Structuri de Date Elementare

Stive Cozi Deque Mars? Batog?

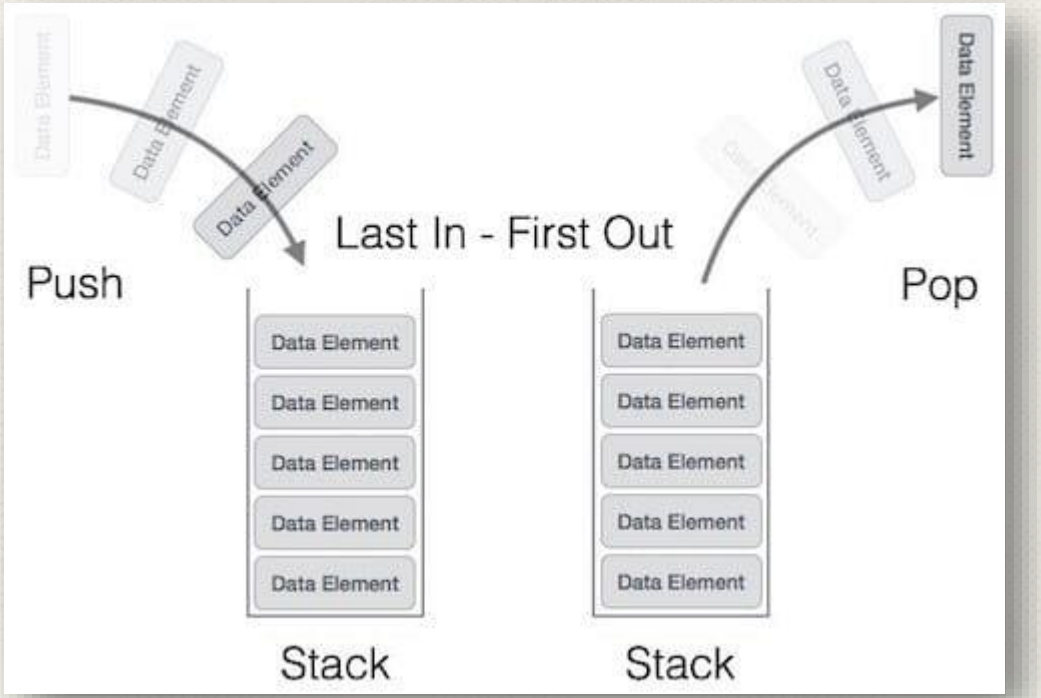
Stive (Stack)

- Sunt structuri de date de tip **LIFO** (**L**ast **I**n **F**irst **O**ut)
- Avem acces numai la elementul din vârf (top)

- Operații de bază:
  - **Push** - adăugarea unui element (în vârf)
  - **Pop** - eliminarea elementului din vârf

- Operații suplimentare:
  - **Size()** - numărul de elemente
  - **isEmpty()** - returnează **true** dacă numărul de elemente este exact 0
  - **Peek()** sau **top()** - ne spune valoarea din vârf fără să o extragă

Stive (Stack)


Data Element
Data Element
Data Element
Last In - First Out
Push
Pop
Data Element
Data Element
Data Element
Data Element
Data Element
Data Element
Data Element
Data Element
Data Element
Data Element
Stack
Stack

Stive (Stack)

- Metode de implementare:
  - Stivă ca Vector
    - Vizualizare
    - Implementare (găsiți în secțiunea de implementare ca array)
  - Stivă ca Listă
    - Vizualizare
    - Implementare (găsiți în secțiunea de implementare ca linked list)
  - Stivă în **C++** - https://en.cppreference.com/w/cpp/container/stack
- **Obs.**
  - Când introducem elemente într-o stivă,trebuie să incrementăm top-ul și apoi să adăugam elementul.
  - Când ștergem un element,trebuie întâi să ștergem elementul și apoi să decrementăm top-ul.

Implementare funcții Push și Pop

```
bool Stack::push(int x)
{
    if (top >= (MAX - 1)) {
        cout << "Stack Overflow";
        return false;
    }
    else {
        a[++top] = x;
        cout << x << " pushed into stack\n";
        return true;
    }
}

int Stack::pop()
{
    if (top < 0) {
        cout << "Stack Underflow";
        return 0;
    }
    else {
        int x = a[top--];
        return x;
    }
}
```

Exerciții

- https://www.infoarena.ro/problema/nrpits
- Inversarea unui text
- Problema parantezelor

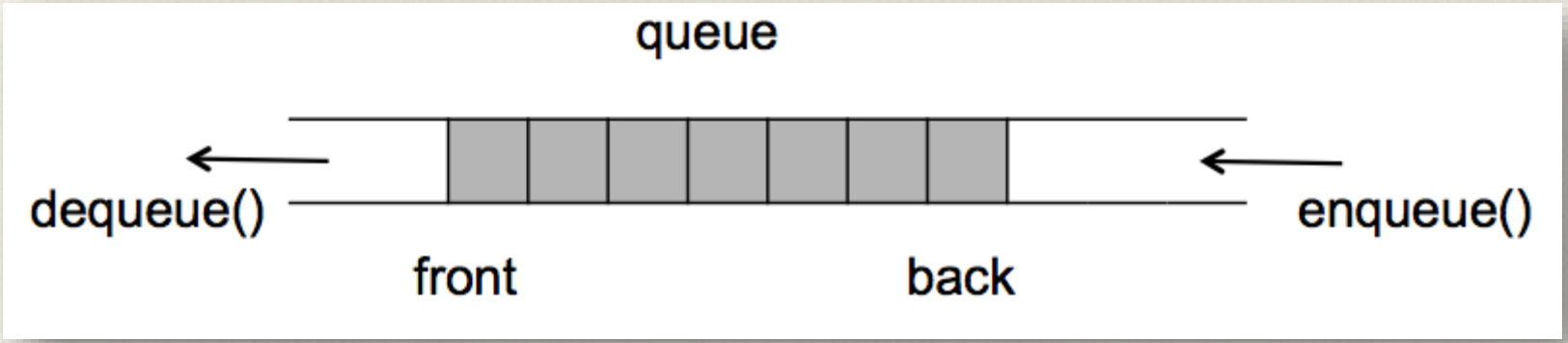
Cozi (Queue)

- Sunt structuri de date de tip **FIFO** (**F**irst **I**n **F**irst **O**ut)
- Avem acces la primul și la ultimul element (head & tail / front & back)

- Operații de bază:
  - Push - adăugarea unui element la coadă
  - Pop - eliminarea unui element

- Operații suplimentare:
  - Size() - numărul de elemente
  - isEmpty() - returnează **true** dacă numărul de elemente este exact 0
  - First() - ne spune valoarea de la început fără să o extragă
  - Last() - ne spune valoarea de la sfârșit fără să o extragă

Cozi (Queue)



Cozi (Queue)

- Metode de implementare:
  - Coadă ca Vector
    - Vizualizare
    - Implementare
  - Coadă ca Listă
    - Vizualizare
    - Implementare
  - Coadă în C++ - https://en.cppreference.com/w/cpp/container/queue

Implementare funcții Enqueue și Dequeue

```
void queueEnqueue(int data)
  {
    // check queue is full or not
    if (capacity == rear) {
      printf("\nQueue is full\n");
      return;
    }

    // insert element at the rear
    else {
      queue[rear] = data;
      rear++;
    }
    return;
  }
```

```
// function to delete an element
  // from the front of the queue
  void queueDequeue()
  {
    // if queue is empty
    if (front == rear) {
      printf("\nQueue is  empty\n");
      return;
    }
    // shift all the elements from index 2
till rear
    // to the left by one
    else {
      for (int i = 0; i < rear - 1; i++) {
        queue[i] = queue[i + 1];
      }
      // decrement rear
      rear--;
    }
    return;
  }
```

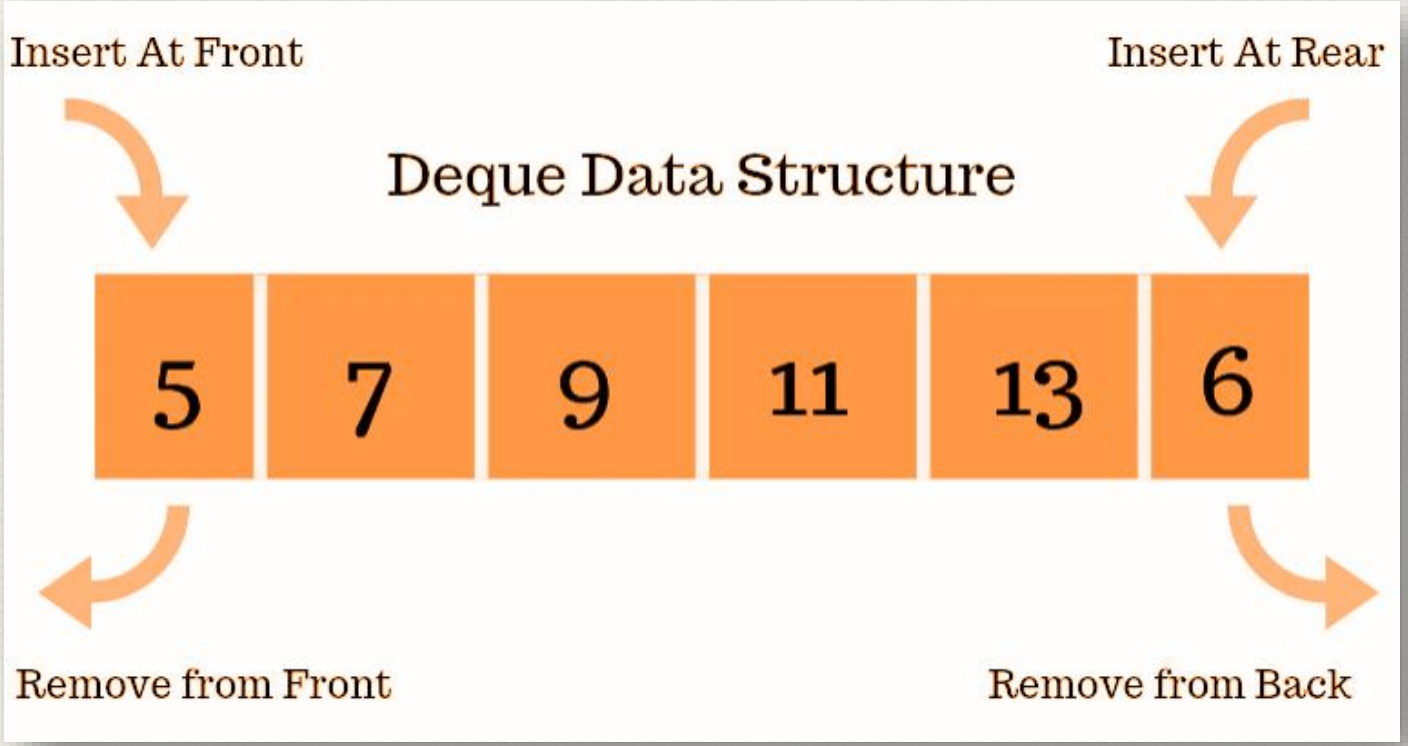
Deque

- Double ended queue (coadă cu două capete)

- Operații de bază:
  - Push Front
  - Push Back
  - Pop Front
  - Pop Back

- Operații suplimentare
  - Size()
  - Front()
  - Back()
  - isEmpty()

Deque


Insert At Front
Insert At Rear
Deque Data Structure
5
7
9
11
13
6
Remove from Front
Remove from Back

Deque

- Double ended queue (coadă cu două capete)

- Metode de implementare:
  - Deque ca Listă
    - Vizualizare
    - Implementare

  - Deque ca Array
    - Implementare

  - Deque în C++ - https://en.cppreference.com/w/cpp/container/deque

Exerciții

- https://infoarena.ro/problema/deque

- Book Pile - https://codeforces.com/problemsets/acmsguru/problem/99999/271

Problemă

Se dă un vector cu n elemente și apoi n operații de genul:

- 1 i j → care este minimul din intervalul [i,j]
- 2 i x → modificați elementul de pe poziția i în x

0	1	2	3	4	5	6	7	8
3	9	2	5	7	34	6	11	8

Problemă

Se dă un vector cu n elemente și apoi n operații de genul:

- 1 i j → care este minimul din intervalul [i,j]
- 2 i x → modificați elementul de pe poziția i în x

Idei?

Șmenul lui Batog

Se dă un vector cu n elemente și apoi n operații de genul:

- 1 i j → care este minimul din intervalul [i,j]
- 2 i x → modificați elementul de pe poziția i în x

Idee:
Împărțim vectorul în zone de lungime L și calculăm minimul pe fiecare zonă în parte.

0	1	2	3	4	5	6	7	8
3	9	2	5	7	34	6	11	8
2			5			6		

Șmenul lui Batog - SQRT Decomposition

Împărțim vectorul în zone de lungime L și calculăm minimul pe fiecare zonă în parte.

Linkuri externe:

- Geeks for geeks
- CpAlgorithms

Șmenul lui Batog

Se dă un vector cu n elemente și apoi n operații de genul:

- 1 i j → care este minimul din intervalul [i,j]
- 2 i x → modificați elementul de pe poziția i în x

Cum răspundem la 1 0 8; 1 0 4; **1 1 7** ?

0	1	2	3	4	5	6	7	8
3	9	2	5	7	34	6	11	8
2			5			6		

Șmenul lui Batog

Se dă un vector cu n elemente și apoi n operații de genul:

- 1 i j → care este minimul din intervalul [i,j]
- 2 i x → modificați elementul de pe poziția i în x

Cum răspundem la **1 1 7** ?

0	1	2	3	4	5	6	7	8
3	9	2	5	7	34	6	11	8
2			5			6		

Șmenul lui Batog - Complexitate 1

Pentru **query** (operația de tip 1):

Împărțim vectorul în **n / L zone** de lungime L

Putem itera aproape complet **2 zone** (de la început și/sau de la final) ⇒ O(2*L)

⇒ O(n/L + 2 * L)

0	1	2	3	4	5	6	7	8
3	9	2	5	7	34	6	11	8
2			5			6		

Șmenul lui Batog - Complexitate 1

Pentru **query** (operația de tip 1):

O(n/L + 2 * L)

Cât trebuie să fie L pentru o complexitate minimă?

- L = sqrt(n)

⇒ O(n/sqrt(n) + 2 * sqrt(n))

= O(sqrt(n) + 2 * sqrt(n))

= **O(sqrt(n))**

Șmenul lui Batog

Se dă un vector cu n elemente și apoi n operații de genul:

- 1 i j → care este minimul din intervalul [i,j]
- 2 i x → modificați elementul de pe poziția i în x

Cum răspundem la 2 0 1; **2 3 10** ?

0	1	2	3	4	5	6	7	8
3	9	2	5	7	34	6	11	8
2			5			6		

Șmenul lui Batog

Se dă un vector cu n elemente și apoi n operații de genul:

- 1 i j → care este minimul din intervalul [i,j]
- 2 i x → modificați elementul de pe poziția i în x

Cum răspundem la **2 3 10** ?

0	1	2	3	4	5	6	7	8
3	9	2	~~5~~ 10	7	34	6	11	8
2			7			6		

Șmenul lui Batog - Complexitate 2

Pentru **update** (operația de tip 2):

Modificăm elementul de pe poziția i

Trebuie să facem update pe zona respectivă (să recalculăm minimul)

⇒ O(L) = **O(sqrt(n))**

0	1	2	3	4	5	6	7	8
3	9	2	~~5~~ 10	7	34	6	11	8
2			7			6		

Șmenul lui Batog

Împărțim vectorul în zone de:

- sqrt(n)
- sqrt(n) / 2
- sqrt(n) * 2
- Variațiuni

- **De ce?**
  - În practică, algoritmul poate rula mai rapid pentru valori diferite de sqrt(n), în funcție de operațiile care se fac pe segmente.

Șmenul lui Batog - sortare

Se dă un vector cu n elemente. Sortați-l folosind șmenul lui Batog.

0	1	2	3	4	5	6	7	8
3	9	2	5	7	34	6	11	8
2			5			6		

https://leetcode.com/problems/sort-an-array/

Complexitate?

- O(n sqrt n)

Final