# Tema 1

**Vlădescu Cristiana**

Inteligență Artificială

—

342 C4

—

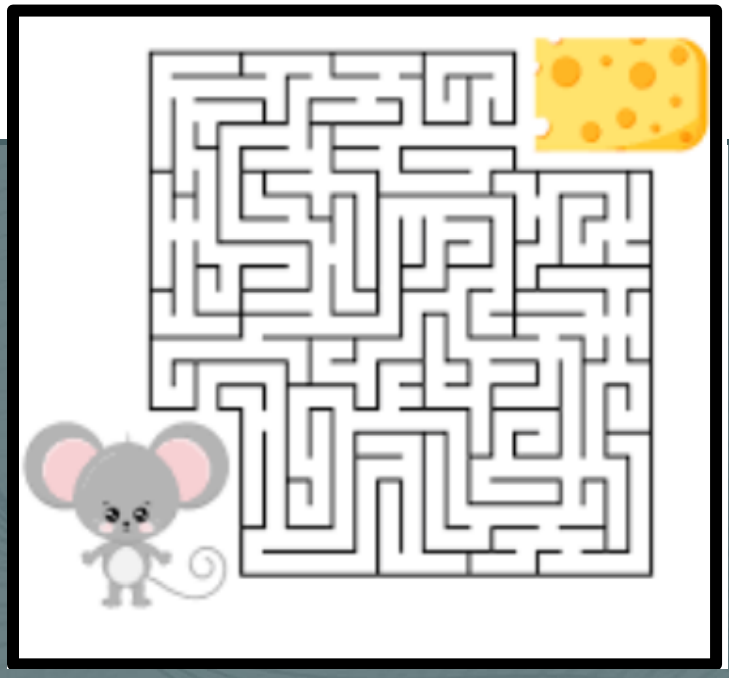
15 noiembrie 2020

## Prezentarea temei

Pentru rularea temei, este necesară existenţa următoarelor module python:

- pickle
- matplotlib
- heapq

Tema poate fi rulată prin executarea comenzii “python3 tema1.py”, iar selectarea fişierelui de input şi a algoritmului folosit se face din fişierul “tema1.py”. După rularea programului, acesta va prezenta spre output costul drumului, calea obţinută şi timpul de rulare.



# Prezentarea algoritmilor folosiți

## REALIZAREA IMPLEMENTĂRII

Pentru a rezolva cerinţele date, am folosit pseudocodul oferit în cadrul cursului, pe care l-am optimizat pentru a reduce timpul de rulare al acestora.

În implementarea *Depth First Iterative Deepening*, *Iterative Deepening A** şi a *euristicii* alese, am utilizat două structuri auxiliare, de tip listă, pentru a memora nodurile deja vizitate, dar şi costul cu care s-a ajuns până la acestea, cu scopul de a reduce numărul de expandări de noduri pentru a obţine calea de la sursă la destinaţie.

În implementarea tehnicii *Branch and Bound*, am utilizat algoritmul *A** realizat în cadrul laboratorului 1 de IA, întrucât reprezintă un exemplu clasic al acestei tehnici, conform site-ului *sciencedirect[1]* şi a unui articol aparţinând *GWU SEAS[2]*.

[1]:https://www.sciencedirect.com/topics/computer-science/branch-and-bound-algorithm-design
[2]:https://www2.seas.gwu.edu/~ayoussef/cs6212/branchandbound.html

## COMPARAREA METODELOR

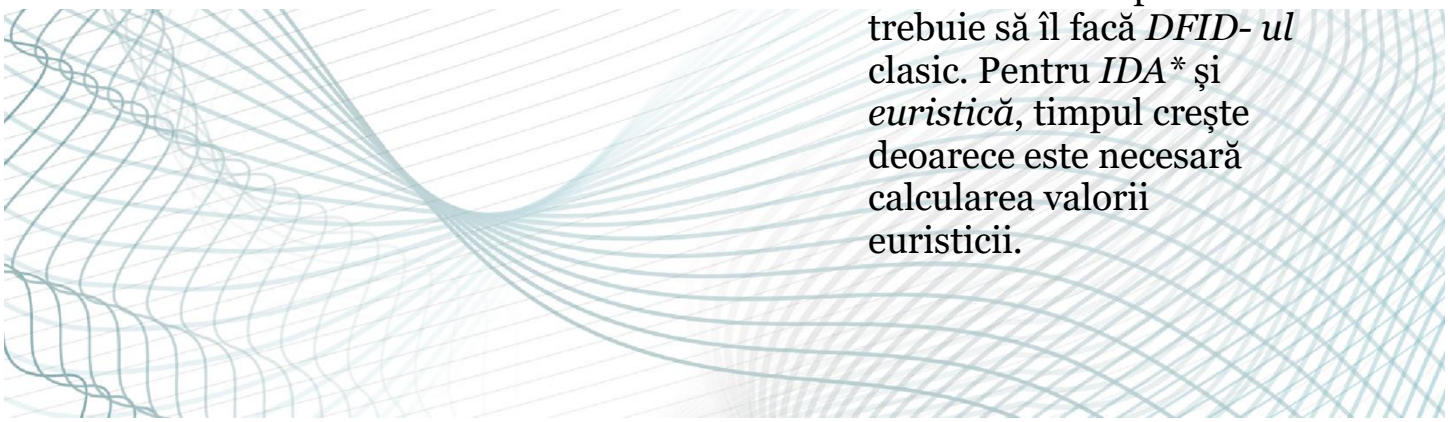
Analizând costul obţinut pentru fiecare fişier de input în parte, se poate observa că fiecare dintre cei 5 algoritmi implementaţi întoarce acelaşi răspuns:

- pentru *input1* : costul minim obţinut este 18.
- pentru *input2* : costul minim obţinut este 89.
- pentru *input3*: costul minim obţinut este 139.

Chiar dacă costurile sunt identice, ceea ce denotă particularitatea fiecărui algoritm este calea aleasă de acesta. Astfel, pe aceste trei inputuri a fost posibilă obţinerea unor căi diferite, în funcţie de modul în care este tratată problema, dar şi în funcţie de euristica aleasă şi de costul întors de aceasta.

Din punct de vedere temporal, se observă că unele euristici au mai mult succes în găsirea rapidă a unei căi, dar şi că unele metode tind să ajungă mai repede la rezultat decât altele. Astfel că:

- implementarea *A*,* specific *Branch and Bound,* este cea mai rapidă modalitate de a găsi calea de la sursă la destinaţie, cu un timp de 0.0021 secunde pentru cel mai mare spaţiu de căutare.
- implementarea *LRTA** se clasează la mijloc, având în continuare timpi de execuţie reduşi, datorită simplităţii acesteia şi a operaţiilor simple.
- implementarile pentru *DFID, IDA** şi *euristica proprie* au cele mai slabe rezultate, datorate în primul rând numărului mare de calcule pe care trebuie să îl facă *DFID- ul* clasic. Pentru *IDA** şi *euristică*, timpul creşte deoarece este necesară calcularea valorii euristicii.



## COMPARAREA EURISTICILOR

Euristica data în enunţul temei, *Distanţa euclidiană,* are avantajul că se poate muta pe orice direcţie, având mereu cel mai mic cost de deplasare între două puncte.

Dezavantajul acestei euristici este că spaţiul într-un labirint este discret, cu toate coordonatele fiind întregi, iar pe un plan 2D există maximum 8 direcţii în care se poate muta, iar distanţa euclidiană nu ţine cont de acest lucru.

În plus, în vederea calculării acesteia, este necesară calcularea radicalului, ce poate conduce la calcule mai costisitoare din punct de vedere temporal.

Euristica aleasă de mine, *Distanţa pe diagonală,* are avantajul de a se putea muta pe fiecare dintre cele 8 posibile poziţii viitoare, iar folosind coeficienţii D şi D2 egali, de valoare 1, costul se calculează mult mai uşor decât în cazul distanţei euclidiene.

În cazul alegerii acestor coeficienţi, distanţa pe diagonală corespunde *Distanţei Chebyshev*.

# Timpi obținuți la rularea algoritmilor

## REPREZENTARE TABELARĂ

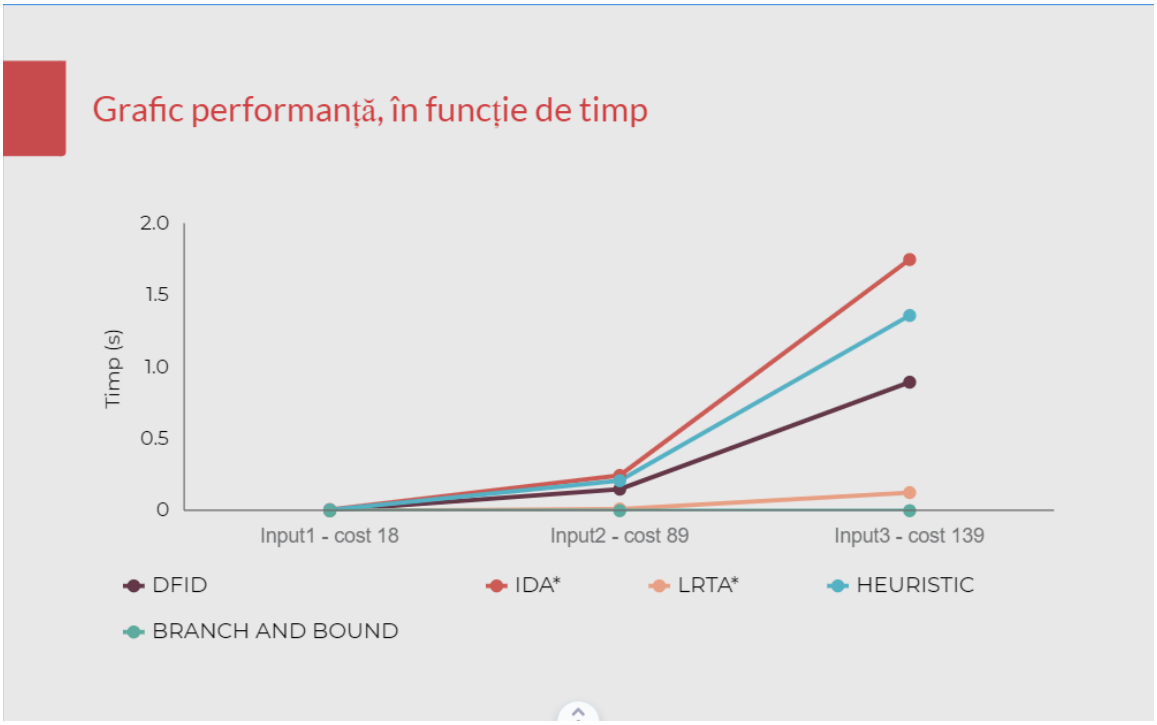
În tabelul de mai jos se pot observa diferenţele de timp necesar rulării şi producerii output-ului.

Se poate remarca faptul că DFID are avantajul de a își termina execuţia mai repede decât IDA* şi euristică aleasă de mine, deoarece nu trebuie să calculeze valoarea euristicii pentru fiecare poziţie ce se încearcă a fi introdusă în cale. Acest lucru se resimte mai ales în cazul inputului 3, unde DFID se termină aproape de două ori mai repede decât celelelalte variante.

Chiar dacă LRTA* se termină mult mai repede decât DFID, necesitatea rulării algoritmului până la stabilizarea căii şi a costului se resimte atunci când este comparat cu varianta de Branch and Bound aleasă, A*, care reuşeşte să se detaşeze complet de restul algoritmilor implementaţi.

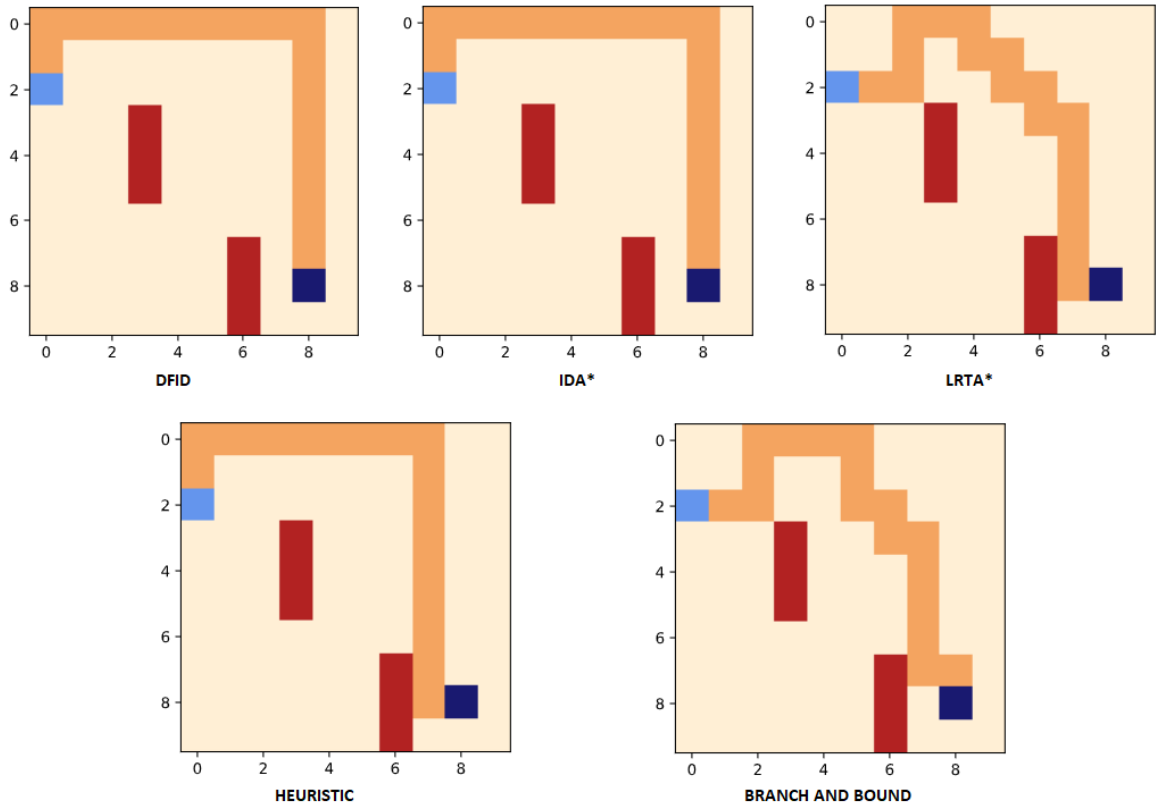
| 1 | | Input1 - cost 18 | Input2 - cost 89 | Input3 - cost 139 |
|---|---|---|---|---|
| 2 | DFID | 0.004 | 0.152 | 0.898 |
| 3 | IDA* | 0.010 | 0.248 | 1.753 |
| 4 | LRTA* | 0.002 | 0.014 | 0.128 |
| 5 | HEURISTIC | 0.008 | 0.211 | 1.363 |
| 6 | BRANCH AND BOUND | 0.0008 | 0.0019 | 0.0021 |

## REPREZENTARE GRAFICĂ

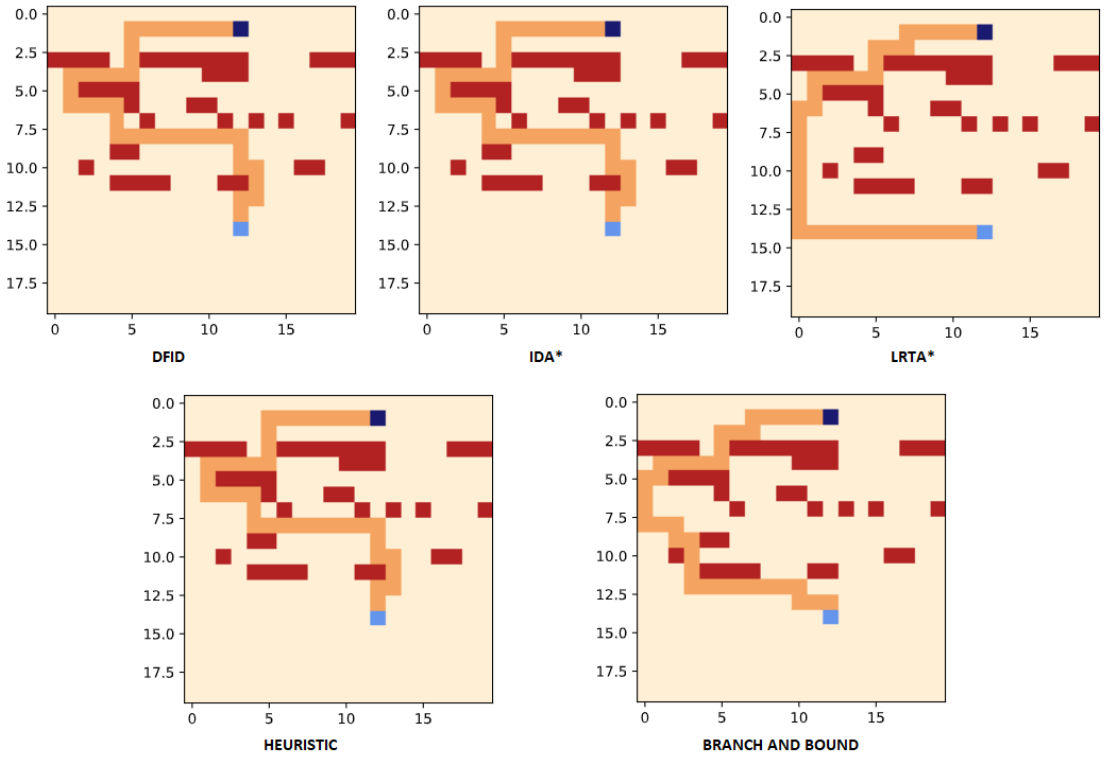



Grafic, diferența de timp dintre algoritmi este foarte ușor de văzut. În timp ce IDA* atinge aproape 2 secunde pentru input3, Branch and Bound nu depașeste pragul de 0.1 secunde.

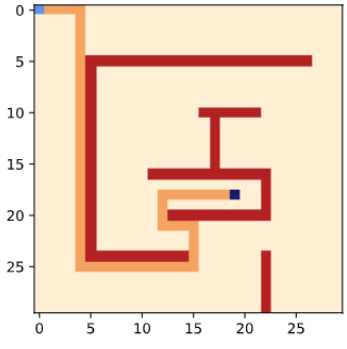
# REPREZENTAREA CĂILOR OBȚINUTE – INPUT 1



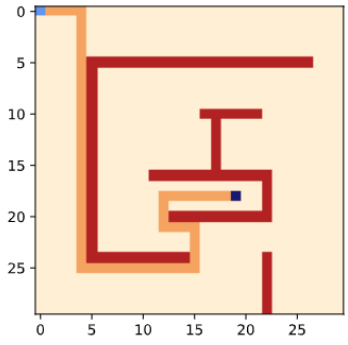
# REPREZENTAREA CĂILOR OBȚINUTE – INPUT 2



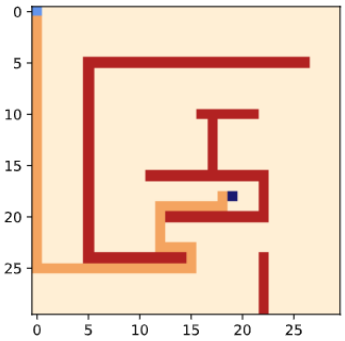
## REPREZENTAREA CĂILOR OBȚINUTE – INPUT 3



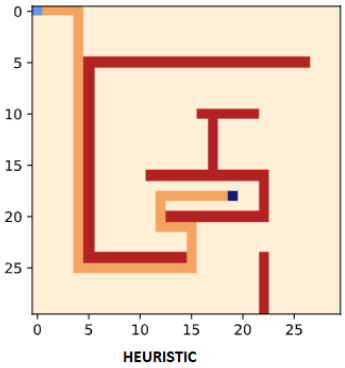

DFID



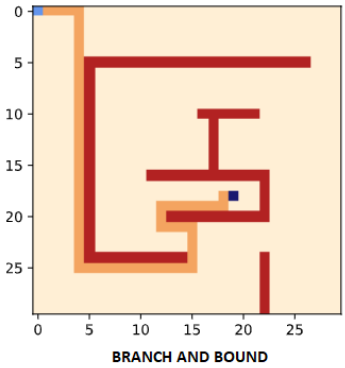

IDA*




LRTA*




HEURISTIC




BRANCH AND BOUND

## INTERPRETAREA CĂILOR OBȚINUTE

În funcţie de euristica aleasă şi de modalitatea de calcul a căii, se observă ca acestea diferă în câteva puncte: desi în mare parte caile sunt asemănătoare, au existat momente în care o mutare a fost mai avantajoasă conform unei euristici, în contradicţie cu alta.

Conform reprezentărilor, toate căile obţinute sunt corecte şi au costuri identice, conform calculelor făcute.

Legendă reprezentare grafică:

- roşu = obstacole
- portocaliu = drum
- albastru = sursă / destinaţie