Univerza *v Ljubljani*
Fakulteta *za računalništvo*
in informatiko

Umetna inteligenca
2. seminarska naloga

Mentor: asist. dr. Petar Vračar in
asist. dr. Tome Eftimov

Datum: 6.1.2023

Avtorja: Urh Jamšek, Blaž Mikec

Uvod

Vizualizacija

Opis algoritmov
- Generiranje množice naslednjih korakov
- BFS
- DFS
- IDDFS
- A*
- IDA*
- Hevristika

Algorithmi na primerih
- Primer 0
- Primer 1
- Primer 2
- Primer 3
- Primer 4
- Primer 5

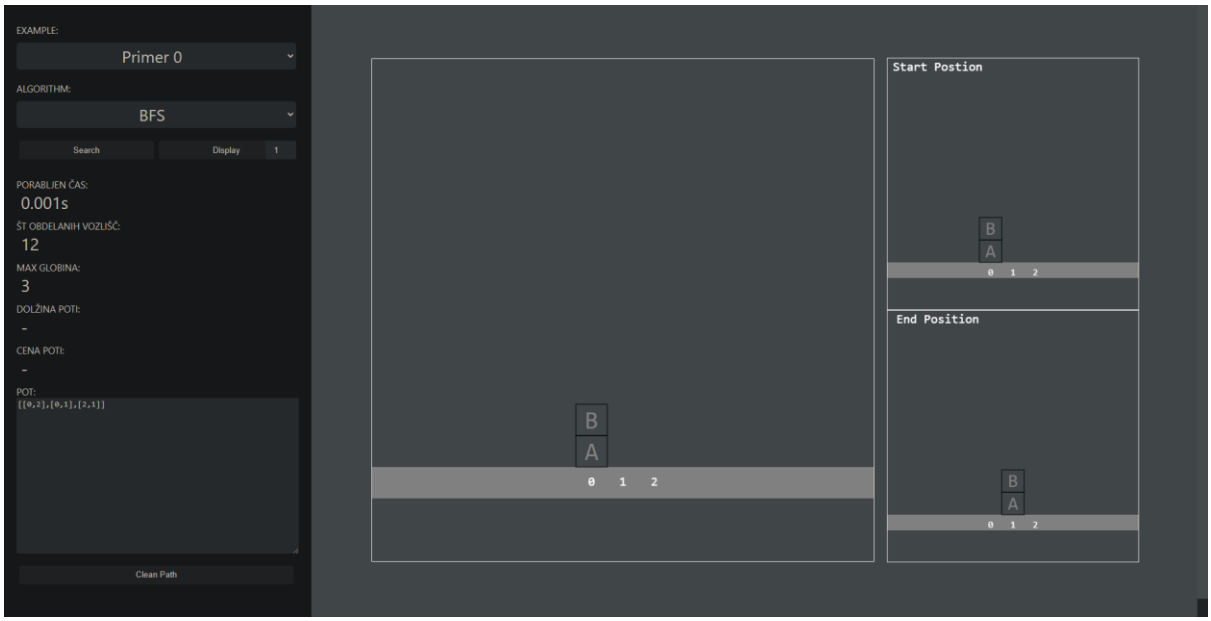
Zaključek

Uvod

Pri drugi seminarski nalogo predmeta umetna inteligenca smo morali implementirati in uporabiti preiskovalne algoritme na domeni robotiziranega skladišča. Skladišče vsebuje škatle, ki so lahko na kupu ali pa na tleh. Robotska roka lahko vzame najbolj zgorno škatlo stolpca ter ga premakne na kateri koli drugi stolpec. Upravljalec robotske roke prejme začetno in končno konfiguracije skladišča, nato pa mora poiskati čim krajpe zaporedje ukazov, s katerimi bo robotska roka preuredila škatle na zahtevani način. Pri tej seminarski nalogi sva implementirala BFS, DFS, IDDFS, A* in IDA*. Odličila sva se izdelati spletno stran ter implementirati iskalne algoritme v programskem jeziku javaScript.

Vizualizacija

Ob začetku te seminarske naloge sva se odločila kreirati spletno stran, saj sva menila da tako najboljše vizualizírati delovanje in prikaz dobljenih rezultatov iskalnih algorithmov.



Na levi strani splene strani se nahaja grafični vmesnik, kjer lahko izbereš primer ter iskalni algorithme. S klikom na gumb Search bo pognal izbrani iskalni algorithem. Obroba gumba Search se lahko obrava zeleno ali rdeče glede na ali je iskalni algorithem našel ustrezno pot iz začetne do končne konfiguracije. Ko se iskalni algorithem uspešno izvede, lahko vidimo najdeno pot levo spodaj. Če želimo prikaz po premikih od začente do končne konfiguracije, lahko pritisnimo gumb Display. Ob gumbu Display imava tudi vrednost ti določa hitrost izrisa. Manjša kot je vrednost hitreje se izrisuje pot. Grafični vmesnik vsebuje tudi nekaj statističnih podatkov kot je čas, ki ga je iskalni algorithem potreboval, da je vrnil rezultat. Pod dobljeno pot pa imava tudi gumb Clear path. Ta gumb pregleda vrnjeno pot ter poišče nepotrebne premike ter jih odstrani tako da optimizira dobljeno pot.

Opis algoritmov

Generiranje množice naslednjih korakov

Vsi sledeči algoritmi so implrementirani tako, da za vsako vozlišče (oz. stanje v skladišču) sproti generiramo množico naslednjih potez. To storimo tako, da z dvema števcema “i” in “j” v intervalu 0 do “število stolpcev v skladišču”, generiramo vsako permutacijo premikov, kjer “i” predstavlja stolpec iz katerega vzamemo škatlo in jo odložimo na j-ti stolpec.
Ker seveda nočemo, da nastanejo napake, lahko že tu izločimo nekatere možnosti:

- če sta “i” in “j” enaka, se premik efektivno ne zgodi,
- če je “i”-ti stolpec prazen,
- če je “j”-ti stolpec poven.

Hkrati pa beleživa že obiskana stanja, kar izloči možnost ciklanja.

BFS

BFS (Breath-First Search, slo.: Iskanje v širino) je neinformirani iskalni algoritem. Kar pomeni, da obišče vsa vozlišča, pri tem pa ne uporablja nobenih dodatnih informacij za izbiranje naslednjega vozlišča. Pri tem algoritmu se premikamo od vozlišča do vozliča po nivojih drevesa in obdelamo vsa vozlišča v nivoju preden se pomaknemo na naslednjega.

DFS

DFS (Depth-First Search, slo.: Iskanje v globino) je iskalni algoritem, pri katerem se najprej poglabljamo v posameznega otroka vozlišča, ga rekurzivno obdelamo in se postopoma vračamo nazaj in obdelujemo preostale otroke. To pomeni, da bomo najprej obdelali vse vozlišče v globino, preden se vrnemo na svojo raven in obdelamo še preostale vozlišče na tej ravni. DFS je primeren za reševanje problemov, kjer je cilj najti kakšno pot v grafu ali drevesu. Algoritem ne nujno najde optimalne rešitve.

IDDFS

IDDFS (Itertive Deppening Depth-First Searh) je varianta DFS (iskanje v gobino) algoritma, ki ga uporabljamo za reševanje problemov. Pri tem algoritmu efektivno večkrat izvedemo iskanje v globino pri čemer vsakič omejimo globino iskanja. Ob primeru, da vozlišče ni bilo najdemo, povečamo omejitev. Algoritem vedno najde optimalno rešitev.

A*

A* (A-star) je informirani iskalni algoritem, pri katerem podamo hevristiko, za izbiranje naslednjega vozlišča. A* vzlišču z najbolje ocenjenim F score-om (hevrisika + cena poti) razvije naslednja vozlišča. To ponavljamo dokler ne najdemo končnega vozlišča. Ta

algoritem je zelo popularen za reševanje problemov z iskanjem poti, kot so navigacije, računalniške igrice, robotika, itd…

IDA*

Algoritem IDA* (Iterative deepening A*) je zelo podoben A*, kateremu se razlikuje v tem, da shranjuje le mejo F-score-a namesto vseh vozlišč, kar je zelo potratno z pomnilnikom. V tem pa je ta algoritem zelo podobem IDDFS, ki tudi uporablja mejo. Ta algoritem je učinkovit za reševanje problemov, ki zahtevajo globoko preiskovanje grafa, saj porabi manj pomnilnika. Je pa kljub temu manj učinkovit kot A*.

Hevristika

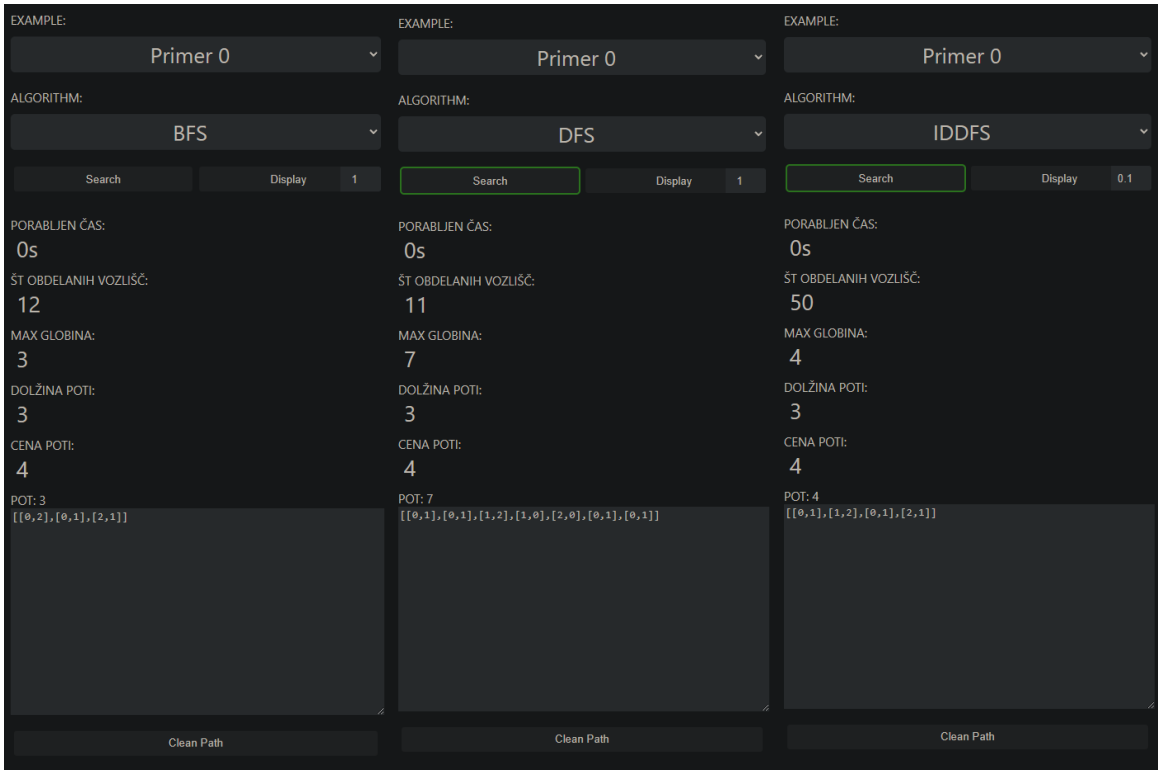
Za hevristiko pri algoritmih A* in IDA* sva imela veliko idej. V prvotni ideji sva za vsako škatlo izračunala premik po x-osi (premik med stolpci), a sva ugotovila, da ta rešitev ni naj boljša. Zato sva dodala kazenjske točke. V primeru, da je izhodni in ciljni stolpec enak, torej škatla ostane v istem stolpcu, še prišteje po 2 kazenjski točki za razliko v y-osi. 2 točki se prišteje saj sva predpostavila, da je treba element odstraniti in kasneje dodati nazaj, kar vzame vsaj 2 premika. Zelo pomembna stvar pa je tudi, da se pozicija po y-osi štejejo od spodaj na vzgor in ne obratno, saj ta deluje kot stack ali sklad.

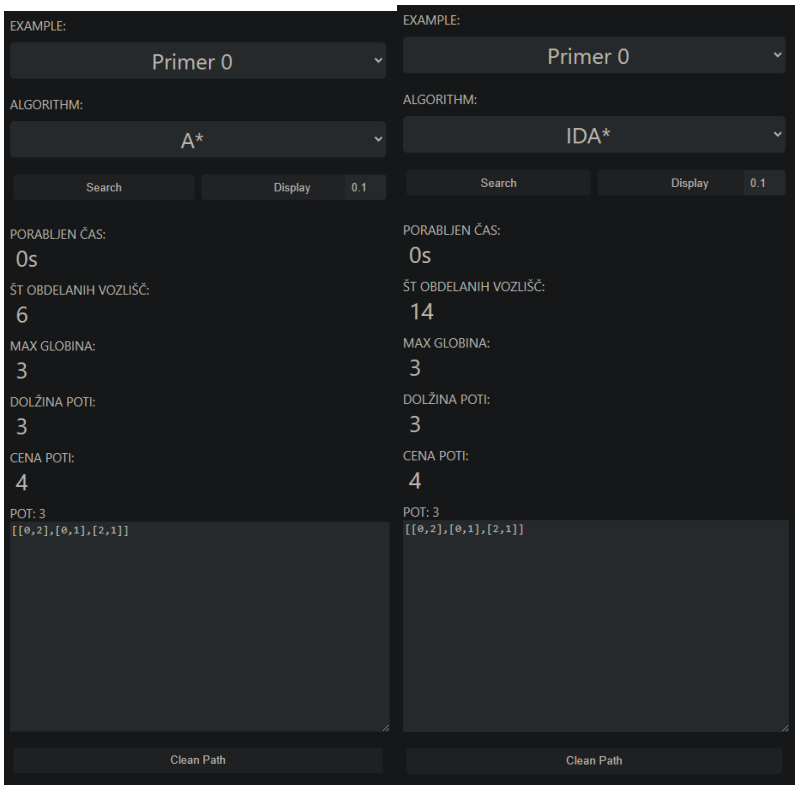
Algorithmi na primerih

Pri drugi seminarski nalogi, smo dobili 5 različnih primerov za testiranje iskalnih algorithmov.

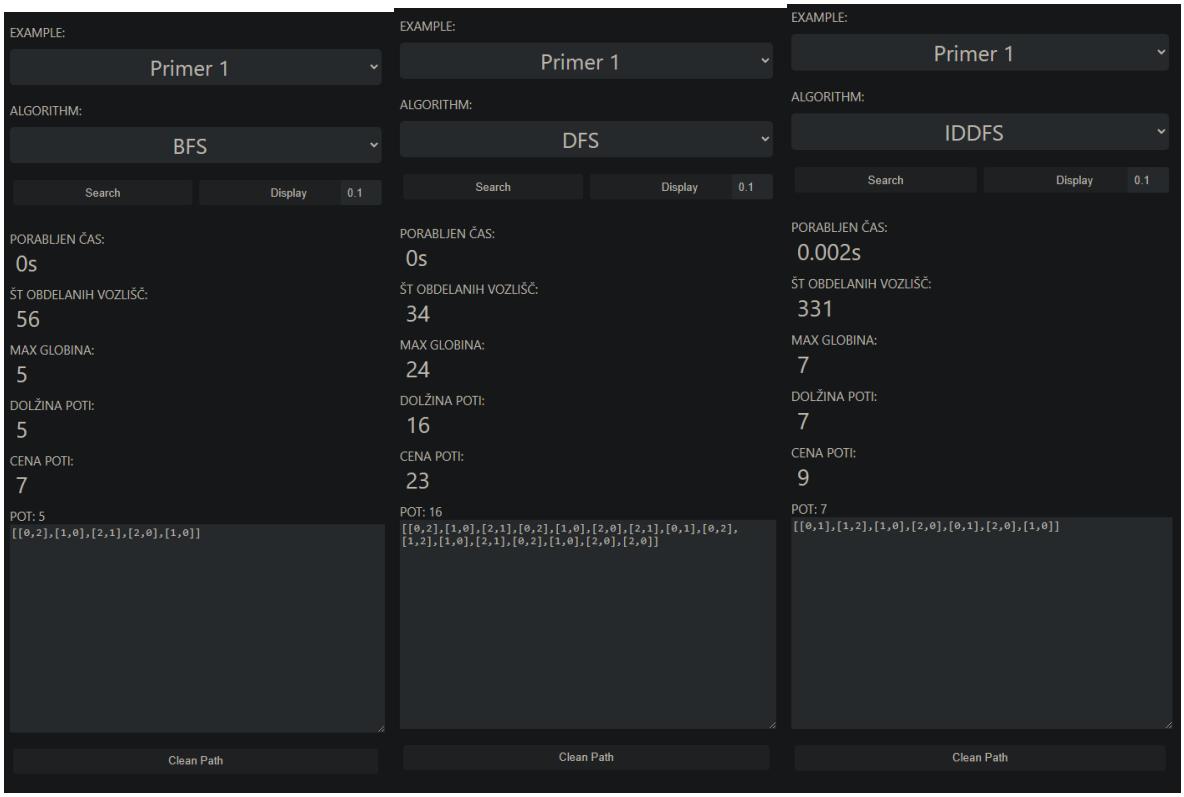
Primer 0

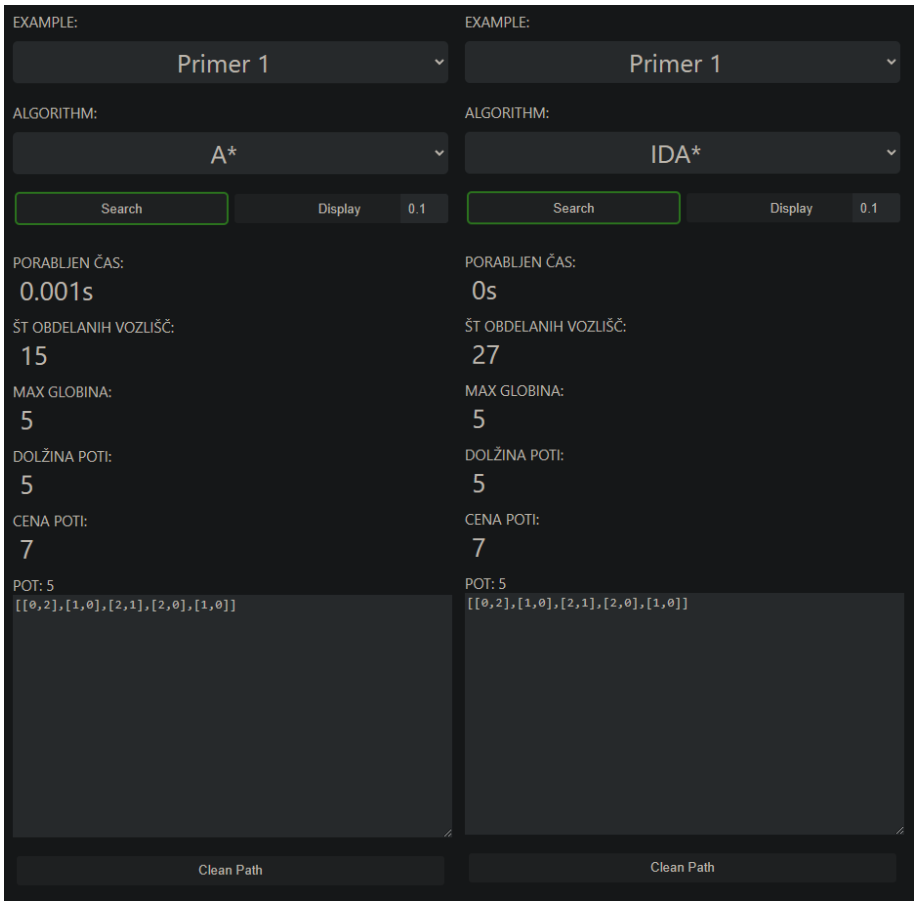
Primer 0, sva dodala midva. Na začetku izdelave algorithmov, sva potrebovala enostaven primer, za testiranje ali algorithem deluje.


EXAMPLE:
Primer 0
ALGORITHM:
BFS
Search
Display 1
PORABLJEN ČAS:
0s
ŠT OBDELANIH VOZLIŠČ:
12
MAX GLOBINA:
3
DOLŽINA POTI:
3
CENA POTI:
4
POT: 3
[[0,2],[0,1],[2,1]]
Clean Path
EXAMPLE:
Primer 0
ALGORITHM:
DFS
Search
Display 1
PORABLJEN ČAS:
0s
ŠT OBDELANIH VOZLIŠČ:
11
MAX GLOBINA:
7
DOLŽINA POTI:
3
CENA POTI:
4
POT: 7
[[0,1],[0,1],[1,2],[1,0],[2,0],[0,1],[0,1]]
Clean Path
EXAMPLE:
Primer 0
ALGORITHM:
IDDFS
Search
Display 0.1
PORABLJEN ČAS:
0s
ŠT OBDELANIH VOZLIŠČ:
50
MAX GLOBINA:
4
DOLŽINA POTI:
3
CENA POTI:
4
POT: 4
[[0,1],[1,2],[0,1],[2,1]]
Clean Path

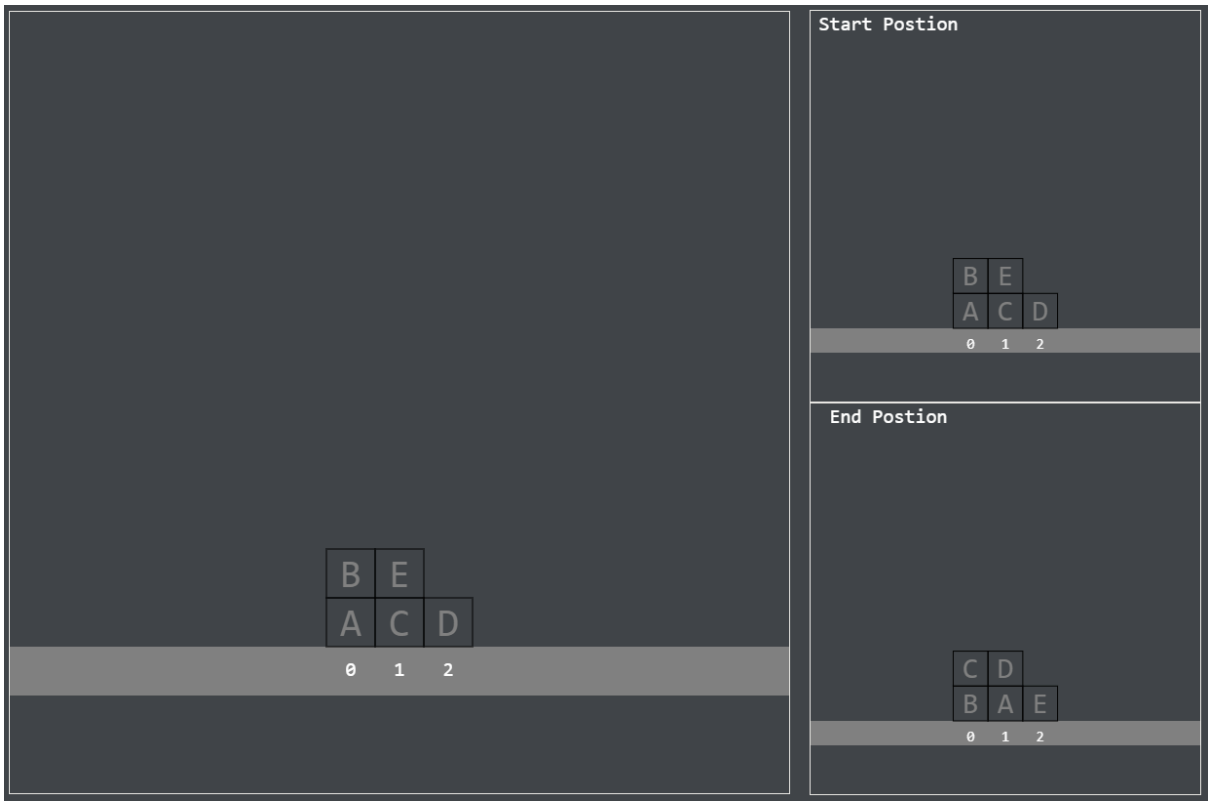

EXAMPLE:
Primer 0
ALGORITHM:
A*
Search
Display 0.1
PORABLJEN ČAS:
0s
ŠT OBDELANIH VOZLIŠČ:
6
MAX GLOBINA:
3
DOLŽINA POTI:
3
CENA POTI:
4
POT: 3
[[0,2],[0,1],[2,1]]
Clean Path
EXAMPLE:
Primer 0
ALGORITHM:
IDA*
Search
Display 0.1
PORABLJEN ČAS:
0s
ŠT OBDELANIH VOZLIŠČ:
14
MAX GLOBINA:
3
DOLŽINA POTI:
3
CENA POTI:
4
POT: 3
[[0,2],[0,1],[2,1]]
Clean Path

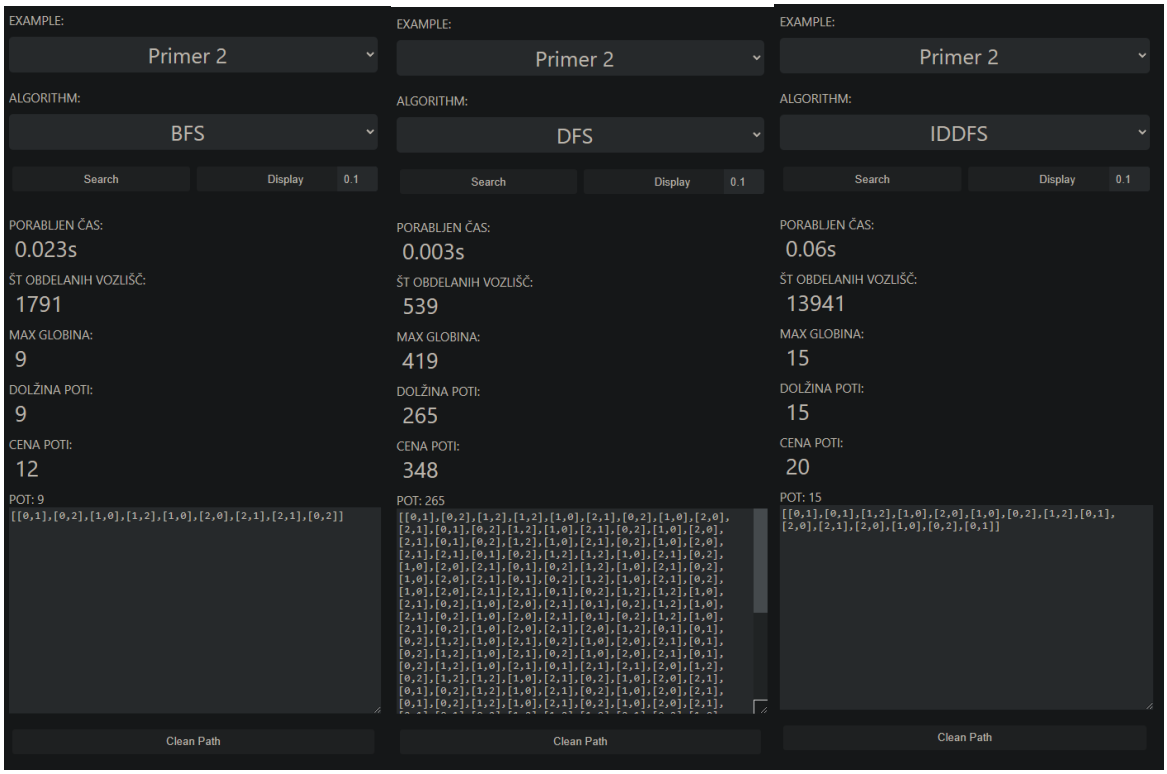
Primer 1

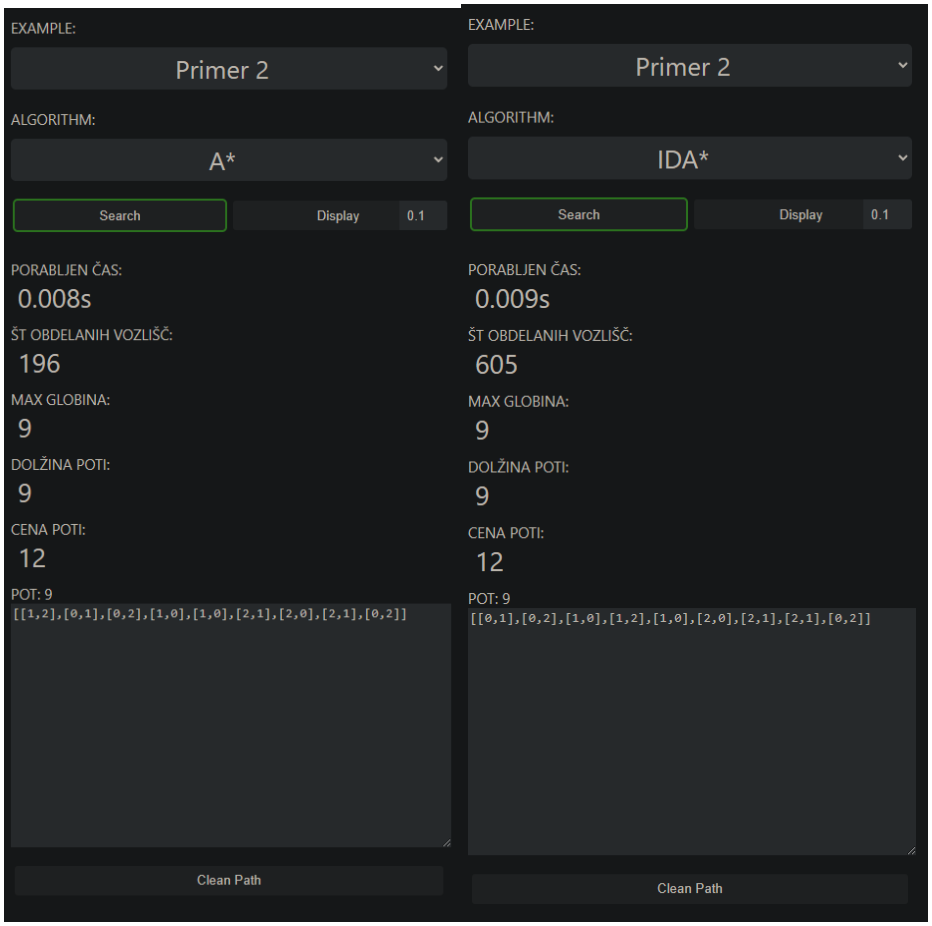

EXAMPLE:
Primer 1
ALGORITHM:
BFS
Search
Display 0.1
PORABLJEN ČAS:
0s
ŠT OBDELANIH VOZLIŠČ:
56
MAX GLOBINA:
5
DOLŽINA POTI:
5
CENA POTI:
7
POT: 5
[[0,2],[1,0],[2,1],[2,0],[1,0]]
Clean Path
EXAMPLE:
Primer 1
ALGORITHM:
DFS
Search
Display 0.1
PORABLJEN ČAS:
0s
ŠT OBDELANIH VOZLIŠČ:
34
MAX GLOBINA:
24
DOLŽINA POTI:
16
CENA POTI:
23
POT: 16
[[0,2],[1,0],[2,1],[0,2],[1,0],[2,0],[2,1],[0,1],[0,2],
[1,2],[1,0],[2,1],[0,2],[1,0],[2,0],[2,0]]
Clean Path
EXAMPLE:
Primer 1
ALGORITHM:
IDDFS
Search
Display 0.1
PORABLJEN ČAS:
0.002s
ŠT OBDELANIH VOZLIŠČ:
331
MAX GLOBINA:
7
DOLŽINA POTI:
7
CENA POTI:
9
POT: 7
[[0,1],[1,2],[1,0],[2,0],[0,1],[2,0],[1,0]]
Clean Path


EXAMPLE:
Primer 1
ALGORITHM:
A*
Search
Display 0.1
PORABLJEN ČAS:
0.001s
ŠT OBDELANIH VOZLIŠČ:
15
MAX GLOBINA:
5
DOLŽINA POTI:
5
CENA POTI:
7
POT: 5
[[0,2],[1,0],[2,1],[2,0],[1,0]]
Clean Path
EXAMPLE:
Primer 1
ALGORITHM:
IDA*
Search
Display 0.1
PORABLJEN ČAS:
0s
ŠT OBDELANIH VOZLIŠČ:
27
MAX GLOBINA:
5
DOLŽINA POTI:
5
CENA POTI:
7
POT: 5
[[0,2],[1,0],[2,1],[2,0],[1,0]]
Clean Path

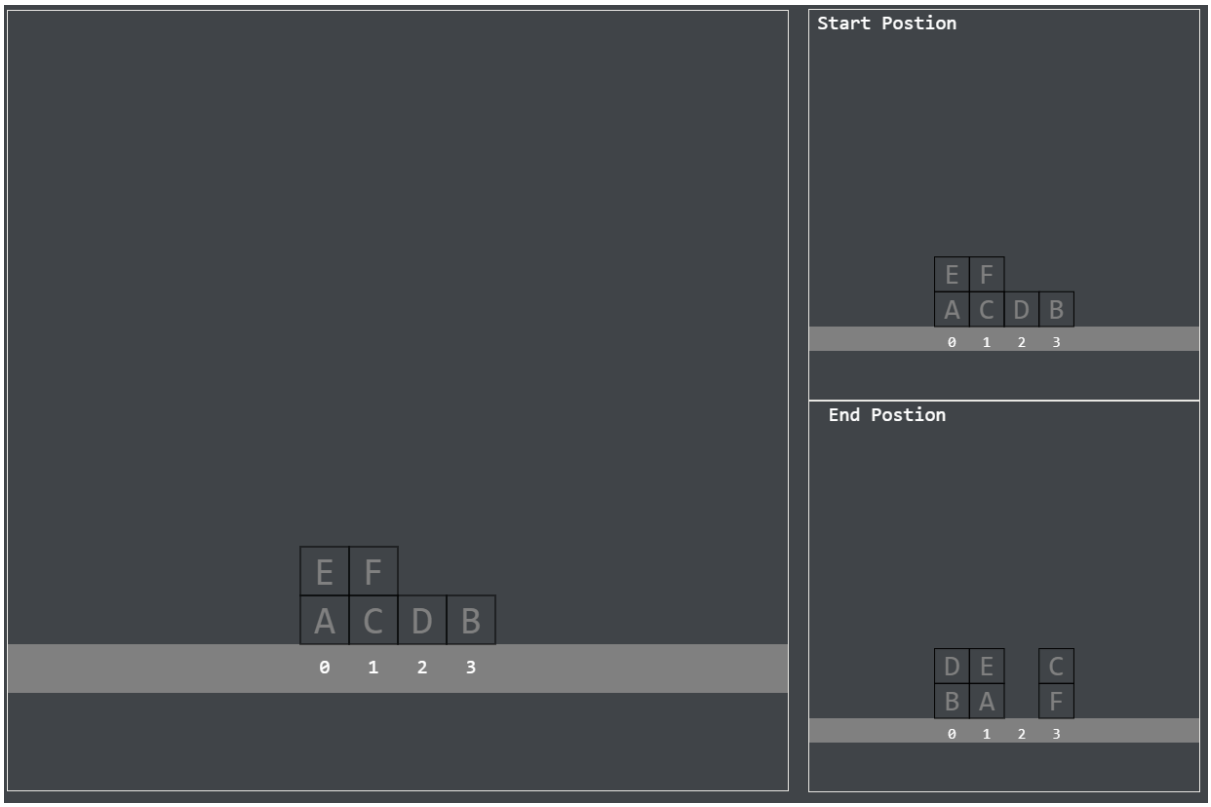
Primer 2

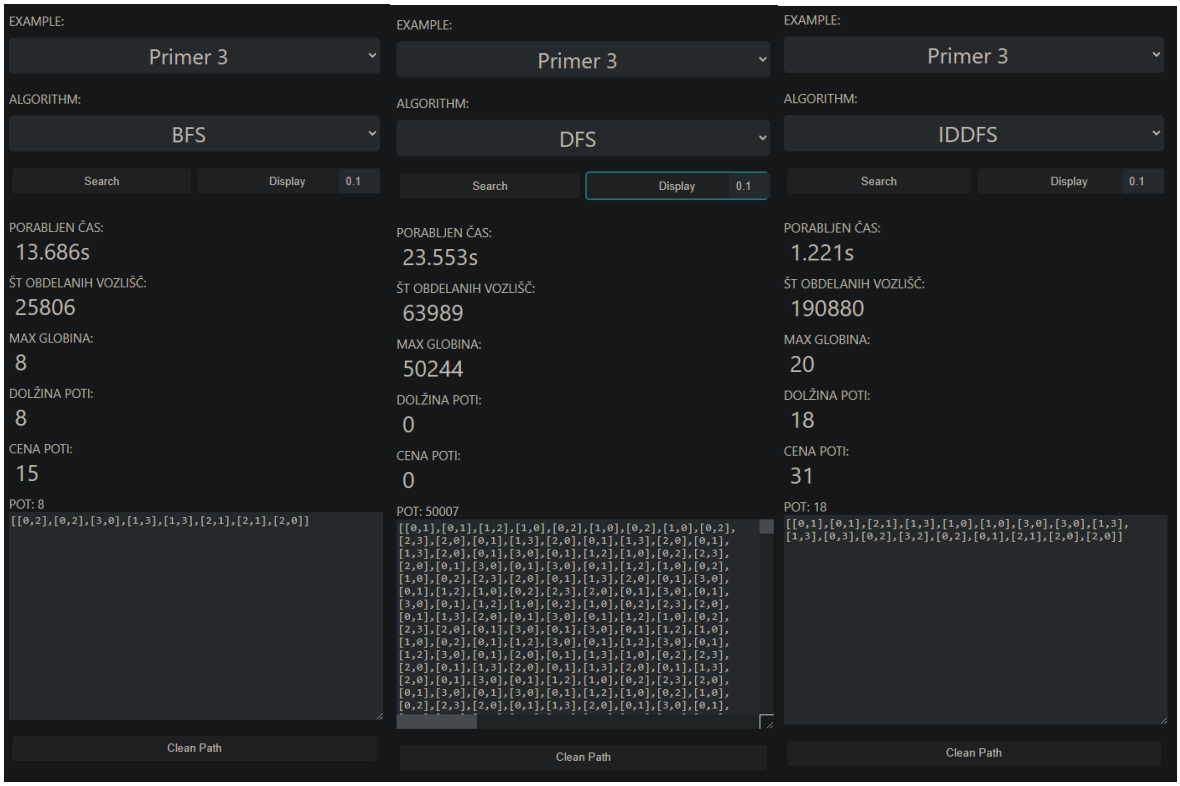


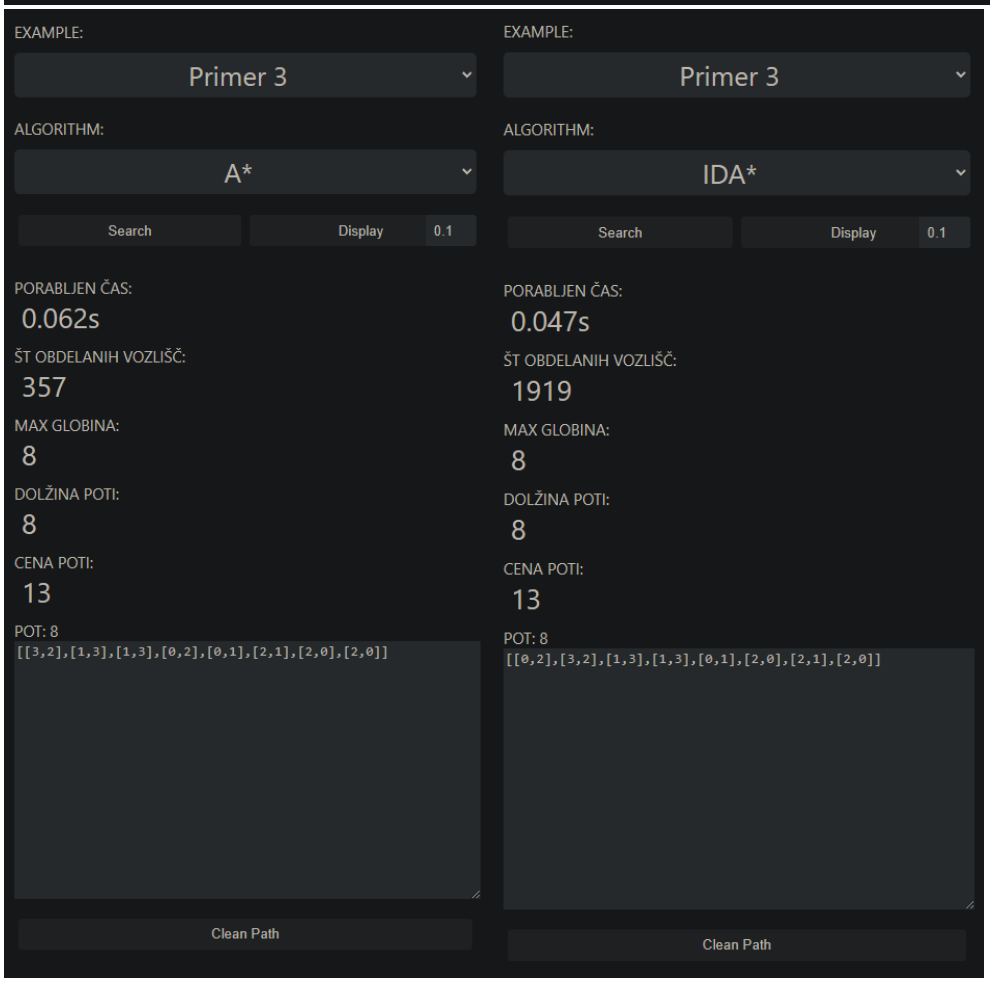

EXAMPLE:
Primer 2
ALGORITHM:
BFS
Search
Display 0.1
PORABLJEN ČAS:
0.023s
ŠT OBDELANIH VOZLIŠČ:
1791
MAX GLOBINA:
9
DOLŽINA POTI:
9
CENA POTI:
12
POT: 9
[[0,1],[0,2],[1,0],[1,2],[1,0],[2,0],[2,1],[2,1],[0,2]]
Clean Path
EXAMPLE:
Primer 2
ALGORITHM:
DFS
Search
Display 0.1
PORABLJEN ČAS:
0.003s
ŠT OBDELANIH VOZLIŠČ:
539
MAX GLOBINA:
419
DOLŽINA POTI:
265
CENA POTI:
348
POT: 265
Clean Path
EXAMPLE:
Primer 2
ALGORITHM:
IDDFS
Search
Display 0.1
PORABLJEN ČAS:
0.06s
ŠT OBDELANIH VOZLIŠČ:
13941
MAX GLOBINA:
15
DOLŽINA POTI:
15
CENA POTI:
20
POT: 15
[[0,1],[0,1],[1,2],[1,0],[2,0],[1,0],[0,2],[1,2],[0,1],
[2,0],[2,1],[2,0],[1,0],[0,2],[0,1]]
Clean Path


EXAMPLE:
Primer 2
ALGORITHM:
A*
Search
Display 0.1
PORABLJEN ČAS:
0.008s
ŠT OBDELANIH VOZLIŠČ:
196
MAX GLOBINA:
9
DOLŽINA POTI:
9
CENA POTI:
12
POT: 9
[[1,2],[0,1],[0,2],[1,0],[1,0],[2,1],[2,0],[2,1],[0,2]]
Clean Path
EXAMPLE:
Primer 2
ALGORITHM:
IDA*
Search
Display 0.1
PORABLJEN ČAS:
0.009s
ŠT OBDELANIH VOZLIŠČ:
605
MAX GLOBINA:
9
DOLŽINA POTI:
9
CENA POTI:
12
POT: 9
[[0,1],[0,2],[1,0],[1,2],[1,0],[2,0],[2,1],[2,1],[0,2]]
Clean Path

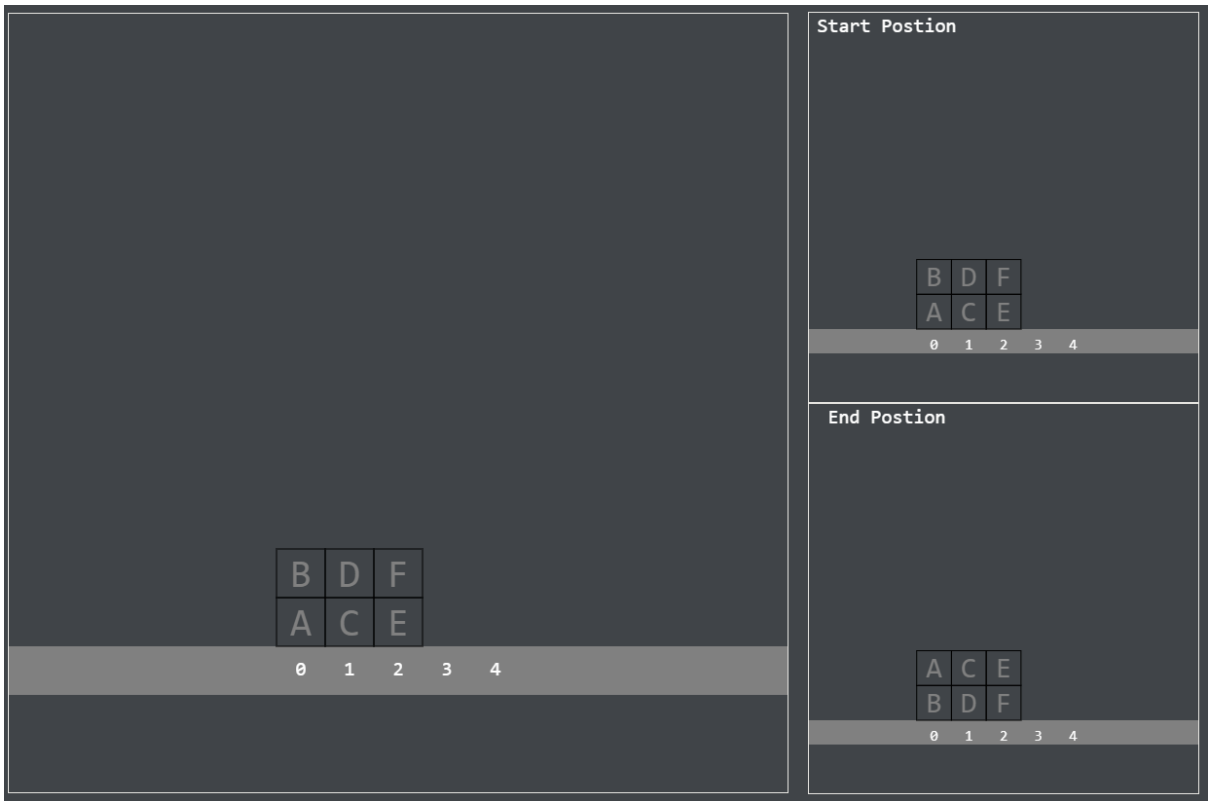
Primer 3

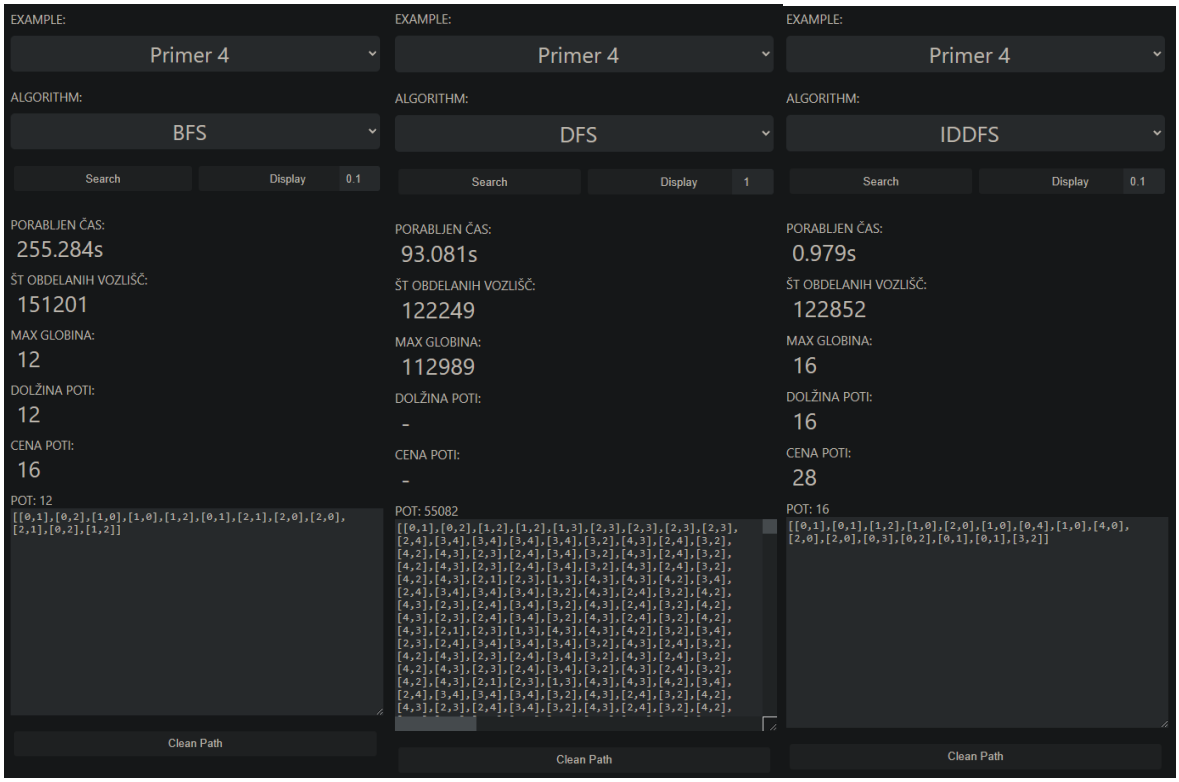

Start Postion
E F
A C D B
0 1 2 3
End Postion
D E C
B A F
0 1 2 3
E F
A C D B
0 1 2 3

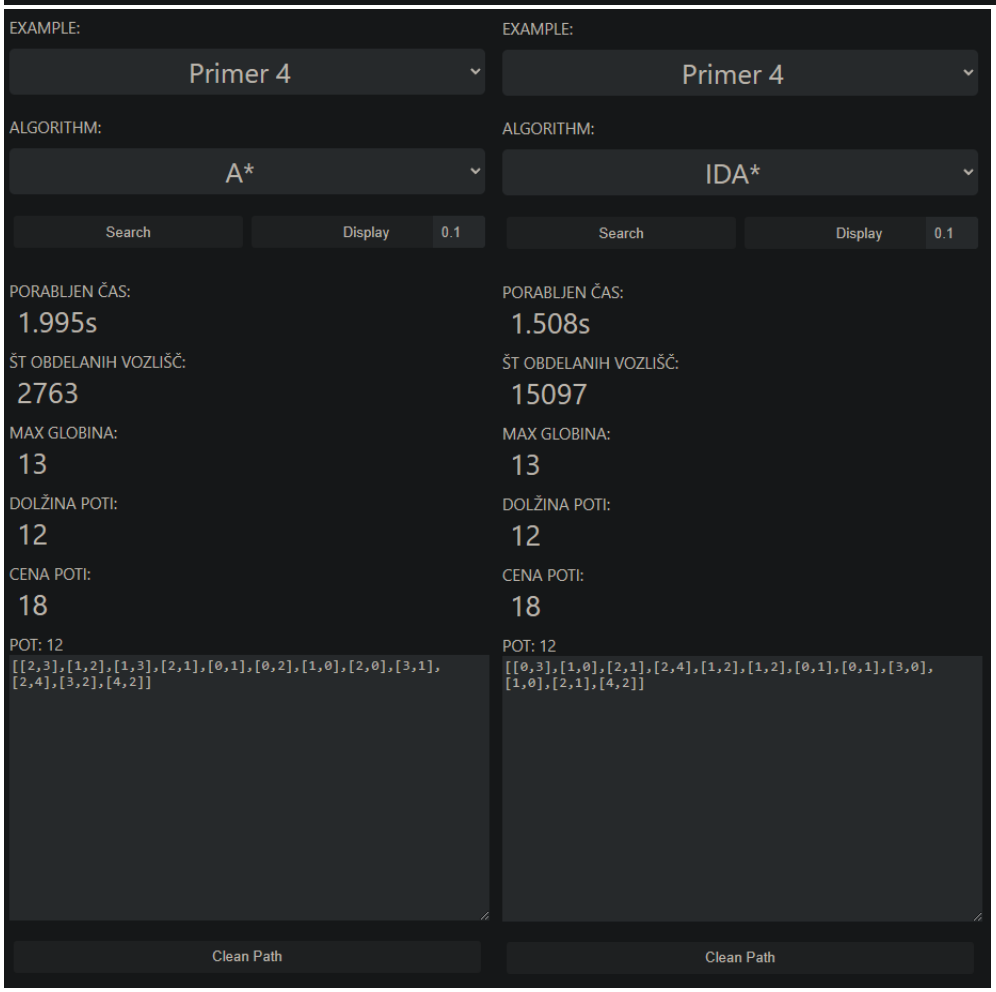

EXAMPLE:
Primer 3
ALGORITHM:
BFS
Search
Display 0.1
PORABLJEN ČAS:
13.686s
ŠT OBDELANIH VOZLIŠČ:
25806
MAX GLOBINA:
8
DOLŽINA POTI:
8
CENA POTI:
15
POT: 8
[[0,2],[0,2],[3,0],[1,3],[1,3],[2,1],[2,1],[2,0]]
Clean Path
EXAMPLE:
Primer 3
ALGORITHM:
DFS
Search
Display 0.1
PORABLJEN ČAS:
23.553s
ŠT OBDELANIH VOZLIŠČ:
63989
MAX GLOBINA:
50244
DOLŽINA POTI:
0
CENA POTI:
0
POT: 50007
Clean Path
EXAMPLE:
Primer 3
ALGORITHM:
IDDFS
Search
Display 0.1
PORABLJEN ČAS:
1.221s
ŠT OBDELANIH VOZLIŠČ:
190880
MAX GLOBINA:
20
DOLŽINA POTI:
18
CENA POTI:
31
POT: 18
[[0,1],[0,1],[2,1],[1,3],[1,0],[1,0],[3,0],[3,0],[1,3],
[1,3],[0,3],[0,2],[3,2],[0,2],[0,1],[2,1],[2,0],[2,0]]
Clean Path


EXAMPLE:
Primer 3
ALGORITHM:
A*
Search
Display 0.1
PORABLJEN ČAS:
0.062s
ŠT OBDELANIH VOZLIŠČ:
357
MAX GLOBINA:
8
DOLŽINA POTI:
8
CENA POTI:
13
POT: 8
[[3,2],[1,3],[1,3],[0,2],[0,1],[2,1],[2,0],[2,0]]
Clean Path
EXAMPLE:
Primer 3
ALGORITHM:
IDA*
Search
Display 0.1
PORABLJEN ČAS:
0.047s
ŠT OBDELANIH VOZLIŠČ:
1919
MAX GLOBINA:
8
DOLŽINA POTI:
8
CENA POTI:
13
POT: 8
[[0,2],[3,2],[1,3],[1,3],[0,1],[2,0],[2,1],[2,0]]
Clean Path

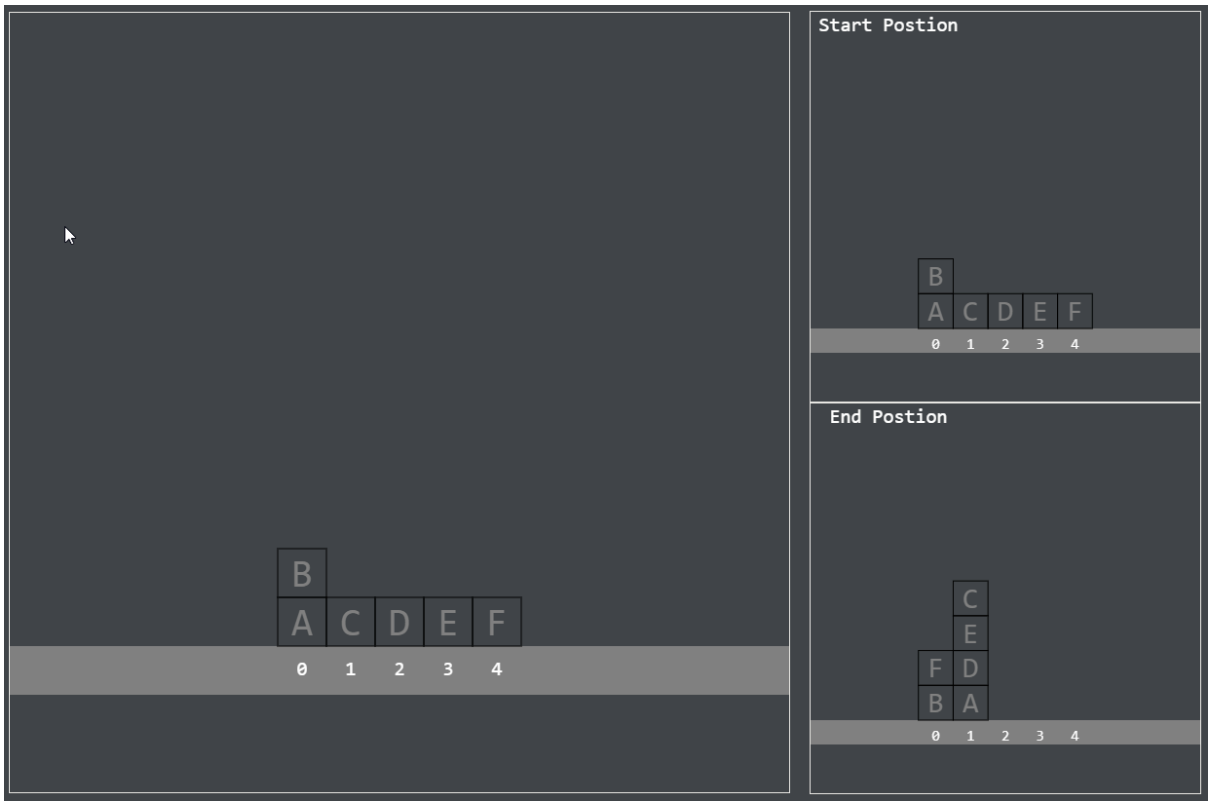
Primer 4

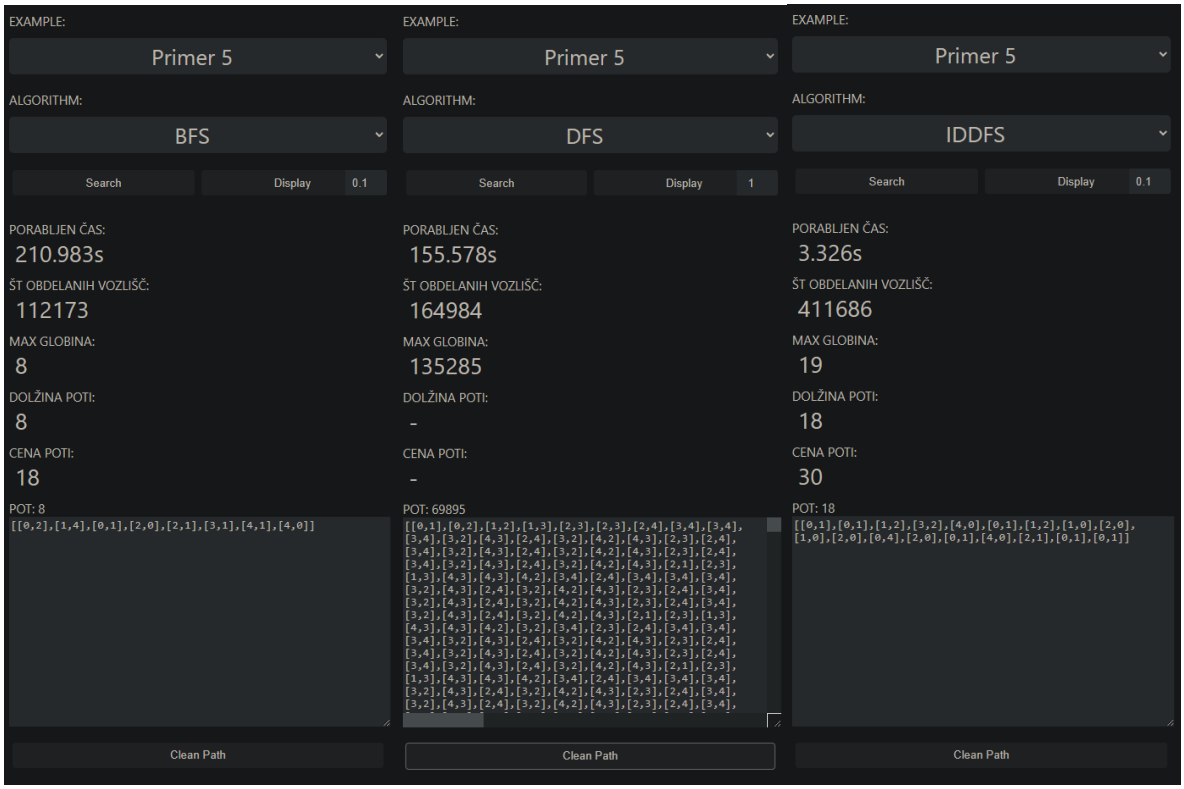

B D F
A C E
0 1 2 3 4
Start Postion
B D F
A C E
0 1 2 3 4
End Postion
A C E
B D F
0 1 2 3 4

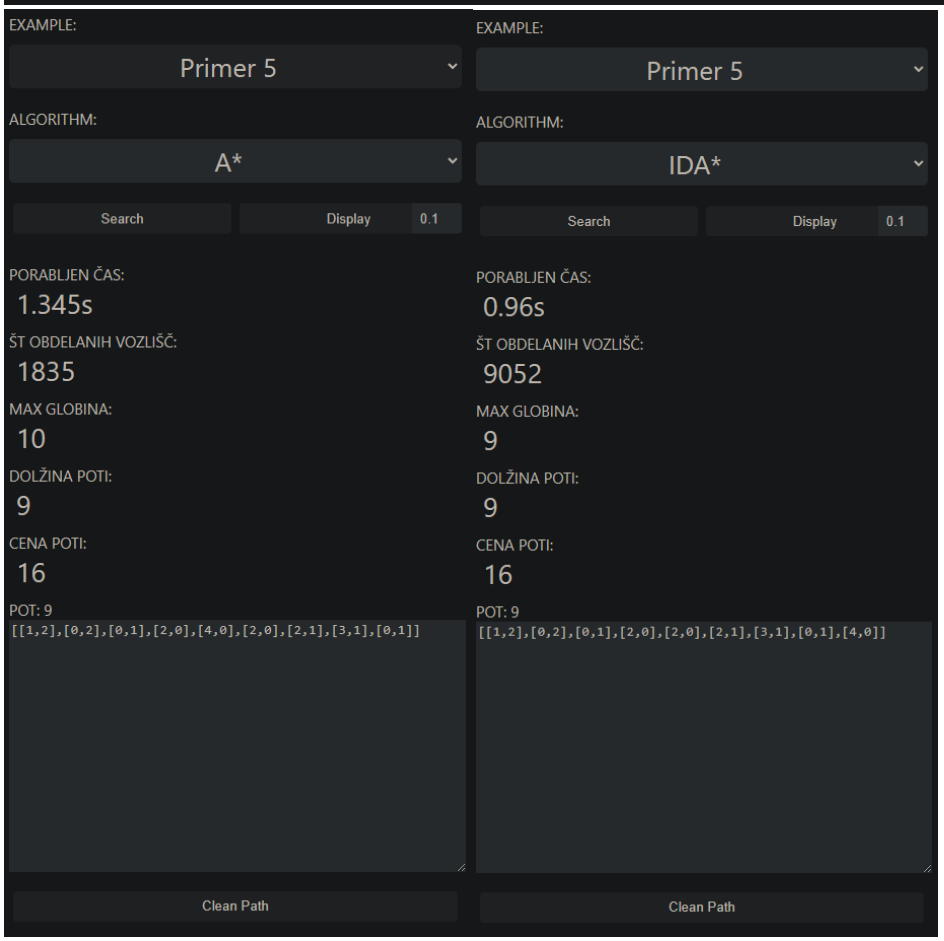

EXAMPLE:
Primer 4
ALGORITHM:
BFS
Search
Display 0.1
PORABLJEN ČAS:
255.284s
ŠT OBDELANIH VOZLIŠČ:
151201
MAX GLOBINA:
12
DOLŽINA POTI:
12
CENA POTI:
16
POT: 12
[[0,1],[0,2],[1,0],[1,0],[1,2],[0,1],[2,1],[2,0],[2,0],
[2,1],[0,2],[1,2]]
Clean Path
EXAMPLE:
Primer 4
ALGORITHM:
DFS
Search
Display 1
PORABLJEN ČAS:
93.081s
ŠT OBDELANIH VOZLIŠČ:
122249
MAX GLOBINA:
112989
DOLŽINA POTI:
-
CENA POTI:
-
POT: 55082
[[0,1],[0,2],[1,2],[1,2],[1,3],[2,3],[2,3],[2,3],[2,3],
[2,4],[3,4],[3,4],[3,4],[3,4],[3,2],[4,3],[2,4],[3,2],
[4,2],[4,3],[2,3],[2,4],[3,4],[3,2],[4,3],[2,4],[3,2],
[4,2],[4,3],[2,3],[2,4],[3,4],[3,2],[4,3],[2,4],[3,2],
[4,2],[4,3],[2,1],[2,3],[1,3],[4,3],[4,3],[4,2],[3,4],
[2,4],[3,4],[3,4],[3,4],[3,2],[4,3],[2,4],[3,2],[4,2],
[4,3],[2,3],[2,4],[3,4],[3,2],[4,3],[2,4],[3,2],[4,2],
[4,3],[2,3],[2,4],[3,4],[3,2],[4,3],[2,4],[3,2],[4,2],
[4,3],[2,1],[2,3],[1,3],[4,3],[4,3],[4,2],[3,2],[3,4],
[2,3],[2,4],[3,4],[3,4],[3,4],[3,2],[4,3],[2,4],[3,2],
[4,2],[4,3],[2,3],[2,4],[3,4],[3,2],[4,3],[2,4],[3,2],
[4,2],[4,3],[2,3],[2,4],[3,4],[3,2],[4,3],[2,4],[3,2],
[4,2],[4,3],[2,1],[2,3],[1,3],[4,3],[4,3],[4,2],[3,4],
[2,4],[3,4],[3,4],[3,4],[3,2],[4,3],[2,4],[3,2],[4,2],
[4,3],[2,3],[2,4],[3,4],[3,2],[4,3],[2,4],[3,2],[4,2],
Clean Path
EXAMPLE:
Primer 4
ALGORITHM:
IDDFS
Search
Display 0.1
PORABLJEN ČAS:
0.979s
ŠT OBDELANIH VOZLIŠČ:
122852
MAX GLOBINA:
16
DOLŽINA POTI:
16
CENA POTI:
28
POT: 16
[[0,1],[0,1],[1,2],[1,0],[2,0],[1,0],[0,4],[1,0],[4,0],
[2,0],[2,0],[0,3],[0,2],[0,1],[0,1],[3,2]]
Clean Path


EXAMPLE:
Primer 4
ALGORITHM:
A*
Search
Display 0.1
PORABLJEN ČAS:
1.995s
ŠT OBDELANIH VOZLIŠČ:
2763
MAX GLOBINA:
13
DOLŽINA POTI:
12
CENA POTI:
18
POT: 12
[[2,3],[1,2],[1,3],[2,1],[0,1],[0,2],[1,0],[2,0],[3,1],
[2,4],[3,2],[4,2]]
Clean Path
EXAMPLE:
Primer 4
ALGORITHM:
IDA*
Search
Display 0.1
PORABLJEN ČAS:
1.508s
ŠT OBDELANIH VOZLIŠČ:
15097
MAX GLOBINA:
13
DOLŽINA POTI:
12
CENA POTI:
18
POT: 12
[[0,3],[1,0],[2,1],[2,4],[1,2],[1,2],[0,1],[0,1],[3,0],
[1,0],[2,1],[4,2]]
Clean Path

Primer 5


Start Postion
B
A C D E F
0 1 2 3 4
End Postion
C
E
F D
B A
0 1 2 3 4
B
A C D E F
0 1 2 3 4


EXAMPLE:
Primer 5
ALGORITHM:
BFS
Search
Display 0.1
PORABLJEN ČAS:
210.983s
ŠT OBDELANIH VOZLIŠČ:
112173
MAX GLOBINA:
8
DOLŽINA POTI:
8
CENA POTI:
18
POT: 8
[[0,2],[1,4],[0,1],[2,0],[2,1],[3,1],[4,1],[4,0]]
Clean Path
EXAMPLE:
Primer 5
ALGORITHM:
DFS
Search
Display 1
PORABLJEN ČAS:
155.578s
ŠT OBDELANIH VOZLIŠČ:
164984
MAX GLOBINA:
135285
DOLŽINA POTI:
-
CENA POTI:
-
POT: 69895
Clean Path
EXAMPLE:
Primer 5
ALGORITHM:
IDDFS
Search
Display 0.1
PORABLJEN ČAS:
3.326s
ŠT OBDELANIH VOZLIŠČ:
411686
MAX GLOBINA:
19
DOLŽINA POTI:
18
CENA POTI:
30
POT: 18
[[0,1],[0,1],[1,2],[3,2],[4,0],[0,1],[1,2],[1,0],[2,0],
[1,0],[2,0],[0,4],[2,0],[0,1],[4,0],[2,1],[0,1],[0,1]]
Clean Path


EXAMPLE:
Primer 5
ALGORITHM:
A*
Search
Display 0.1
PORABLJEN ČAS:
1.345s
ŠT OBDELANIH VOZLIŠČ:
1835
MAX GLOBINA:
10
DOLŽINA POTI:
9
CENA POTI:
16
POT: 9
[[1,2],[0,2],[0,1],[2,0],[4,0],[2,0],[2,1],[3,1],[0,1]]
Clean Path
EXAMPLE:
Primer 5
ALGORITHM:
IDA*
Search
Display 0.1
PORABLJEN ČAS:
0.96s
ŠT OBDELANIH VOZLIŠČ:
9052
MAX GLOBINA:
9
DOLŽINA POTI:
9
CENA POTI:
16
POT: 9
[[1,2],[0,2],[0,1],[2,0],[2,0],[2,1],[3,1],[0,1],[4,0]]
Clean Path

Zaključek

Med izdelovanjem seminarske naloge sva ugotovila veliko stvari.

Pri izdelavi spletne strani in vizualizaciji rezultatov iskalnih algorithmov sva poznala, kako se vsak algorithem razlikuje ter njihove prednosti in slabosti pri uporabi ter izdelavi. Pri prikazovanju rezultatov sva želela bit čim bolj kreativna, zato sva dodala animacijo poti vsakega algorithma. Za lažjo primerjavo sva pa še dodala začetno in končno stanje, da uporabnik lažje razbere delovanje.

Pri implementaciji in testiranju vseh iskalnih algorithmov sva opazila, da algorithma A* in IDA* sta daleč najhitrejša pri izvedbi ter njune rešitve so vedno krajše v primerjavi z ostalimi algorithmi. Med slabšimi algorithmi pri tej nalogi pa je algorithem DFS. Poskusila sva tudi izdelati genetski algorithem, ampak zaradi kompleksnosti nama to ni uspelo.