Stive Cozi Aplicatii

Cosmin Glod

Ce este o stiva/coada ?

Stiva (Stack) este o structura de date in care elementele se aduna si se scot dupa principiul LIFO (Last In First Out). Asta inseamna ca ultimul element adaugat este primul care va fi scos.

O coada (Queue) este o structura in care elementele sunt procesate dupa principiul FIFO (First In First Out). Primul element adaugat este primul element care va fi scos

Vom vedea in continuare ca o coada este similara unei stive, din mai multe puncte de vedere...

Exemple de stive

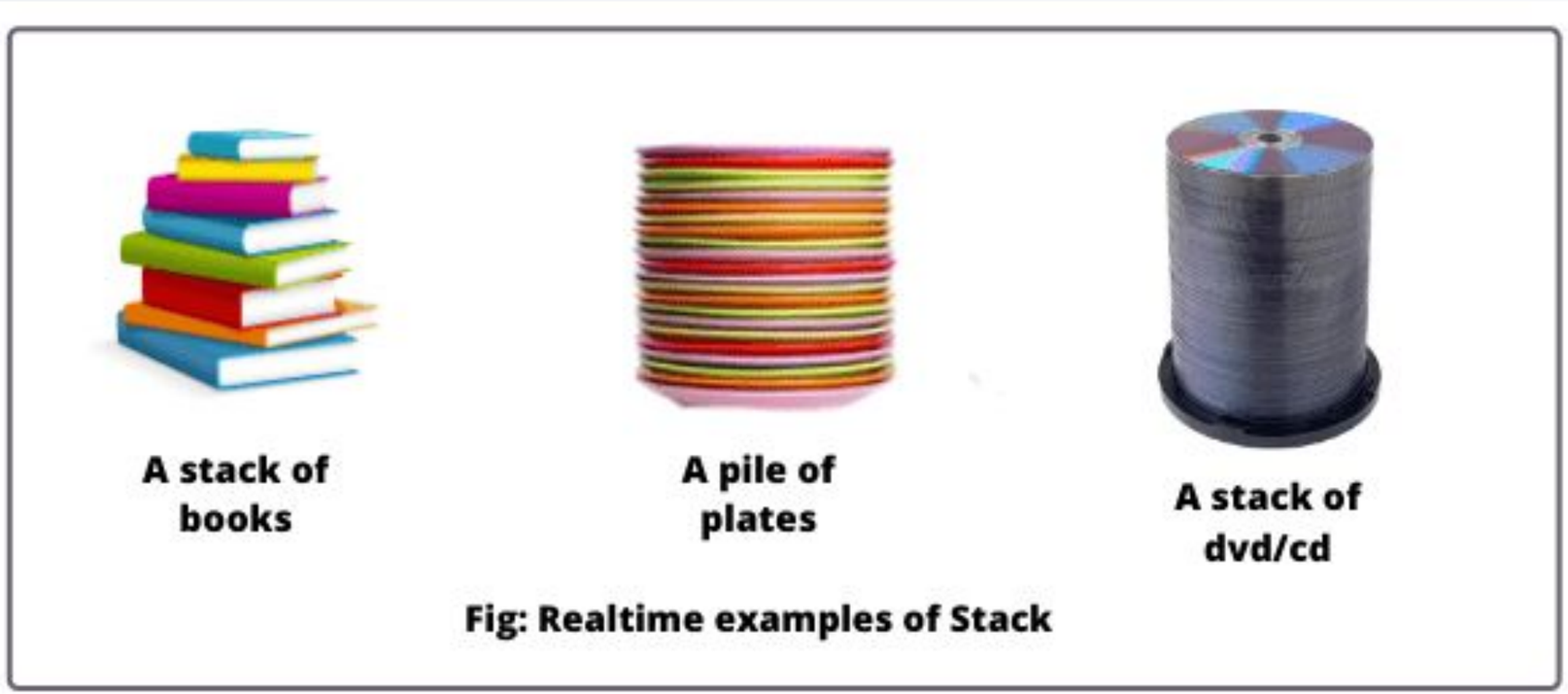


Fig: Realtime examples of Stack

Aplicatia 1 – Paranteze

Se dau `n` șiruri de paranteze rotunde. Să se stabilească, despre fiecare șir, dacă este corect parantezat – adică dacă parantezele se închid corect.

Un șir de paranteze `S` rotunde este corect parantezat dacă:

- `S` este șirul vid, sau
- `S = (T)` și `T` este corect parantezat, sau
- `S = AB`, iar `A` și `B` sunt corect parantezate.

https://www.pbinfo.ro/probleme/848/paranteze1 - link problema

Exemplu:

`paranteze1.in`

```
4
(())
)(()
()((()())())
()(
```

`paranteze1.out`

```
1
0
1
0
```

Aplicatia 1 – Paranteze

Exemplu:

paranteze1.in

```
4
(())
)(()
()((()())())
()(
```

paranteze1.out

```
1
0
1
0
```

Pasi de rezolvare:

1. Initiem o stiva goala
2. Parcurgem fiecare caracter din expresie:
   a. Daca intalnim paranteza deschisa, o adaugam in stiva
   b. Daca intalnim paranteza inchisa, scoatem o paranteza deschisa de pe stiva (daca nu exista paranteza deschisa pe stiva, afisam 0)
3. La sfarsit, daca stiva este goala, atunci expresia este corect parantezata, in caz contrar este incorecta.

Aplicatia 1 – Paranteze
Implementare stiva

	Implementare cu array:	**Implementare cu STL:** #include <stack>
Declarare:	int stiva[N_MAX], nr_elem = 0;	stack<int> stiva;
Adaugarea elementului x:	stiva[nr_elem++] = x;	stiva.push(x);
Eliminarea unui element:	--nr_elem;	stiva.pop();
Accesarea elementului din varf:	stiva[nr_elem - 1]	stiva.top();

Aplicatia 1 - Paranteze

Implementare

```
bool paranteze(char str[]) {
    stack<char> stiva;
    bool ok = true;
    // iteram prin sir
    for (int i = 0; ok && str[i] != '\0'; ++i) {
        if (str[i] == '(') {
            // adaugam in stiva
            stiva.push(str[i]);
        }
        else {
            // verificam daca stiva e goala
            if (stiva.empty())
                ok = false;
            else // daca nu e goala
                stiva.pop(); // scoatem ultima paranteza pusa
        }
    }
    if (!stiva.empty()) // daca stiva mai are elemente
            ok = false; // inseamna ca nu e corect parantezata

    return ok;
}
```

Aplicatia 2 - Flood Fill

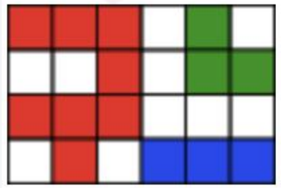
Se dă o matrice cu `n` linii și `m` coloane și elemente `0` sau `1`, care reprezintă harta unei planete, în care `1` înseamnă uscat, iar `0` înseamnă apă. Două elemente `1` care se învecinează pe linie sau pe coloană (nu și pe diagonală) fac parte din același continent.

Să se determine câte continente sunt pe hartă.

`fill.in`

```
4 6
1 1 1 0 1 0
0 0 1 0 1 1
1 1 1 0 0 0
0 1 0 1 1 1
```

Cele `3` continente sunt evidenţiate mai jos:



`fill.out`

```
3
```

https://www.pbinfo.ro/probleme/837/fill - link problema

Aplicatia 2 – Flood Fill

Pasi de rezolvare:

Pentru fiecare celula (i,j) din matrice:

1. Daca elementul curent este 1, am gasit un continent
   a. Incrementam continentele cu 1
   b. Aplicam Flood Fill de la celula (i, j) pentru a marca tot continentul curent, astfel incat sa nu fie numarat din nou
2. Altfel, nu facem nimic, intrucat cautam alt continent

Flood Fill implementare

a. Initializeaza o stiva si adauga coordonata (i,j) in ea
b. Cat timp stiva nu e goala
   i. Marcheaza celula din varful stivei ca vizitata
   ii. Scoate celula vizitata din stiva
   iii. Verifica vecinii celulei scoase. Daca vreunul are valoarea 1, adauga-l in stiva.

Aplicatia 2 - Flood Fill
Implementare

```
int solve(int a[][M_MAX], int n, int m) {
    int continente = 0;
    for (int i = 1; i <= n; ++i) {
        for (int j = 1; j <= m; ++j) {
            if (a[i][j] == 1) { // am dat de un nou continent
                ++continente;
                floodFill(a, n, m, i, j);
            }
        }
    }

    return continente;
}
```

Aplicatia 2 - Flood Fill

Implementare

```
void floodFill(int a[][M_MAX], int n, int m, int i, int j) {
    stack<pair<int, int> > stiva;
    stiva.push(make_pair(i, j));

    while (!stiva.empty()) {
        pair<int, int> celulaCurenta = stiva.top();
        // marchez celula din varful stivei ca vizitata
        a[celulaCurenta.first][celulaCurenta.second] = -1;

        // scot din stiva celula curenta
        stiva.pop();

        // adaug in stiva vecinii celulei curente care au valoarea 1
        for (int k = 0; k < 4; ++k) { // folosesc vectorii de directie
            int x = celulaCurenta.first + dx[k];
            int y = celulaCurenta.second + dy[k];
            if (a[x][y] == 1)
                stiva.push(make_pair(x, y));
        }
    }
}
```

Tema

https://www.pbinfo.ro/probleme/882/lac

https://www.pbinfo.ro/probleme/1461/meteoriti

https://www.infoarena.ro/problema/labirint2

https://infoarena.ro/problema/parantezare

Tema nu este obligatorie, dar este puternic recomandata!