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

Algoritmi e Strutture Dati – 20/06/2023 – 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

Si consideri questo algoritmo:

```
int P(int[][]A, int n)
    return R(A, 1, 1, n, n)
```

```
int R(int[][]A, int x1, int y1, int x2, int y2)
    if x1 == x2 and y1 == y2 then
        return A[x1][y1]
    else
        int tot = 0
        for x = x1 + 1 to x2 − 1 do
            tot = tot + A[x][y1]
            tot = tot + A[x][y2]
        for y = y1 to y2 do
            tot = tot + A[x1][y]
            tot = tot + A[x2][y]
        int xm = ⌊(x2 + x1)/2⌋
        int ym = ⌊(y2 + y1)/2⌋
        tot = tot + R(A, x1, y1, xm, ym)
        tot = tot + R(A, xm + 1, ym + 1, x2, y2)
        return tot
```

Scrivere l'equazione di ricorrenza associata a questo algoritmo e calcolare la complessità computazionale che ne deriva. Per semplificarvi la vita, assumete che la dimensione n sia una potenza di 2.

A2 – Somma su albero – Punti ≥ 10

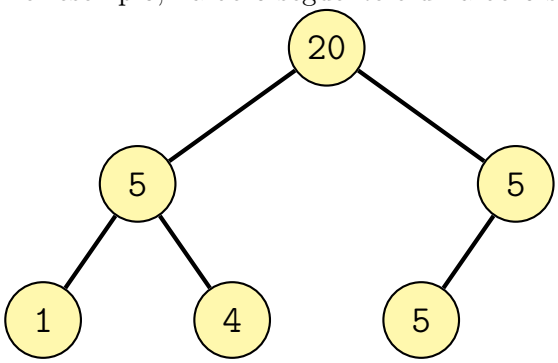
Scrivere un algoritmo

boolean isSumTree(TREE T)

che prenda in input un albero binario e restituisca **true** se T è un albero sommario, **false** altrimenti. Un albero binario è *sommario* se per ogni nodo interno u di T, il valore contenuto nel nodo è pari alla somma dei valori contenuti nei sottoalberi sinistro e destro di u.

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

Per esempio, l'albero seguente è un albero sommario.



A3 – Tree fire – Punti ≥ 12

Sia t un nodo qualunque di un albero binario e supponiamo di appiccare un fuoco in tale nodo. In 1 unità di tempo, il fuoco si diffonde dal nodo t ai suoi figli e a suo padre; da lì, il fuoco può propagarsi ulteriormente. Scrivere un algoritmo

int fire(TREE t)

che restituisca il tempo necessario affinché tutti i nodi dell'albero prendano fuoco a partire dal nodo t.
Discutere informalmente la correttezza e la complessità dell'algoritmo proposto.

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

Algoritmi e Strutture Dati – 20/06/2023 – Parte B

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.

B1 – Misciotto 1 – Punti ≥ 8

Date due stringhe S_1 e S_2 di lunghezza n_1 e n_2, una stringa T è un *misciotto* di S_1 e S_2 se T ha lunghezza $n = n_1 + n_2$ ed è ottenuta intercalando caratteri di S_1 e S_2, conservando però l'ordine originale dei caratteri all'interno di ciascuna delle due stringhe.

Scrivere un algoritmo

printMixing(ITEM[] S_1, ITEM[] S_2, **int** n_1, **int** n_2)

che prenda in input le stringhe S_1, S_2 e le rispettive lunghezze, e stampi tutti i misciotti di S_1 e S_2 Discutere informalmente correttezza e complessità computazionale.

Per esempio,

- se S_1 = `"ABC"` e S_2 = `"DEF"`, la stringa T = `"ADBCEF"` è un misciotto di S_1 e S_2;
- se S_1 = `"ABC"` e S_2 = `"ABC"`, la stringa T = `"ABCCBA"` non è un misciotto di S_1 e S_2.

B2 – Misciotto 2 – Punti ≥ 12

Scrivere un algoritmo

boolean isMixing(ITEM[] S_1, ITEM[] S_2, ITEM[] T, **int** n_1, **int** n_2, **int** n)

che prenda in input le stringhe S_1, S_2, T e le rispettive lunghezze, e restituisca **true** se T è un misciotto di S_1 e S_2, **false** altrimenti. Discutere informalmente correttezza e complessità computazionale.
Algoritmi corretti di complessità non polinomiale prenderanno 50% del punteggio.

B3 – Conferenza – Punti ≥ 10

State organizzando una conferenza scientifica. Gli articoli (*paper*) sono stati sottomessi; il vostro compito è assegnare gli articoli ai revisori (*reviewer*), i quali li leggeranno e daranno la loro opinione (*review*). I revisori dovrebbero essere esperti dell'area scientifica associata agli articoli che leggono, ma non sempre è possibile soddisfare totalmente questo requisito.

Il problema da risolvere è il seguente:

- esiste un insieme R contenente n_r revisori;
- esiste un insieme P contenente n_p articoli sottomessi;
- esiste un insieme A contenente n_a aree scientifiche;
- ogni articolo $p \in P$ è associato a un'area scientifica $a_p \in A$;
- ogni revisore $r \in R$ è esperto di un insieme di aree scientifiche $E_r \subseteq A$;
- ogni articolo deve avere esattamente 3 revisori; almeno 2 di esso devono essereesperti nell'area scientifica dell'articolo, il terzo può essere esperto oppure no;
- ogni revisore può scrivere al massimo k review.

Descrivere un algoritmo che trovi un assegnamento fra revisori e articoli, che rispetti le regole descritte, e discuterne la complessità computazionale. Se possibile, identificare sotto quale condizione tale assegnamento non esiste.