M&I SYSTEMS GROUP

VOJVODINA ICT CLUSTER

Algoritmi i strukture podataka

GO4Code

Marko Cvijanović
Mihailo Bubnjević

Šta su strukture podataka?

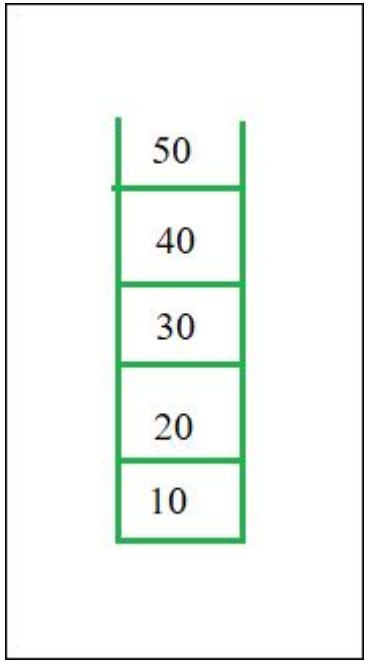
- Strukture podataka su način na koji organizujemo i čuvamo skupove podataka tako da ih možemo koristiti efikasno
- Korišćenjem ispravnih struktura podataka za određeni problem poboljšavamo performanse našeg sistema ili algoritma
- Način na koji radimo sa bilo kojom strukturom podataka je definisana **apstraktnim tipom podataka**

Apstraktni tip podataka (Abstract data type)

- Definiše skup operacija koju određena struktura podataka mora da poseduje
- Ne definiše konkretan način kako će ta struktura podataka biti implementirana unutar nekog programskog jezika
- Na primer, "List" predstavlja apstrakciju koja je unutar C#-a implementirana putem dinamičkih nizova
- Često korišćeni ATP: List, Stack, Queue, Dictionary

Stack kao apstraktni tip podataka

- Možemo pristupiti samo elementu na vrhu stack-a
- LIFO struktura (last in first out)
- Metode koje Stack struktura podataka mora implementirati:
  - push()
  - pop()
  - peek()
  - size()
  - isEmpty()
  - isFull()



Big O notacija

- Definiše način za merenje efikasnosti algoritama
- Koristi se kao metrika za vreme izvršavanja (time complexity) kao i memorijske zahteve (space complexity) algoritma
- Govori nam kako će skalirati naše rešenje kada se poveća input
- Daje gornju granicu kompleksnosti tako što razmatra najgori slučaj za izvršenje algoritma

O(1) - konstantno vreme

- Vreme izvršavanje nije zavisno od veličine input-a

```
public void FirstInArray(int[] intArray)
{
    Console.WriteLine("First element in array is: {0}", intArray[0]);
}
```

O(n) - linearno vreme

- Vreme izvršavanje linearno zavisi od veličine input-a
- Ako povećamo input 10 puta, količina posla će biti 10 puta veća
- Standardan primer je bilo koja petlja koja iterira kroz input

```
public void IterateOverArray(int[] intArray)
{
    var n = intArray.Length;

    for (int i = 0; i < n; i++)
    {
            Console.WriteLine("Element at index {0} is {1}", i, intArray[i]);
    }
}
```

O(n^2) - kvadratno vreme

- Vreme izvršavanje zavisi od kvadrata veličine inputa
- Ako povećamo input 10 puta, količina posla će biti 100 puta veća
- Standardan primer su ugnježdene petlje

```
public void IterateOverMatrix(int[][] intArray)
{
    var n = intArray[0].Length;

    for (int i = 0; i < n; i++)
    {
        for(int j = 0; j < n; j++)
        {
            Console.WriteLine("Element at index {0},{1} is {2}", i, j, intArray[i][j]);
        }
    }
}
```

O(log n) - logaritamsko vreme

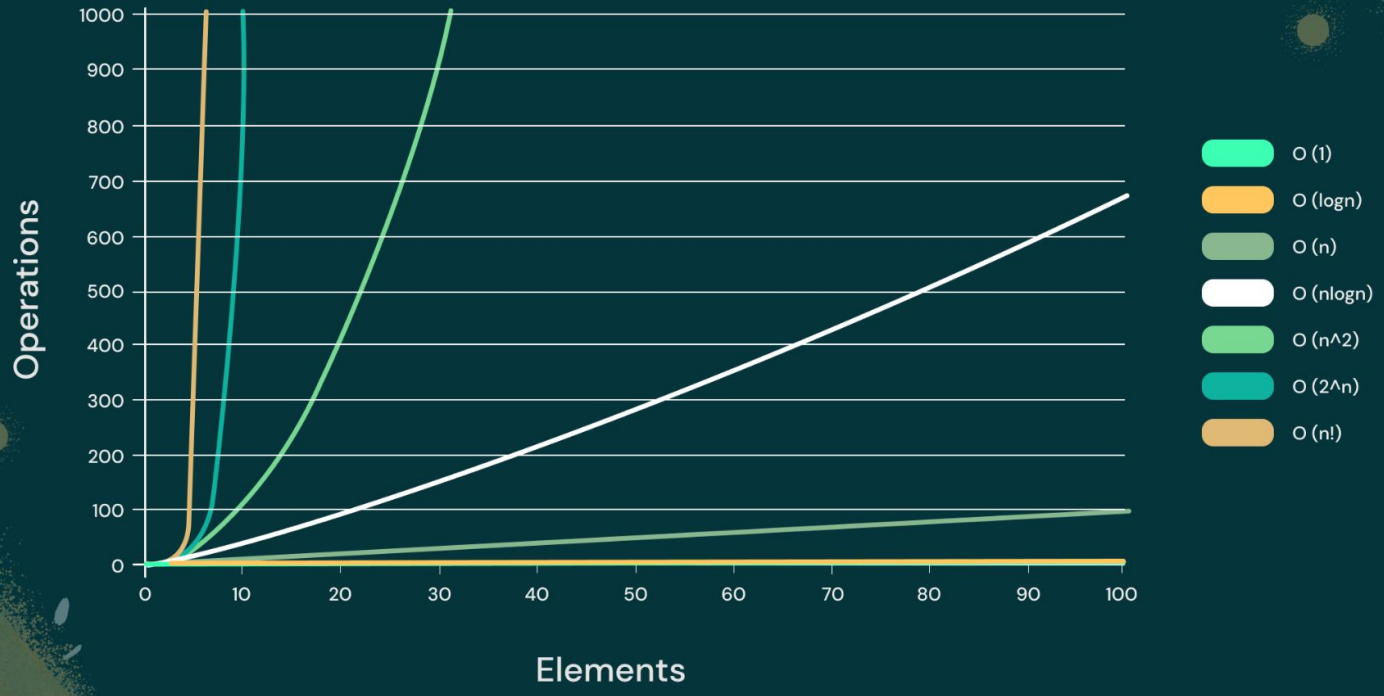
- Logaritamsko vreme izvršavanja možemo tipično videti u algoritmima u kojima se količina podataka koju obrađujemo smanjuje za neki faktor u svakom koraku
- Primetno brže od linearnog vremena
- Standardan primer je binarna pretraga

O(2^n) - eksponencijalno vreme

- Dodavanjem svakog elementa u input, vreme izvršavanja se duplira
- Veoma spori algoritmi, obično želimo da izbegnemo ovu vremensku kompleksnost
- Primer - funkcija koja pronalazi n-ti Fibonačijev broj uz pomoć rekurzije

```
public int Fibonacci(int n)
{
    if (n <= 2)
        return 1;
    else
        return Fibonacci(n - 2) + Fibonacci(n - 1);
}
```

Big-O Complexity



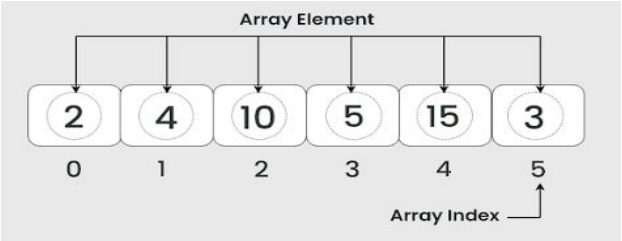
Strukture podataka - metrike

- Najčešće operacije za koje razmatramo performanse neke strukture podataka su:
  - Access - pronalazak vrednosti nekog elementa unutar strukture, obično po index-u, ključu ili poziciji
  - Search - traženje elementa unutar strukture, kako bismo utvrdili njegovo postojanje ili lokaciju
  - Insert - dodavanje elementa u strukturu
  - Delete - brisanje elementa iz strukture

Strukture podataka - Array

- Mogu biti statični ili dinamični u zavisnosti od toga da li su fiksne veličine
- Obično zauzimaju uzastopne lokacije unutar memorije
- Elementi su indeksovani

- Vremenske kompleksnosti operacija:
  - Access: O(1)
  - Search: O(n)
  - Insert: O(n)
  - Delete: O(n)

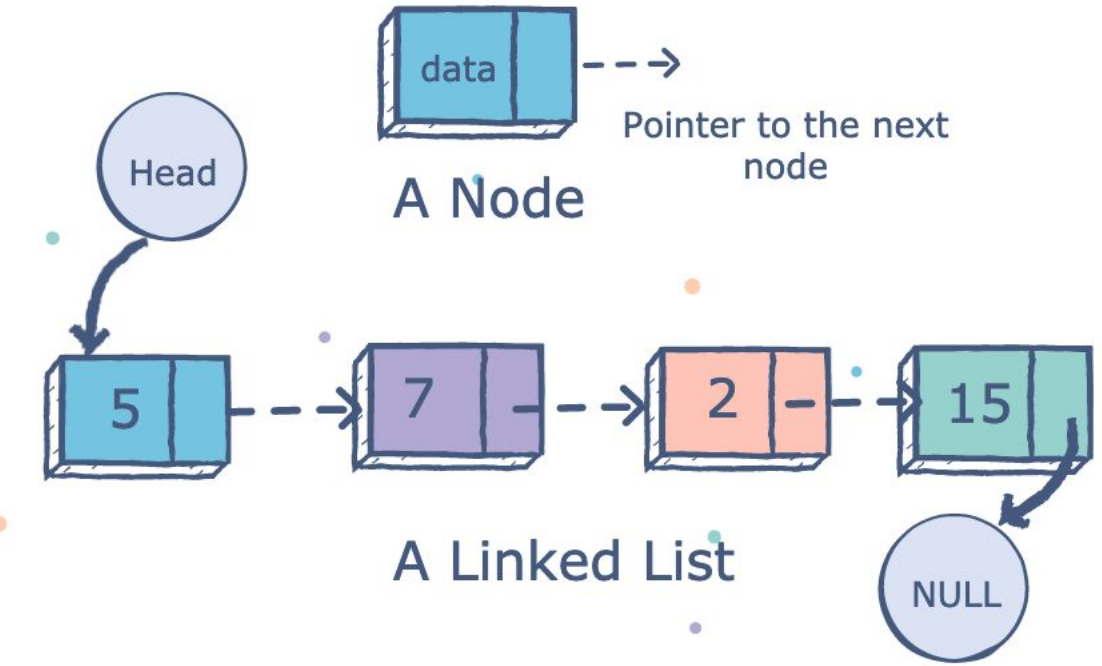


- List<T> u C#-u

Strukture podataka - Linked List

- Elementi su povezani pokazivačima
- Svaki element sadrži referencu na sledeći
- Mogu biti jednostruko ili dvostruko povezani
- Vremenske kompleksnosti operacija:
  - Access: O(n) - moramo iterirati kroz strukturu, nije indeksovana
  - Search: O(n)
  - Insert: O(1) - na početak ili kraj; O(n) inače
  - Delete: O(1) - ako već imamo element koji brišemo; O(n) inače, jer ga moramo prvo pronaći

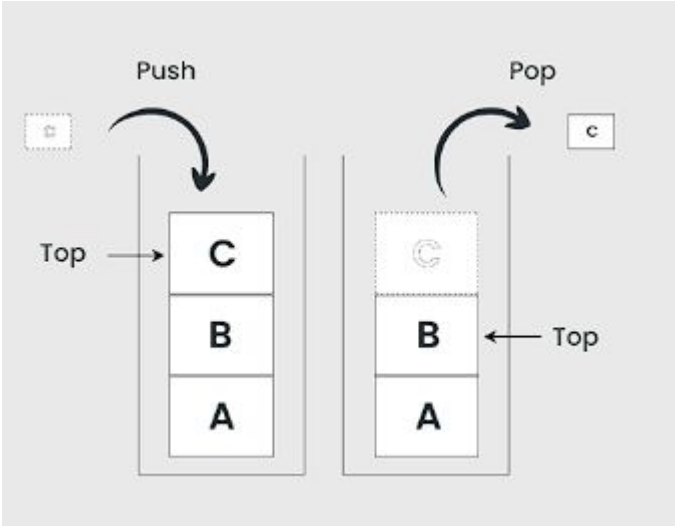
Strukture podataka - Linked List



Strukture podataka - Stack

- LIFO (last in first out) struktura
- Dozvoljava samo čitanje i pisanje gornjeg (poslednjeg) elementa

- Vremenske kompleksnosti operacija:
  - Access: O(1) za poslednji dodat (peek); O(n) inače
  - Search: O(n)
  - Insert: O(1) - push
  - Delete: O(1) - pop



Strukture podataka - Queue

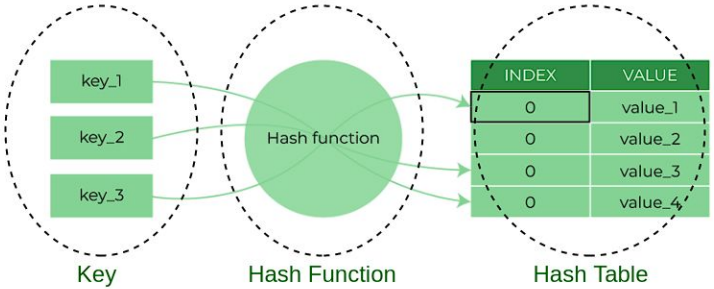
- FIFO (first in first out) struktura
- Dozvoljava čitanje prvog elementa iz strukture, i dodavanje na kraj

- Vremenske kompleksnosti operacija:
  - Access: O(1) za prvi element; O(n) inače
  - Search: O(n)
  - Insert: O(1) - enqueue
  - Delete: O(1) - dequeue

Strukture podataka - Hash table

- Implementacija ATP Dictionary (asocijativni niz) - vrednosti su u parovima ključ vrednost

- Vremenske kompleksnosti operacija:
  - Access: O(1) amortizovano; O(n) u najgorem slučaju
  - Search: O(1) amortizovano; O(n) u najgorem slučaju
  - Insert: O(1) amortizovano; O(n) u najgorem slučaju
  - Delete: O(1) amortizovano; O(n) u najgorem slučaju



Strukture podataka - Hash table