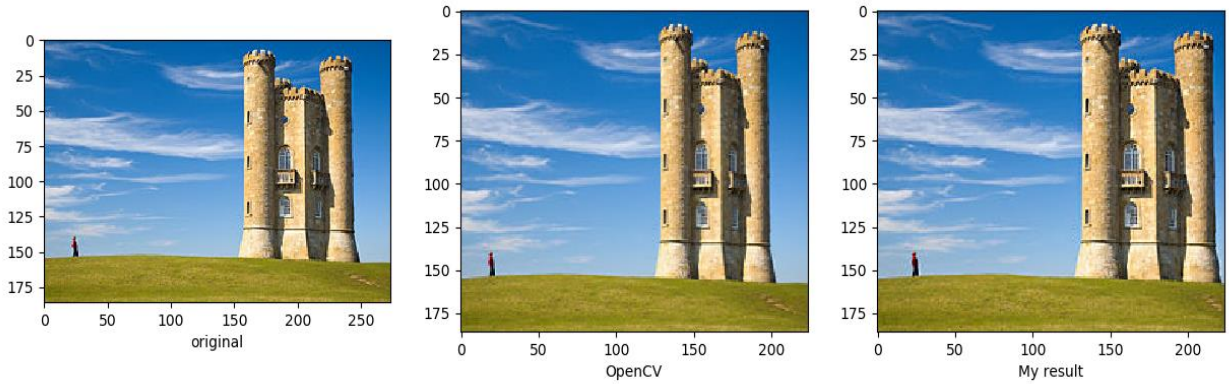
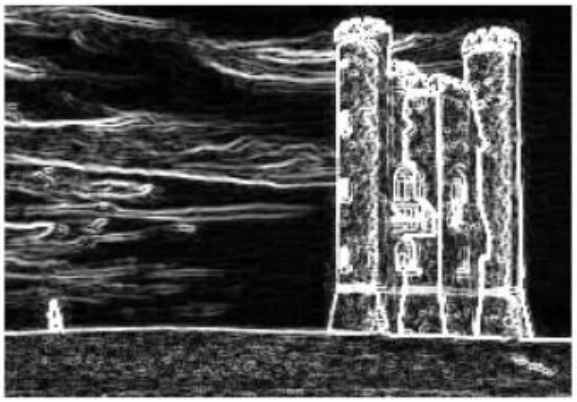
Candrea Laurențiu-Vasile
Grupa 332

Concepte și Aplicații în Vederea Artificială - Tema 2
Redimensionarea imaginilor cu păstrarea conținutului

1.1 Micșorarea imaginii pe lățime:

Imagine: castel.jpg
Metoda: Programare Dinamica
Dimensiune: reducere 50 pixeli

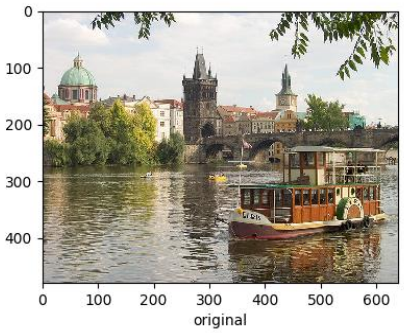


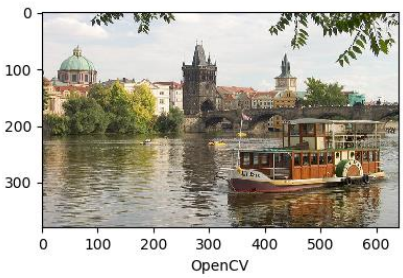


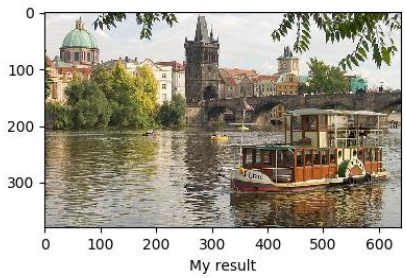
Avand matricea de “energie” putem observa ușor care “drumuri” ar trebui eliminate.

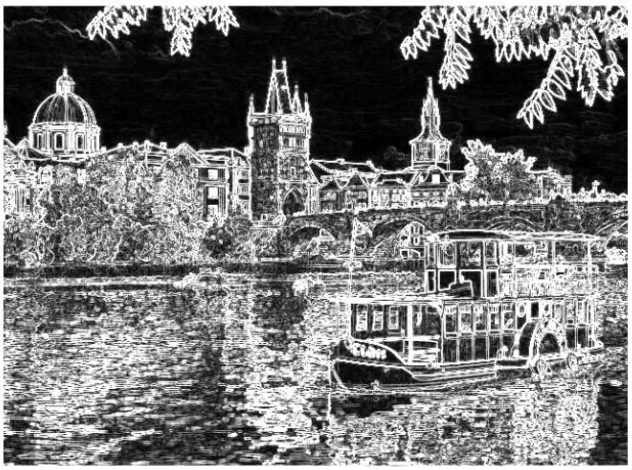
1.2 Micșorarea imaginii pe înălțime:

Imagine: praga.jpg
Metoda: Programare Dinamica
Dimensiune: reducere 100 pixeli





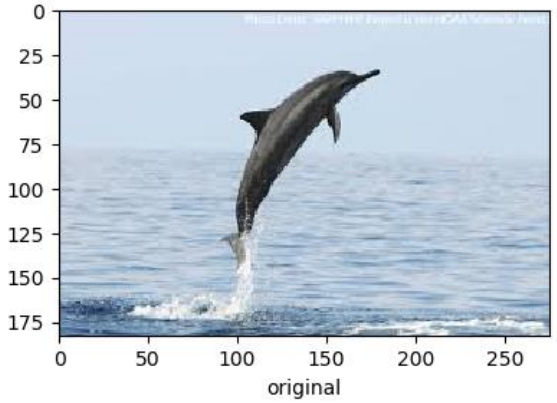


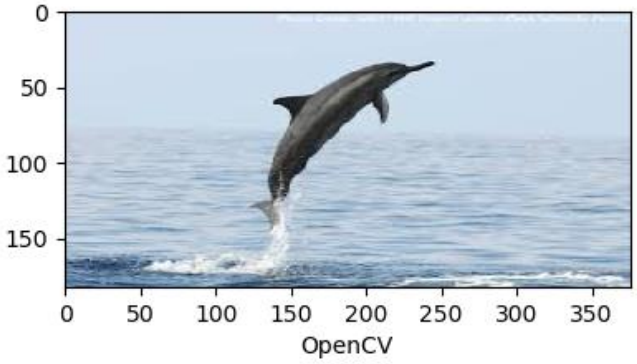


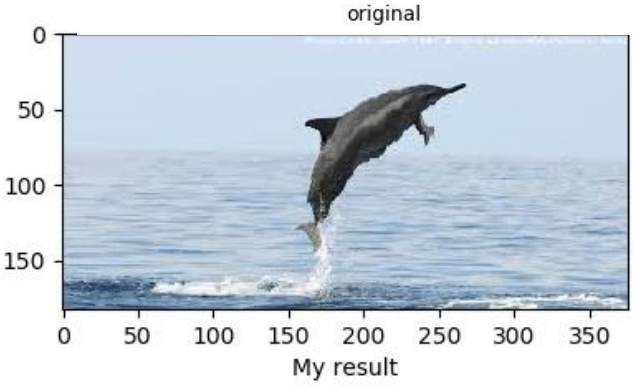
1.3 Mărirea imaginilor:

1.3.1 Mărirea pe lățime:

Imagine: delfin.jpeg
Metoda: Programare Dinamica
Dimensiune: mărire 50 pixeli

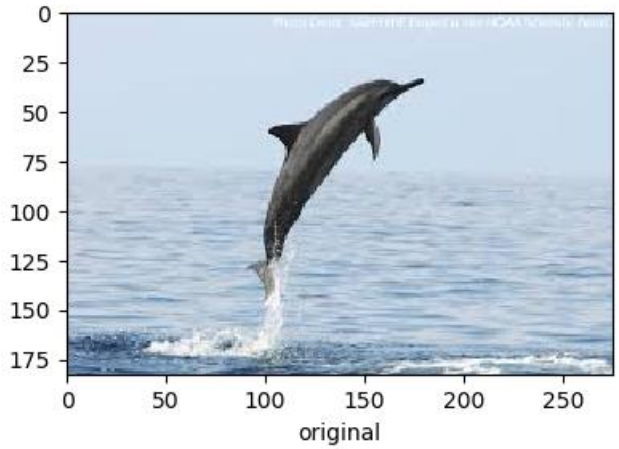


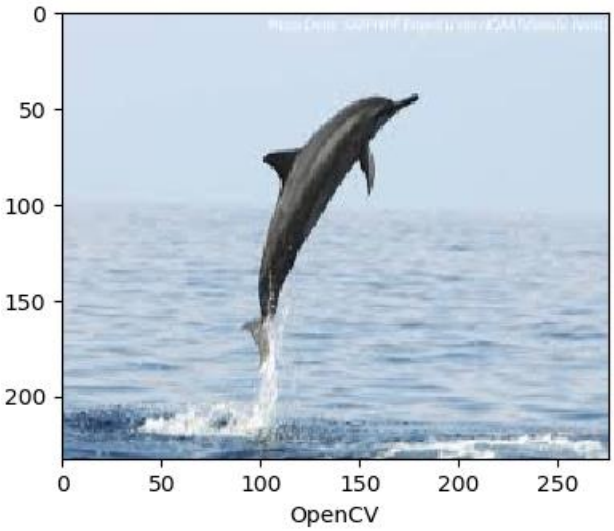


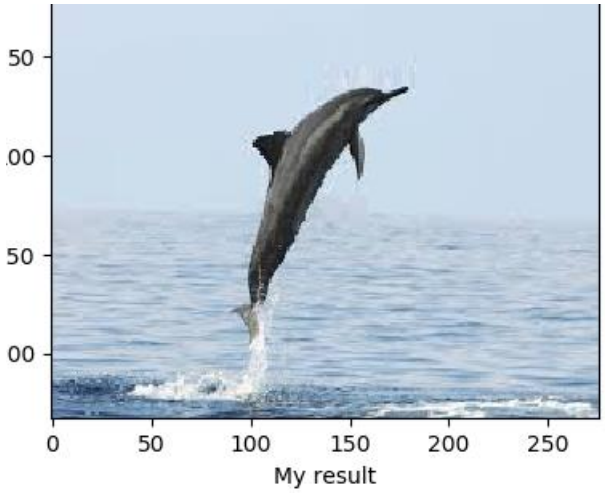


1.3.2 Mărirea pe Înălțime:

Imagine: delfin.jpeg
Metoda: Programare Dinamica
Dimensiune: mărire 50 pixeli







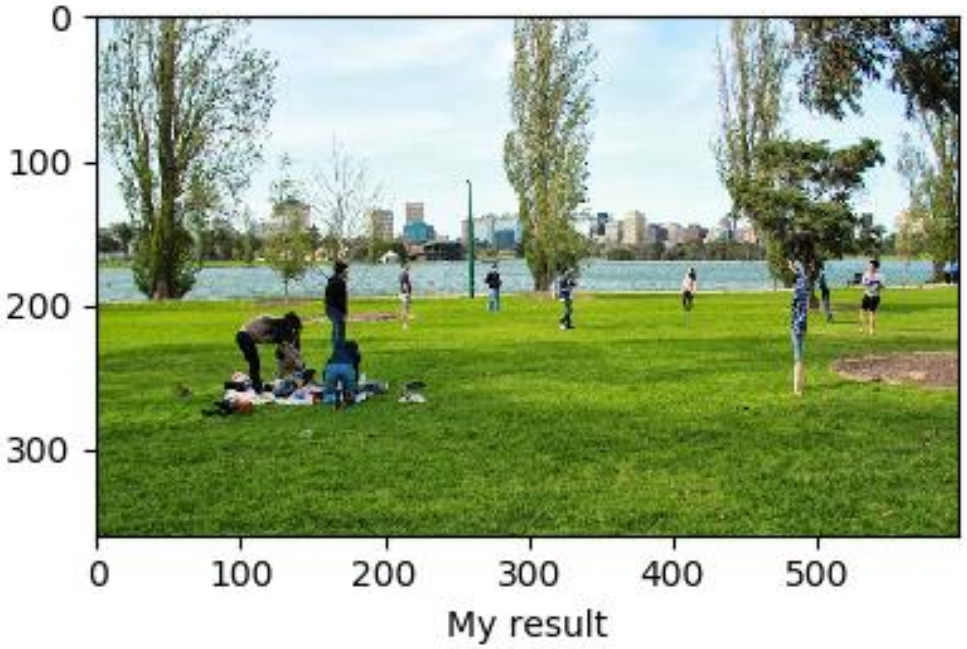
Aici am avut noroc deoarece sunt puține „drumuri” de cost minim care ating delfinul prin urmare nu este „stricat” la fel de mult ca in 1.3.1. Problema este aceeași.

1.5 Eliminarea obiectelor din imagini:

Imagine: lac.jpeg
Metoda: Programare Dinamica
Dimensiune: dimensiune selecție

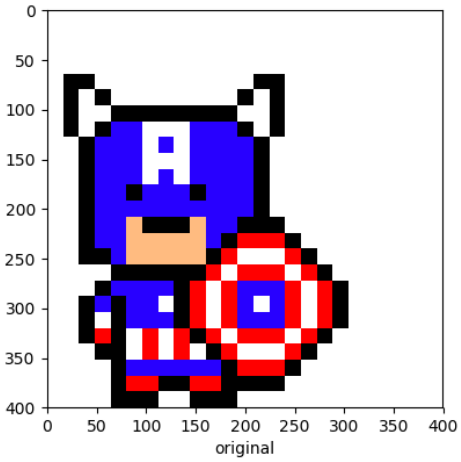
Algoritmul este unul simplu: dau valori negative in chenarul selectat de către utilizator apoi folosesc funcia de ștergere pe lățime în funcție de lățimea chenarului. Cum se poate vedea și în figurile alăturate, eliminarea nu se face corect.

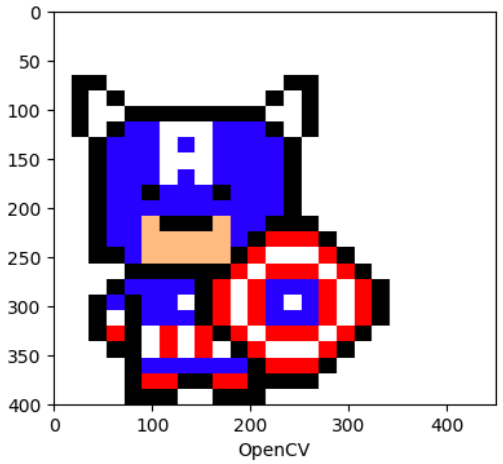
Motivul este deplasarea indicilor pentru fiecare drum șters.

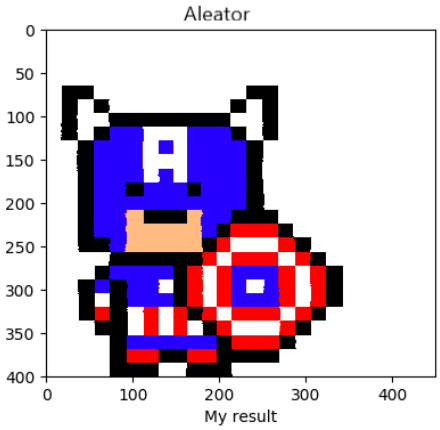


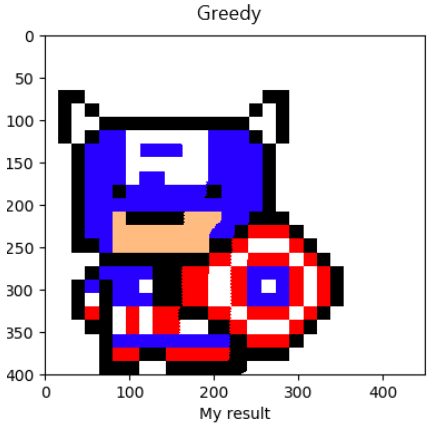
1.6 Exemple noi:

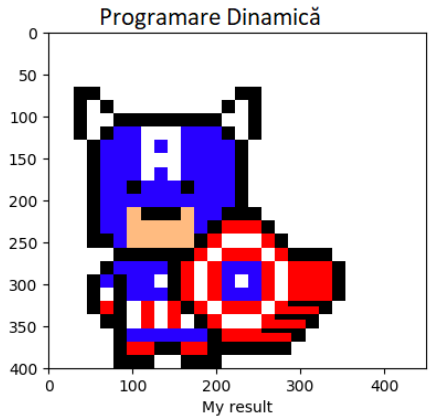
1. **Mărire pe lățime:**
   Imagine: capitan.jpg
   Metoda: Random, Greedy, Programare dinamică
   Dimensiune: 50 pixeli











Se poate observa ușor că varianta mea nu este una bună. Problema vine de la indicii drumurilor de cost minim care se modifica la fiecare pas. Actualizarea mea este una greșită, motiv pentru care nu se multiplică drumurile „bune”.

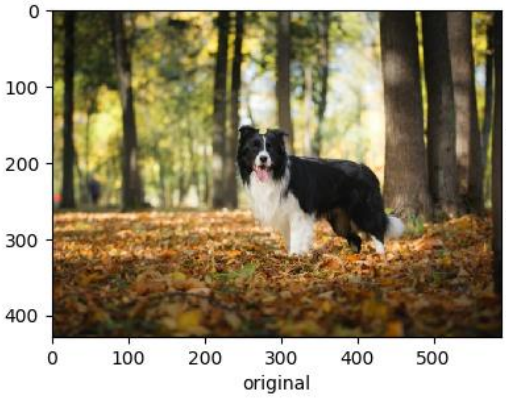
2. **Micșorarea imaginii pe lățime:**

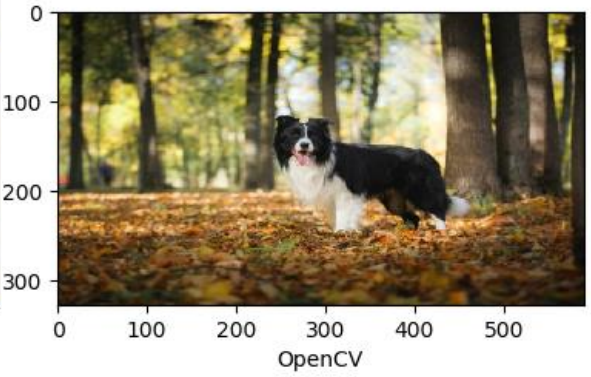
   Imagine: catMeme.jpg
   Metoda: Random, Greedy, Programare dinamică
   Dimensiune: 50 pixeli

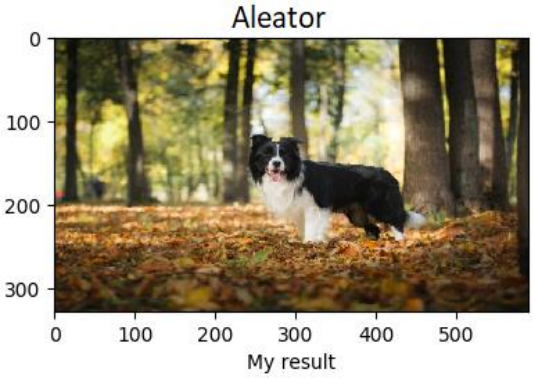
Algoritmul merge bine pe partea “Greedy” si “Programare Dinamică”, se evită linia albă unde este o diferență clară de intensitate, se iau drumuri cu cost final scăzut. Bineînțeles, partea “Aleator” se comportă în mod neașteptat.

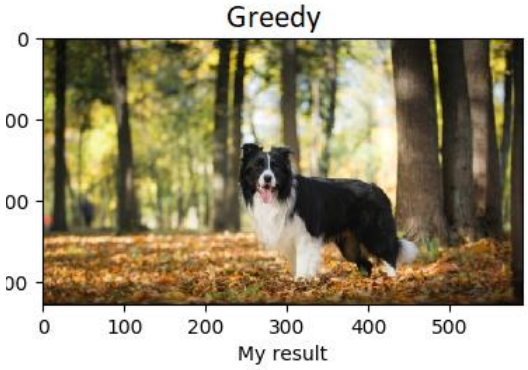
3. **Micșorarea imaginii pe înălțime:**

   Imagine: dog.jpg
   Metoda: Random, Greedy, Programare dinamică
   Dimensiune: 100 pixeli





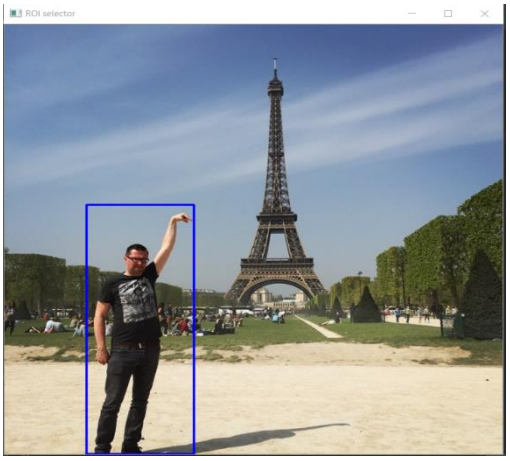


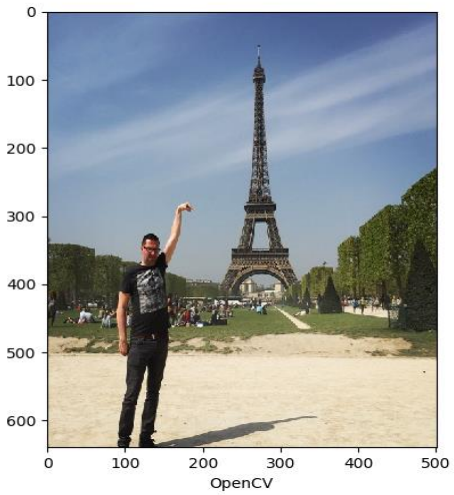


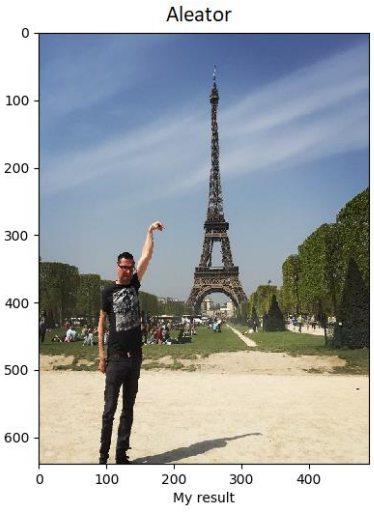
Algoritmul funcționează foarte bine cu metoda “Programare Dinamică” și “Greedy”, câinele fiind “tăiat” foarte puțin. Diferența de contraste ajută la evidențierea câinelui.

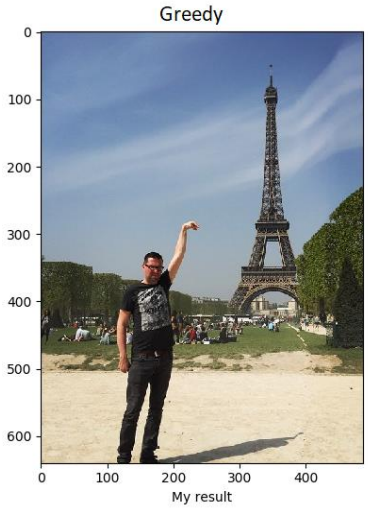
4. Eliminarea unui obiect

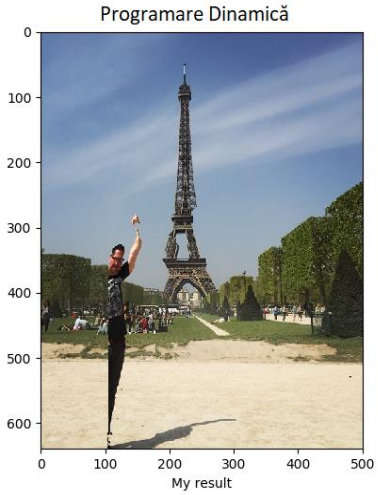
Imagine: ifell.jpg
Metoda: Random, Greedy, Programare dinamică
Dimensiune: dimensiune selecție







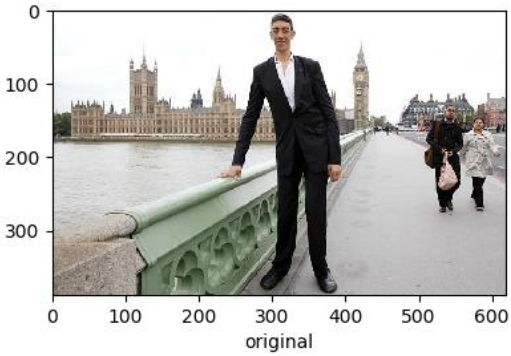


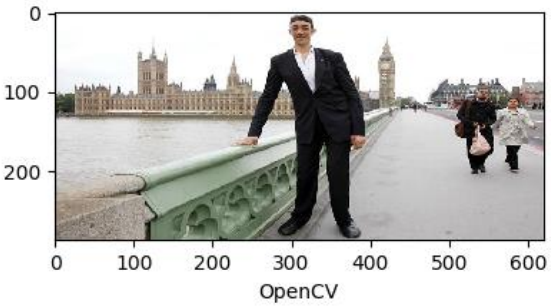


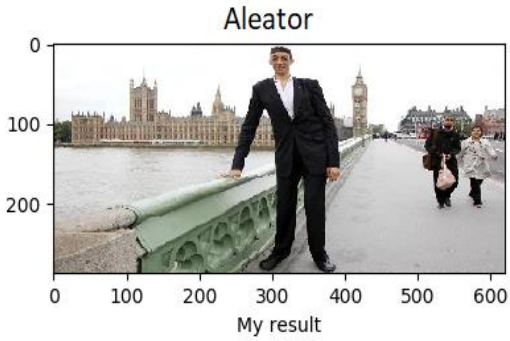
Cum am menționat si mai sus, funcția nu tratează cazul de modificare al indicilor. Algoritmul șterge liniile cu pozițiile primite inițial și nu se actualizează

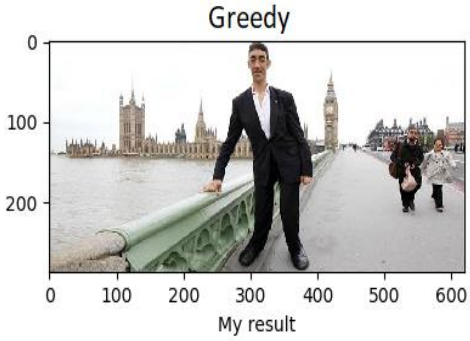
5. **Marirea imaginii pe înălțime:**

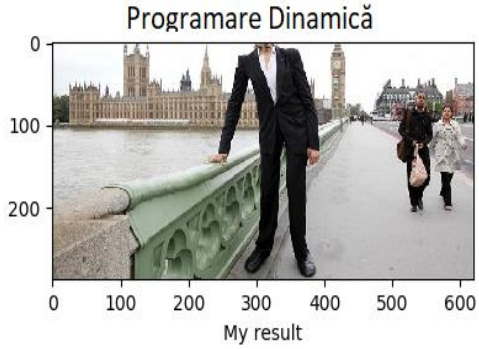
   Imagine: tall.jpg
   Metoda: Random, Greedy, Programare dinamică
   Dimensiune: 100 pixeli











Un exemplu bun în care algoritmul merge bine însă nu obținem ce ne dorim. Din cauza luminozității deschise ale feței și a peisajului, o buna parte din drumurile de cost minim trec prin fața domnului din imagine și astfel algoritmul dinami o va șterge. Surprinzător, varianta “Aleator” are cel mai bun rezultat în acest caz.