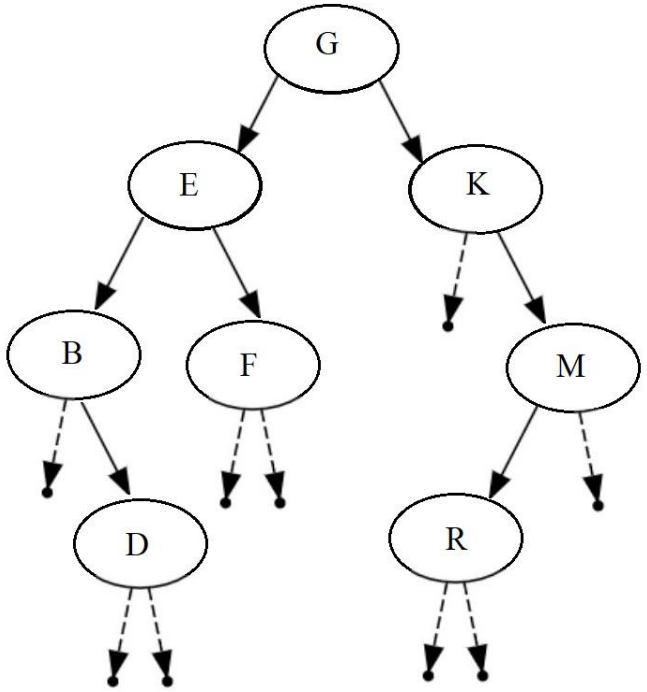
7. Quale definizione è corretta per il seguente albero? (nodi in ordine alfabetico con A<Z, per chiarire la differenza tra figlio sinistro e destro, sono riportati anche i figli vuoti dei nodi)



   ✓ Si definisce albero binario di ricerca DEGENERE

8. Fare la visita anticipata del seguente albero (pre-order) (nodi in ordine alfabetico con A<Z, per chiarire la differenza tra figlio sinistro e destro, sono riportati anche i figli vuoti dei nodi)*

*La domanda potrebbe differire leggermente (nella richiesta o sia nella richiesta che nella risposta) da quella realmente presente al quiz in quanto: si basa sul ricordo di chi ha sostenuto il quiz quel giorno oppure è stata ricostruita in base alle caratteristiche note della domanda.



✓ G E B D F K M R

ALTRE DOMANDE:

Domande sulla gerarchia delle classi

Domande sulle funzioni template

ESEMPIO:

Data una funzione template<class tipo> con un valore statico incrementato ad ogni chiamata, alla modifica del “tipo” in input alla funzione veniva creata una nuova istanza che, alla chiamata della funzione, stampava un valore della variabile statica differente a seconda del “tipo” (esempio con static a=0; a++; : chiamata con<int> a=1, chiamata <double> a= 1, chiamata <int> a= 2)

*La domanda potrebbe differire leggermente (nella richiesta o sia nella richiesta che nella risposta) da quella realmente presente al quiz in quanto: si basa sul ricordo di chi ha sostenuto il quiz quel giorno oppure è stata ricostruita in base alle caratteristiche note della domanda.

PARTE PRATICA

ALFEO:

-**AlberiBinariDiRicerca**

INPUT DI ESEMPIO: 8 5 3 8 2 7 12 11 14

Un output di esempio verrà mostrato basandosi su l'input sopra riportato.

1. Somma dei nodi concordi, dove per concordi si definisce il nodo il cui valore e pari (o dispari) come anche il padre, la radice è concorde.

Output atteso: **34**

Una soluzione possibile:

```
int sommaConcordi(Node* tree, int dad = -1){
    if(!tree)return 0;
    int value=0;
    if((dad == -1)||((tree->value%2==0) == (dad%2==0))) value = tree->value;
    return sommaConcordi(tree->left, tree->value) + sommaConcordi(tree->right, tree-
>value) + value;
}
```

Nel **main()**{…cout<< sommaConcordi(albero.getRoot());}

2. Sommatoria dei nodi con altezza dispari meno la sommatoria dei nodi con altezza pari.

Output atteso: **10**

Una soluzione possibile:

```
int diffPariDispari(Node* tree, int level = 0){
    if(!tree)return 0;
    return diffPariDispari(tree->left, level+1) + diffPariDispari(tree->right,
level+1) + (level%2!=0?tree->value:(-tree->value));
}
```

Nel **main()**{…cout<< diffPariDispari(albero.getRoot());}

3. Sommatoria dei soli nodi pari dove si definisce pari il nodo la cui altezza è pari, solo nodi, niente foglie (la radice per convezione non si considera)

Output atteso: **12**

Una soluzione possibile:

```
int sommaNodiPari(Node* tree, int level = 0){
    if(!tree)return 0;
    if(level != 0 && level%2==0) {
        if (tree->left == nullptr && tree->right == nullptr)
            return tree->value + sommaNodiPari(tree->left, level + 1) +
```

*La domanda potrebbe differire leggermente (nella richiesta o sia nella richiesta che nella risposta) da quella realmente presente al quiz in quanto: si basa sul ricordo di chi ha sostenuto il quiz quel giorno oppure è stata ricostruita in base alle caratteristiche note della domanda.

```
sommaNodiPari(tree->right, level + 1);
    }
    return sommaNodiPari(tree->left, level + 1) + sommaNodiPari(tree->right, level +
1);
}
```

Nel **main()**{…cout<< sommaNodiPari(albero.getRoot());}

4. Prodotto dell'altezza del sottoalbero destro per il l'altezza del sottoalbero sinistro

Output atteso: **6**

Una soluzione possibile:

```
int altezza(Node* tree, int level = 0){
    if(!tree)return level-1;
    return max(altezza(tree->right, level+1), altezza(tree->left,level+1));
}

int prodotto(Node *tree){
    return altezza(tree->left, 1)*altezza(tree->right, 1);
}
```

Nel **main()**{…cout<< prodotto(albero.getRoot());}

5. Sommatoria dei nodi concordi dove concorde è il nodo con etichetta pari (o dispari) ed altezza pari (o dispari)

Output atteso: **28**

Una soluzione possibile:

```
int sommatoriaConcordi(Node* tree, int level = 0){
    if(!tree)return 0;
    return (((tree->value%2==0) == (level%2==0))?tree->value:0)
    +sommatoriaCocordi(tree->left, level+1)+sommatoriaCocordi(tree->right, level+1);
}
```

Nel **main()**{…cout<< sommatoriaConcordi(albero.getRoot());}

6. Implementare la ricerca in un albero binario

7. Sommatoria tra il minimo ed il massimo di un albero

8. Sommatori di nodi completi di altezza dispari, per completo si definisce il nodo che ha due figli (radice altezza 1)

-Heap

*La domanda potrebbe differire leggermente (nella richiesta o sia nella richiesta che nella risposta) da quella realmente presente al quiz in quanto: si basa sul ricordo di chi ha sostenuto il quiz quel giorno oppure è stata ricostruita in base alle caratteristiche note della domanda.

1. Implementare la stampa dello heap (dove i nodi sono disposti graficamente in modo da rappresentare lo heap come un albero), come da esempio:

10

9 6

8 4 3 2

Dove gli elementi appartenenti al sottoalbero di una etichetta K sono tra le parentesi quadre

10 (K=6)

9 6

8 4 [3] [2]

Una soluzione possibile:

```
void printInevidence(int k, int i = 0, int evidenced = -1, int nChild = 0, int j = 0)
{
    if (i >= lista_.size()) return;
    if (isFirstChild(i+1)) std::cout << std::endl;
    if (evidenced + j == i) std::cout << '[' << lista_[i] << "]\t";
    else std::cout <<' '<< lista_[i] << " \t ";
    if (lista_[i] == k)
    {
        evidenced = 2*i + 1;
        nChild = 2;
        j = 0;
    }
    else if (i == evidenced + j)
    {
        if (j < nChild - 1)
        {
            j++;
        }
        else
        {
            evidenced = 2*evidenced + 1;
            nChild *= 2;
            j = 0;
        }
    }

    printInevidence(k, i + 1, evidenced, nChild, j);
}
```

Nel **main()**{…h.printInevidence(K);}

2. Stampa uno heap dove l'elemento il cui figlio destro di un determinato nodo K è tra parentesi quadre

13 (k=3)

[9] 8

4 3 1

*La domanda potrebbe differire leggermente (nella richiesta o sia nella richiesta che nella risposta) da quella realmente presente al quiz in quanto: si basa sul ricordo di chi ha sostenuto il quiz quel giorno oppure è stata ricostruita in base alle caratteristiche note della domanda.

Una soluzione possibile:

```
void printRight(int elem, int i = 0){
    if(i >= lista_.size())return;
    if(isFirstChild(i+1)) std::cout<<std::endl;
    if(2*i+1 < lista_.size()&& lista_[2*i+2] == elem)
        std::cout<<'['<<lista_[i]<<']'<<'\t';
    else
        std::cout<<' '<<lista_[i]<<' '<<'\t';
    printRight(elem, i+1);
}
```

Nel **main()**{…h.printRight(K);}

DUCANGE:

- Implementare lo HeapSort
  Quale caratteristica sfrutta?
  (Si tratta della caratteristica dello Heap di avere la radice con valore maggiore rispetto al resto dei nodi dell'albero)

- Implementare il mergeSort

- Creare una struttura dati albero e creare la funzione di inserimento

- Implementare l'algoritmo di Huffman

- Implementare il bubbleSort

- Implementare il selectionSort

- Implementare il quickSort

- Implementare un miniHeap

*La domanda potrebbe differire leggermente (nella richiesta o sia nella richiesta che nella risposta) da quella realmente presente al quiz in quanto: si basa sul ricordo di chi ha sostenuto il quiz quel giorno oppure è stata ricostruita in base alle caratteristiche note della domanda.

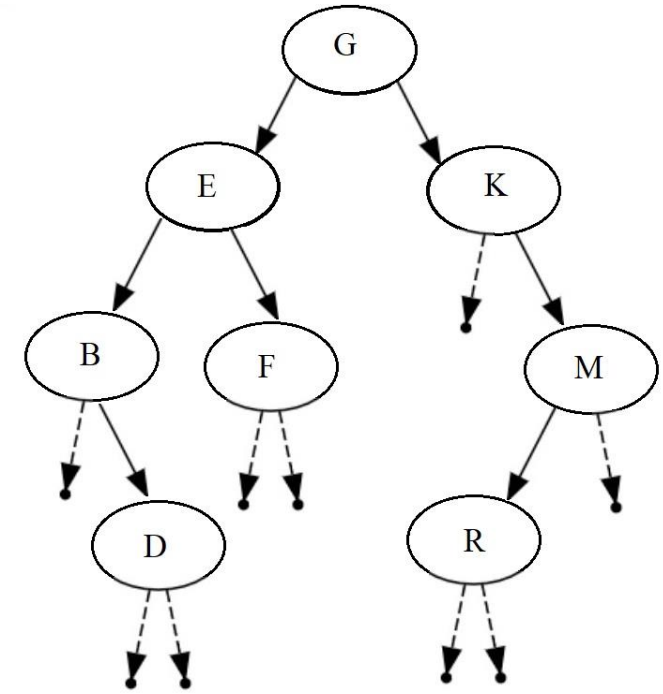
Ultima revisione: giovedì 15 luglio 2021

Parte delle domande proposte al quiz e all'esame orale tenuto il 30 giugno 2021 (secondo appello estivo)

LA RISPOSTA ALLE DOMANDE DEI QUIZ (SE PRESENTE) È EVIDENZIATA

QUIZ

1. Fare la visita anticipata del seguente albero (pre-order) (nodi in ordine alfabetico con A<Z, per chiarire la differenza tra figlio sinistro e destro, sono riportati anche i figli vuoti dei nodi)*



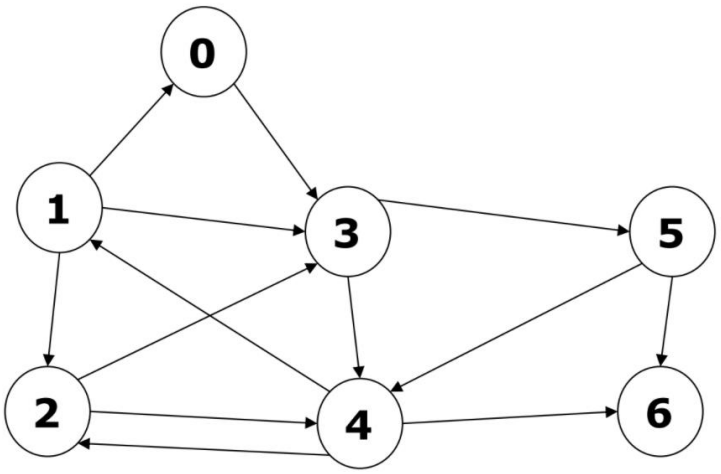
✓ G E B D F K M R

2. Date le seguenti lettere e le relative frequenze{(a, 45) (b, 13) (c, 12) (d, 16) (e, 9) (f, 5)} determinare la loro codifica binaria utilizzando l'algoritmo di Huffman.*

✓ (a, 0) (b, 101) (c, 100) (d, 111) (e, 1101) (f, 1100)

3. Visita in profondità di un grafico salvato in lista di adiacenza*

*La domanda potrebbe differire leggermente (nella richiesta o sia nella richiesta che nella risposta) da quella realmente presente al quiz in quanto: si basa sul ricordo di chi ha sostenuto il quiz quel giorno oppure è stata ricostruita in base alle caratteristiche note della domanda.



✓ 0,3,4,1,2,6,5

4. A quale categoria di problemi (P,NP, NP-Completi) appartiene la soddisfattibilità di una formula nella logica del I ordine?

   ✓ SAT nella logica del I ordine non è risolvibile con un algoritmo

5. Un problema appartiene a P quando:

   ✓ Quando il problema è decisionale ed è risolvibile in tempo polinomiale con un algoritmo deterministico

6. Date le seguenti stringhe (xyzzy – xxyzx) quanto è lunga la PLSC?*

   ✓ 3(xyz)

7. Qual è la complessità del MergeSort?

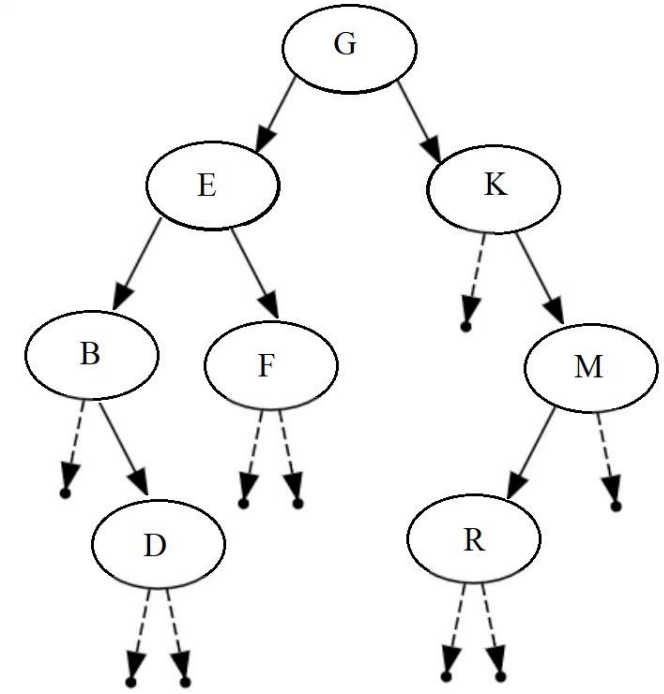
   ✓ Il MergeSort è sempre O(nlogn)

8. Mostrare l'output al secondo step del radix sort sulla lista [190, 051, 054, 207, 088, 010]*

   ✓ 207, 010, 051, 054, 088, 190

*La domanda potrebbe differire leggermente (nella richiesta o sia nella richiesta che nella risposta) da quella realmente presente al quiz in quanto: si basa sul ricordo di chi ha sostenuto il quiz quel giorno oppure è stata ricostruita in base alle caratteristiche note della domanda.

9. Quale definizione è corretta per il seguente albero? [le opzioni erano: è degenere, è bilanciato, non è un vero albero di ricerca, nessuna delle altre] (nodi in ordine alfabetico con A<Z, per chiarire la differenza tra figlio sinistro e destro, sono riportati anche i figli vuoti dei nodi)*



- ✓ In questo caso, tra le opzioni presentate, quella giusta era nessuna delle altre; infatti l'albero è un vero albero binario di ricerca non bilanciato e non degenere.

ALTRE DOMANDE:

Domande sul try catch

Domande sul polimorfismo (tipo l'esempio nel primo appello)

*La domanda potrebbe differire leggermente (nella richiesta o sia nella richiesta che nella risposta) da quella realmente presente al quiz in quanto: si basa sul ricordo di chi ha sostenuto il quiz quel giorno oppure è stata ricostruita in base alle caratteristiche note della domanda.

PARTE PRATICA

ALFEO:

-**AlberiBinariDiRicerca**

INPUT DI ESEMPIO: 8 5 3 8 2 7 12 11 14

Un output di esempio verrà mostrato basandosi su l'input sopra riportato.

1. Riferiamo come incompleti i nodi con meno di due figli. Crea una funzione che stampi la sommatoria dei label dei nodi incompleti di altezza dispari. La radice ha altezza 1.

   Output atteso: **9**

   Una soluzione possibile:

```
int sommatoria(Node* tree, int livello = 1){
    if(tree == NULL)
        return 0;
    if((tree->left == NULL || tree->right == NULL) && livello%2 != 0)
        return tree->value + sommatoria(tree->left,livello+1) + sommatoria(tree-
>right,livello+1);
    return sommatoria(tree->left,livello+1) + sommatoria(tree->right,livello+1);
}
```

Nel **main()**{…cout<< sommatoria(albero.getRoot());}

2. Definisci una funzione la cui chiamata sfrutti le eventuali variabili passate nel main e che stampi la sommatoria dei label dei nodi concordi. Un nodo si dice concorde se ha label pari e anche il padre ha label pari, oppure se ha label dispari e anche il padre ha label dispari. La radice per convenzione è concorde.

   Output atteso: **34**

   Una soluzione possibile:

```
int sommatoriaConcordi(Node* tree, int father = -1){
    if(tree == NULL)
        return 0;
    if((father == -1)||((tree->value%2==0) == (father%2==0))){
        return tree->value + sommatoriaConcordi(tree->left,tree->value) +
sommatoriaConcordi(tree->right,tree->value);
    }
    return sommatoriaConcordi(tree->left,tree->value) + sommatoriaConcordi(tree-
>right,tree->value);
}
```

Nel **main()**{…cout<< sommatoriaConcordi(albero.getRoot());}

3. Definisci una funzione la cui chiamata sfrutti le eventuali variabili passate nel main e che stampi la differenza tra la sommatoria dei nodi ad altezza pari e la sommatoria dei nodi ad altezza dispari. La radice per convenzione è ad altezza uno.*

*La domanda potrebbe differire leggermente (nella richiesta o sia nella richiesta che nella risposta) da quella realmente presente al quiz in quanto: si basa sul ricordo di chi ha sostenuto il quiz quel giorno oppure è stata ricostruita in base alle caratteristiche note della domanda.

Output atteso: **10**

Una soluzione possibile:

```
int differenza(Node* tree, int livello = 1){
    if(tree == NULL)
        return 0;
    if(livello%2 == 0){
        return tree->value + differenza(tree->left,livello+1) + differenza(tree-
>right,livello+1);
    }else{
        return -tree->value + differenza(tree->left,livello+1) + differenza(tree-
>right,livello+1);
    }
}
```

Nel **main()**{…cout<< differenza(albero.getRoot());}

4. Scrivi una funzione che stampa la differenza tra la sommatoria delle foglie ad altezza dispari e il valore della radice dell'albero. La radice per convenzione ha altezza zero*

   Output atteso: **20**

   Una soluzione possibile:

(Controlla bene l'utilizzo nel main!!)

```
int SommaFoglie(Node* tree, int livello){
    if(tree == NULL)
        return 0;
    if(tree->left == NULL && tree->right == NULL && livello%2 != 0)
        return tree->value + SommaFoglie(tree->left,livello+1) + SommaFoglie(tree-
>right,livello+1);
    return SommaFoglie(tree->left,livello+1) + SommaFoglie(tree->right,livello+1);
}
```

Nel main(){…

```
cout<< SommaFoglie(albero.getRoot()->right,1)+SommaFoglie(albero.getRoot()->left,1) -
albero.getRoot()->value;
```

}

5. Definisci una funzione che stampa dell'albero binario solo le foglie con etichette dispari.

   Output atteso: **7 11**

   Una soluzione possibile:

```
void EtichetteDispari(Node* tree){
    if(tree == NULL)
        return;
    if((tree->value%2 != 0) && (tree->left == NULL && tree->right == NULL)){
       cout << tree->value <<' ';
    }
    EtichetteDispari(tree->left);
    EtichetteDispari(tree->right);

}
```

Nel **main()**{…EtichetteDispari(albero.getRoot());}

*La domanda potrebbe differire leggermente (nella richiesta o sia nella richiesta che nella risposta) da quella realmente presente al quiz in quanto: si basa sul ricordo di chi ha sostenuto il quiz quel giorno oppure è stata ricostruita in base alle caratteristiche note della domanda.

6. Definisci una funzione che produca la stampa della sommatoria dei label dei nodi concordi. Un nodo si dice concorde se ha label pari e anche il padre ha label pari, oppure se ha label dispari e anche il padre ha label dispari. La radice per convenzione non è concorde*

   Output atteso: **29**

   Una soluzione possibile:

```
int sommaDeiConcordi(Node* tree, int father = -1){
    if (tree == NULL)
        return 0;
    if((father != -1)&&((tree->value%2==0) == (father%2==0))){
        return tree->value + sommaDeiConcordi(tree->left, tree->value) +
sommaDeiConcordi(tree->right, tree->value);
    }
    return sommaDeiConcordi(tree->left, tree->value) + sommaDeiConcordi(tree->right,
tree->value);
}
```

Nel **main()**{…cout<< sommaDeiConcordi(albero.getRoot());}

7. Una funzione che stampa la differenza tra la sommatoria dei nodi di altezza dispari e nodi di altezza pari. La radice per convenzione ha altezza 0.

   Output atteso: **10**

   Una soluzione possibile:

```
int differenza(Node* tree, int livello = 0){
    if(tree == NULL)
        return 0;
    if(livello%2 == 0){
        return tree->value + differenza(tree->left,livello+1) + differenza(tree-
>right,livello+1);
    }else{
        return -tree->value + differenza(tree->left,livello+1) + differenza(tree-
>right,livello+1);
    }
}
```

Nel **main()**{…cout<< differenza(albero.getRoot());}

8. Una funzione che ritorna l'altezza di un elemento passato in input. La radice ha altezza 0.

   Si passi "7" come elemento in input alla funzione:
   Output atteso: **2**

   Una soluzione possibile:

```
int AltezzaElemento(Node* tree, int x,int livello = 0){
    if(tree == NULL){
        return 0;
    }
    if(x < tree->value){
        return AltezzaElemento(tree->left,x,livello+1) + AltezzaElemento(tree-
>right,x,livello+1);
    }else if(x > tree->value){
        return AltezzaElemento(tree->left,x,livello+1) + AltezzaElemento(tree-
>right,x,livello+1);
    }else{
```

*La domanda potrebbe differire leggermente (nella richiesta o sia nella richiesta che nella risposta) da quella realmente presente al quiz in quanto: si basa sul ricordo di chi ha sostenuto il quiz quel giorno oppure è stata ricostruita in base alle caratteristiche note della domanda.

```
        return livello;
    }
}
```

Nel **main()**{…cout<<AltezzaElemento(albero.getRoot(), 7);}

9. Funzione che produce la stampa della sommatoria dei label dei nodi incompleti di altezza dispari. Incompleti: nodi che hanno meno di due figli. La radice per convenzione ha altezza 1.

   Output atteso: **9**

   Una soluzione possibile:

```
int SommatoriaIncompleti(Node* tree, int livello = 1){
    if(tree == NULL)
        return 0;
    if((livello%2 != 0) && (tree->left == NULL || tree->right == NULL)){
        return tree->value + SommatoriaIncompleti(tree->left,livello+1) +
SommatoriaIncompleti(tree->right,livello+1);
    }
    return SommatoriaIncompleti(tree->left,livello+1) + SommatoriaIncompleti(tree-
>right,livello+1);
}
```

Nel **main()**{…cout<<SommatoriaIncompleti(albero.getRoot());}

10. Scrivi la funzione che produce la stampa della sommatoria delle foglie di altezza dispari. La radice dell'albero ha altezza 0.

    Output atteso: **25**

    Una soluzione possibile:

```
int SommatoriaFoglie(Node* tree, int livello = 0){
    if(tree == NULL)
        return 0;
    if((livello%2 != 0) && (tree->left == NULL && tree->right == NULL)){
        return tree->value;
    }
    return SommatoriaFoglie(tree->left,livello+1) + SommatoriaFoglie(tree-
>right,livello+1);
}
```

Nel **main()**{…cout<<SommatoriaFoglie(albero.getRoot());}

-Heap

1. Crea una funzione che, lavorando su uno heap, stampa il contenuto dello heap, le etichette dei nodi che hanno un determinato valore k passato per valore vengono stampate tra parentesi quadre.

   [10] (K=6)

   9 6

   8 4 3 2

*La domanda potrebbe differire leggermente (nella richiesta o sia nella richiesta che nella risposta) da quella realmente presente al quiz in quanto: si basa sul ricordo di chi ha sostenuto il quiz quel giorno oppure è stata ricostruita in base alle caratteristiche note della domanda.

Una soluzione possibile:

```
void stampa(int k) {
    for (int i = 0; i < lista_.size(); i++) {
        if (isFirstChild(i+1)) {
            std::cout << std::endl;
        }
        if (lista_[i] == k) {
            std::cout << '[' << lista_[i] << "]\t";
        } else {
            std::cout << lista_[i] << "\t";
        }
    }
}
```

Nel **main()**{…stampa(K);}

2. Scrivi la funzione che stampa lo heap e in particolare stampa tra parentesi quadre gli elementi il cui figlio destro ha un determinato valore k passato dal main.

   [10] (K=6)

   9 6

   8 4 3 2

Una soluzione possibile:

```
void stampaDestro(int k){
    for(int i = 0; i < lista_.size(); ++i){
        if(isFirstChild(i+1)){
            std:: cout << std:: endl;
        }
        if((2*i+2 < lista_.size()) && (lista_[2*i+2] == k)){
            std::cout << '[' << lista_[i] << "]\t";
        }else{
            std::cout <<' '<< lista_[i] << " \t";
        }
    }
}
```

Nel **main()**{…stampaDestro(K);}

3. Stampa dello heap, in modo che sia tra parentesi quadre il valore delle etichette dei nodi, il cui figlio sinistro ha un determinato valore k.

   10 (K=8)

   [9] 6

   8 4 3 2

Una soluzione possibile:

```
void stampaSinistro(int k){
    for(int i = 0; i < lista_.size(); ++i){
        if(isFirstChild(i+1)){
            std:: cout << std:: endl;
        }
        if((2*i+1 < lista_.size()) && (lista_[2*i+1] == k)){
            std::cout << '[' << lista_[i] << "]\t";
```

*La domanda potrebbe differire leggermente (nella richiesta o sia nella richiesta che nella risposta) da quella realmente presente al quiz in quanto: si basa sul ricordo di chi ha sostenuto il quiz quel giorno oppure è stata ricostruita in base alle caratteristiche note della domanda.

```
        }else{
            std::cout << lista_[i] << " \t";
        }
    }
}
```

Nel **main()**{…stampaSinistro(K);}

4. Stampa dello heap, in modo che sia fra parentesi quadre il valore delle etichette dei nodi il cui padre ha un determinato valore k.

   10 (K=7)

   7 6

   [4] [3] 2 5

Una soluzione possibile:

```
void stampa(int k) {
    for (int i = 0; i < lista_.size(); i++) {
        if (isFirstChild(i+1)) {
            std::cout << std::endl;
        }
        if ((2*i+1< lista_.size()) && (lista_[2*i+1] == k)) {
            std::cout << '[' << lista_[i] << "]\t";
        } else {
            std::cout << lista_[i] << " \t";
        }
    }
}
```

Nel **main()**{…stampa(K);}

DUCANGE:

Il professore all'inizio dell'orale chiede chiarimenti riguardo le domande a cui si è risposto in maniera errata nel quiz. In base al tipo di errore può chiedere spiegazioni a riguardo oppure approfondire e/o incentrare l'interrogazione su quello specifico argomento.

- Cos'è una relazione di ricorrenza?
- Complessità del mergesort?
- Scrivi e spiega il corpo principale del mergesort (a scelta tra quello per vettori e quello per liste)
- Come si chiama il paradigma che utilizziamo nel mergesort?
- Dato un albero binario, scrivi la funzione che calcola il numero di foglie

*La domanda potrebbe differire leggermente (nella richiesta o sia nella richiesta che nella risposta) da quella realmente presente al quiz in quanto: si basa sul ricordo di chi ha sostenuto il quiz quel giorno oppure è stata ricostruita in base alle caratteristiche note della domanda.

- Dato un albero binario, scrivi la funzione che calcola il numero di nodi
- Dato un albero binario, scrivi la funzione che cancella l'intero albero
- Scrivi e spiega il corpo principale dell'heapsort. Scrivi la buildheap. Complessità dell'heapsort?
- Metodo hash. Perché e in quale contesto è stato introdotto questo concetto?
- Qual è la differenza tra un hash in cui non posso cancellare gli elementi e un hash in cui posso farlo?
- Scrivi la funzione di inserimento nell'hash
- Dichiarazione della classe grafo gestito con liste di adiacenza e implementazione dei metodi più importanti
- Crea un pezzo di codice che implementa l'algoritmo di huffman
- Scrivi l'algoritmo per calcolare la lunghezza della più grande sotto-sequenza comune
- Scrivi la funzione di stampa della plsc. Complessità dell'algoritmo?
- Spiega e scrivi l'algoritmo di dijkstra come pseudocodice
- Qual è il limite inferiore raggiungibile da un algoritmo di ordinamento? Quali algoritmi lo raggiungono?
- Scrivi e spiega il codice del counting sort
- Parla dello heap, scrivi la classe heap e i principali metodi pubblici
- Parla del radix sort e scrivi lo pseudo-codice

*La domanda potrebbe differire leggermente (nella richiesta o sia nella richiesta che nella risposta) da quella realmente presente al quiz in quanto: si basa sul ricordo di chi ha sostenuto il quiz quel giorno oppure è stata ricostruita in base alle caratteristiche note della domanda.