Problema spectacolelor cu Greedy

**Enunt:** Se organizeaza mai multe spectacole in aceasi zi. Fiecare spectacol avand o ora de inceput si unul de sfarsit. Sa se scrie un program care, prin metoda Greedy, sa ordoneze spectacolele astfel inca sa se poata organiza un numarul maxim de spectacole fara suprapuneri.

**Solutie:**

1. Acest pas 0 il vom face doar pentru noi si este total optional dar ne va usura treaba. Cream un tip de date pentru a putea tine codul curat si organizat.

struct Spectacol

{

public string nume;

public int inceput;

public int sfarsit;

public Spectacol(string \_nume, int \_inceput, int \_sfarsit)

{

this.nume = \_nume;

this.inceput = \_inceput;

this.sfarsit = \_sfarsit;

}

}

Odata creat vom putea sa ne declaram lista de spectacole intr-un mod mai elegant.

static void Main(string[] args)

{

Spectacol[] spectacole = {

new Spectacol("A1",8, 10),

new Spectacol("A2",10, 11),

new Spectacol("A3",9, 13),

new Spectacol("A4",12, 13),

new Spectacol("A5",14, 16),

new Spectacol("A6",17, 19)

};

}

1. Ordonam **crescatoar** tabloul nostrum cu spectacole, in functie de ora de **sfarsit.**

Pentru ca in momentul de fata algoritmul de sortare este irelavant, vom presupune ca exista undeva o metoda Ordoneaza, care ne va rezolva acest pas.

Ordoneaza(spectacole);

1. Suntem gata sa obtinem raspunsul la aceasta problema. Fiindca nu stim exact numarul de spectacole pe care le putem organiza, vom folosi un tablou dynamic, numit si Lista.

List<Spectacol> raspuns = new List<Spectacol>();

1. Pentru vectorul nostru cu spectacole este pregatit deja ( fiindca la pasul 1 a fost ordonat ). Putem stii sigur ca primul element este unul bun.

raspuns.Add(spectacole[0]);

1. Mergand mai departe trebuie sa gasim urmatorul spectacol care se poate organiza.

Este organizabil daca incepe exact cand sau dupa ce s-a terminat ultimul spectacol. Pentru a putea tine evidenta mai usor, vom introduce o variabila care sa retina ultimul spectacol organizat. Astfel codul de la pasul 3 se va schimba in:

Spectacol ultimul = spectacole[0];

raspuns.Add(ultimul);

1. Din acest moment avem tot ce ne trebuie sa construim raspunsul complet.

for(int i = 1; i < spectacole.Length; i++)

if( ultimul.sfarsit <= spectacole[i].inceput )

{

ultimul = spectacole[i];

raspuns.Add(ultimul);

}

1. In acest pas deja am obtinut lista de spectacole care se pot organiza. Pentru verificare vom afisa pe ecran.

foreach(Spectacol spectacol in raspuns)

Console.Write(spectacol.nume+" ");

**Codul integral:**

struct Spectacol

{

public string nume;

public int inceput;

public int sfarsit;

public Spectacol(string \_nume, int \_inceput, int \_sfarsit)

{

this.nume = \_nume;

this.inceput = \_inceput;

this.sfarsit = \_sfarsit;

}

}

static void Main(string[] args)

{

Spectacol[] spectacole = {

new Spectacol("A1",8, 10),

new Spectacol("A2",10, 11),

new Spectacol("A3",9, 13),

new Spectacol("A4",12, 13),

new Spectacol("A5",14, 16),

new Spectacol("A6",17, 19)

};

Ordoneaza(spectacole);

List<Spectacol> raspuns = new List<Spectacol>();

Spectacol ultimul = spectacole[0];

raspuns.Add(ultimul);

for(int i = 1; i < spectacole.Length; i++)

{

if( ultimul.sfarsit <= spectacole[i].inceput )

{

ultimul = spectacole[i];

raspuns.Add(ultimul);

}

}

foreach(Spectacol spectacol in raspuns)

Console.Write(spectacol.nume+" ");

}