ESAME 24/06/20

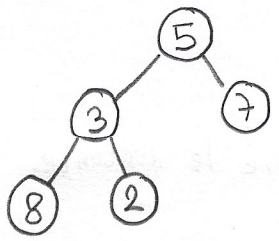
1)
```
void AlternaArray (int A[], int n) {
    for (int i=1; i<n; i++) {
        //Verifica le condizioni per rendere alternate
        if (i %=1 && A[i-1] ≤ A[i]) {
            // Se siamo in una posizione dispari e la condizione non è rispettata, scambio
            int temp= A[i-1];
            A[i-1] = A[i];
            A[i] = temp;
        }
        else if (i %2 ==0 && A[i-1]≥ A[i]) {
            // Se siamo in una posizione pari e la condizione non è rispettata, scambio
            int temp = A[i-1];
            A[i-1] = A[i]
            A[i] = temp;
        }
    }
}
```

$T(n) = O(n)$

2) ESISTENZA DI T'

Un albero binario di ricerca (BST) può avere la stessa visita in pre-ordine di T solo se T soddisfa le proprietà di un BST durante la visita in pre ordine. NON ESISTE SEMPRE UN T'

Esempio



PRE-ORDINE: 5-3-8-2-7

Questo ordine non può rappresentare un BST, poiché 8>5, ma si trova nel sottoalbero sinistro.

```
bool check(int preorder[], int n){
    int stack[n];
    int top = -1;
    int root = INT_MIN;
    for(int i=0; i<n; i++){
        //se troviamo un valore minimo della radice corrente, non è un BST
        if(preorder[i] < root){
            return false;
        }
        //Pop dallo stack finchè non troviamo il genitore corretto
        while(top ≥ 0 && preorder[i] > stack[top]){
            root = stack[top--];
        }
        //Push del nodo corrente nello stack
        stack[++top] = preorder[i];
    }
    return true;
}
```

$T(n) = O(n)$

3) L'algoritmo è una variante dell'algoritmo di Floyd-Warshall per individuare cicli negativi in un grafo diretto pesato.

FUNZIONAMENTO:

- Inizializza la matrice A con i logaritmi dei pesi degli archi
- Usa MyFunction, che implementa la regola di aggiornamento di Floyd-Warshall
- Ritorna TRUE se rileva un ciclo negativo ($B[i][i] \neq 0$).

CORRETTEZZA:

- L'algoritmo sfrutta la relazione $d_{ij} = \min(d_{ij}, d_{ik} + d_{kj})$ per aggiornare le distanze.
- Se $B[i][i] \neq 0$, allora c'è un ciclo negativo.

COMPLESSITÀ:

$T(n) = O(n^4)$ poiché l'algoritmo esegue n iterazioni di Floyd-Warshall ($O(n^3)$)

ESAME 24/06/20

$D^{(0)}$

	1	2	3	4	5
1	0	1	∞	∞	∞
2	∞	0	0,1	∞	10
3	0,1	∞	0	1	∞
4	∞	∞	∞	0	∞
5	1	∞	∞	0,1	0

$D^{(1)}$

	1	2	3	4	5
1	0	1	∞	∞	∞
2	∞	0	0,1	∞	10
3	0,1	1,1	0	1	∞
4	∞	∞	∞	0	∞
5	1	2	∞	0,1	0

$D^{(2)}$

	1	2	3	4	5
1	0	1	1,1	∞	11
2	∞	0	0,1	∞	10
3	0,1	1,1	0	1	11,1
4	∞	∞	∞	0	∞
5	1	2	2,1	0,1	0

$D^{(3)}$

	1	2	3	4	5
1	0	1	1,1	2,1	11
2	0,2	0	0,1	1,1	10
3	0,1	1,1	0	1	11,1
4	∞	∞	∞	0	∞
5	1	2	2,1	0,1	0

$D^{(4)}$

	1	2	3	4	5
1	0	1	1,1	2,1	11
2	0,2	0	0,1	1,1	10
3	0,1	1,1	0	1	11,1
4	∞	∞	∞	0	∞
5	1	2	2,1	0,1	0

$D^{(5)}$

	1	2	3	4	5
1	0	1	1,1	2,1	11
2	0,2	0	0,1	1,1	10
3	0,1	1,1	0	1	11,1
4	∞	∞	∞	0	∞
5	1	2	2,1	0,1	0

FALSE, NO CICLI NEGATIVI SULLA DIAGONALE, TUTTI I VALORI SONO ≥ 0

4) a) Se CICLO-NEGATIVO $\leq_p$ CLIQUE, allora P=NP VERO

CICLO-NEGATIVO è in NP e ridurlo a CLIQUE implica che NP $\leq$ P

b) Se ISOMORFISMO-DI-GRAFI $\leq_p$ CLIQUE, allora P=NP FALSO

ISOMORFISMO-DI-GRAFI non è noto essere in NP-completo

c) Se CLIQUE $\leq_p$ ISOMORFISMO-DI-GRAFI, allora P=NP FALSO

Non esiste prova che ISOMORFISMO-DI-GRAFI sia NP-completo

d) Se CLIQUE $\leq_p$ CICLO-NEGATIVO, allora P=NP VERO