POROČILO ZA 2. SEMINARSKO NALOGO PRI PREDMETU UMETNA INTELIGENCA

Šolsko leto 2020/21

Nik Prinčič (63190240)

1. Prostor stanj ______ *3*

2. Implementirani algoritmi ______ *3*

2.1. BFS ______ **4**

2.2. DFS ______ **4**

2.3. IDDFS ______ **5**

2.4. A* ______ **5**

2.5. Hill Climbing ______ **6**

2.6. Best First ______ **7**

3. Primerjava učinkovitosti algoritmov ______ *7*

3.1. Primerjava algoritmov, ki dobijo optimalno pot ______ **8**

3.2. Primerjava učinkovitosti algoritma A* z uporabo različnih hevristik ______ **8**

3.1. Časovno in prostorsko najbolj učinkoviti algoritmi (optimalni in neoptimalni) na posameznih primerih ______ **9**

1. Prostor stanj

Prostor stanj je predstavljen z matrikami, ki vsebujejo trenutno postavitev škatel. Iz vsakega stanja pa se naslednike lahko razvije tako, da se vse vrhnje škatle premakne na vsa dovoljena mesta.

Primer:

Začetno stanje:

A	B	C

Nasledniki se bodo generirali v naslednjem zaporedju:

	A	
	B	C

		A
	B	C

B		
A		C

		B
A		C

C		
A	B	

	C	
A	B	

Matrike sem implementiral tako, da matrika hrani indekse že premaknjenih stolpcev, tako da se iz enega stanja ne bi dvakrat razvilo isto stanje.

Pri algoritmih BFS, DFS, Hill climbing in Best first sem razvijanje novih stanj omejil tudi tako da se eno stanje lahko razvije samo enkrat, pri algoritmu IDDFS sem pa razvijanje omejil tako da se v trenutnem skladu ne smeta pojaviti dve isti stanji.

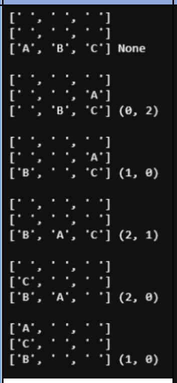
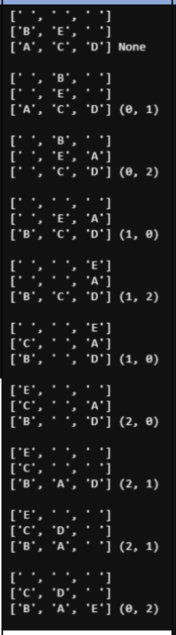
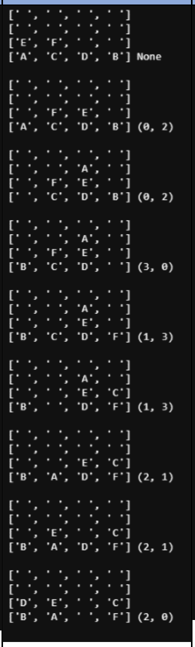
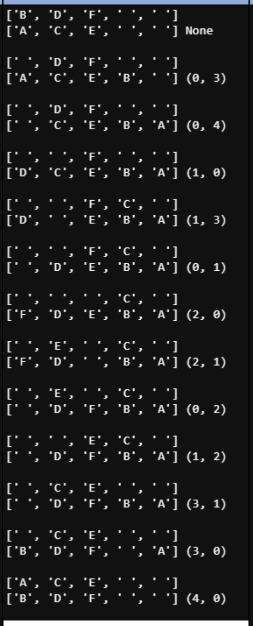
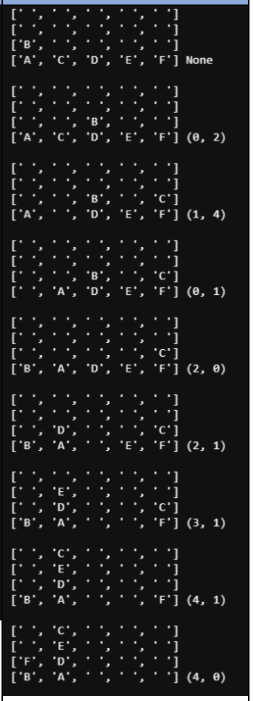
2. Implementirani algoritmi

Implementiral sem 6 algoritmov in pri informiranih algoritmih uporabil 5 različnih hevristik. Vse algoritme sem implementiral iterativno v jeziku Python. Uspešnost algoritmov pa sem meril z sledečimi metrikami:

- Število opravljenih korakov do cilja (Steps)
- Število razvitih stanj/vozlišč (Total nodes generated)
- Število stanj/vozlišč hranjenih v spominu v trenutku ko je bil najden cilj (Nodes in memory)
- Maksimalno število hranjenih stanj/vozlišč v spominu tekom izvajanja (Max. nodes in memory)

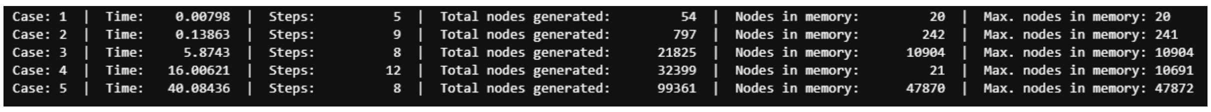
Optimalne premike sem dobil z uporabo DFS, IDDFS (izvedel sem samo do 3 primera zaradi predolgega trajanja) in A* z uporabo hevristike »misplaced«.

Optimalne rešitve so sledeče:

Primer	1	2	3	4	5
	 [' ', ' ', ' '] [' ', ' ', ' '] ['A', 'B', 'C'] None [' ', ' ', ' '] [' ', ' ', 'A'] [' ', 'B', 'C'] (0, 2) [' ', ' ', ' '] [' ', ' ', 'A'] ['B', ' ', 'C'] (1, 0) [' ', ' ', ' '] [' ', ' ', ' '] ['B', 'A', 'C'] (2, 1) [' ', ' ', ' '] ['C', ' ', ' '] ['B', 'A', ' '] (2, 0) ['A', ' ', ' '] ['C', ' ', ' '] ['B', ' ', ' '] (1, 0)	 [' ', ' ', ' '] ['B', 'E', ' '] ['A', 'C', 'D'] None [' ', 'B', ' '] [' ', 'E', ' '] ['A', 'C', 'D'] (0, 1) [' ', 'B', ' '] [' ', 'E', 'A'] [' ', 'C', 'D'] (0, 2) [' ', ' ', ' '] [' ', 'E', 'A'] ['B', 'C', 'D'] (1, 0) [' ', ' ', 'E'] [' ', ' ', 'A'] ['B', 'C', 'D'] (1, 2) [' ', ' ', 'E'] ['C', ' ', 'A'] ['B', ' ', 'D'] (1, 0) ['E', ' ', ' '] ['C', ' ', 'A'] ['B', ' ', 'D'] (2, 0) ['E', ' ', ' '] ['C', ' ', ' '] ['B', 'A', 'D'] (2, 1) ['E', ' ', ' '] ['C', 'D', ' '] ['B', 'A', ' '] (2, 1) [' ', ' ', ' '] ['C', 'D', ' '] ['B', 'A', 'E'] (0, 2)	 [' ', ' ', ' ', ' '] [' ', ' ', ' ', ' '] ['E', 'F', ' ', ' '] ['A', 'C', 'D', 'B'] None [' ', ' ', ' ', ' '] [' ', ' ', ' ', ' '] [' ', 'F', 'E', ' '] ['A', 'C', 'D', 'B'] (0, 2) [' ', ' ', ' ', ' '] [' ', ' ', 'A', ' '] [' ', 'F', 'E', ' '] [' ', 'C', 'D', 'B'] (0, 2) [' ', ' ', ' ', ' '] [' ', ' ', 'A', ' '] [' ', 'F', 'E', ' '] ['B', 'C', 'D', ' '] (3, 0) [' ', ' ', ' ', ' '] [' ', ' ', 'A', ' '] [' ', ' ', 'E', ' '] ['B', 'C', 'D', 'F'] (1, 3) [' ', ' ', ' ', ' '] [' ', ' ', 'A', ' '] [' ', ' ', 'E', 'C'] ['B', ' ', 'D', 'F'] (1, 3) [' ', ' ', ' ', ' '] [' ', ' ', ' ', ' '] [' ', ' ', 'E', 'C'] ['B', 'A', 'D', 'F'] (2, 1) [' ', ' ', ' ', ' '] [' ', ' ', ' ', ' '] [' ', 'E', ' ', 'C'] ['B', 'A', 'D', 'F'] (2, 1) [' ', ' ', ' ', ' '] [' ', ' ', ' ', ' '] ['D', 'E', ' ', 'C'] ['B', 'A', ' ', 'F'] (2, 0)	 ['B', 'D', 'F', ' ', ' '] ['A', 'C', 'E', ' ', ' '] None [' ', 'D', 'F', ' ', ' '] ['A', 'C', 'E', 'B', ' '] (0, 3) [' ', 'D', 'F', ' ', ' '] [' ', 'C', 'E', 'B', 'A'] (0, 4) [' ', ' ', 'F', ' ', ' '] ['D', 'C', 'E', 'B', 'A'] (1, 0) [' ', ' ', 'F', 'C', ' '] ['D', ' ', 'E', 'B', 'A'] (1, 3) [' ', ' ', 'F', 'C', ' '] [' ', 'D', 'E', 'B', 'A'] (0, 1) [' ', ' ', ' ', 'C', ' '] ['F', 'D', 'E', 'B', 'A'] (2, 0) [' ', 'E', ' ', 'C', ' '] ['F', 'D', ' ', 'B', 'A'] (2, 1) [' ', 'E', ' ', 'C', ' '] [' ', 'D', 'F', 'B', 'A'] (0, 2) [' ', ' ', 'E', 'C', ' '] [' ', 'D', 'F', 'B', 'A'] (1, 2) [' ', 'C', 'E', ' ', ' '] [' ', 'D', 'F', 'B', 'A'] (3, 1) [' ', 'C', 'E', ' ', ' '] ['B', 'D', 'F', ' ', 'A'] (3, 0) ['A', 'C', 'E', ' ', ' '] ['B', 'D', 'F', ' ', ' '] (4, 0)	 [' ', ' ', ' ', ' ', ' '] [' ', ' ', ' ', ' ', ' '] ['B', ' ', ' ', ' ', ' '] ['A', 'C', 'D', 'E', 'F'] None [' ', ' ', ' ', ' ', ' '] [' ', ' ', ' ', ' ', ' '] [' ', ' ', 'B', ' ', ' '] ['A', 'C', 'D', 'E', 'F'] (0, 2) [' ', ' ', ' ', ' ', ' '] [' ', ' ', ' ', ' ', ' '] [' ', ' ', 'B', ' ', 'C'] ['A', ' ', 'D', 'E', 'F'] (1, 4) [' ', ' ', ' ', ' ', ' '] [' ', ' ', ' ', ' ', ' '] [' ', ' ', 'B', ' ', 'C'] [' ', 'A', 'D', 'E', 'F'] (0, 1) [' ', ' ', ' ', ' ', ' '] [' ', ' ', ' ', ' ', ' '] [' ', ' ', ' ', ' ', 'C'] ['B', 'A', 'D', 'E', 'F'] (2, 0) [' ', ' ', ' ', ' ', ' '] [' ', ' ', ' ', ' ', ' '] [' ', 'D', ' ', ' ', 'C'] ['B', 'A', ' ', 'E', 'F'] (2, 1) [' ', ' ', ' ', ' ', ' '] [' ', 'E', ' ', ' ', ' '] [' ', 'D', ' ', ' ', 'C'] ['B', 'A', ' ', ' ', 'F'] (3, 1) [' ', 'C', ' ', ' ', ' '] [' ', 'E', ' ', ' ', ' '] [' ', 'D', ' ', ' ', ' '] ['B', 'A', ' ', ' ', 'F'] (4, 1) [' ', 'C', ' ', ' ', ' '] [' ', 'E', ' ', ' ', ' '] ['F', 'D', ' ', ' ', ' '] ['B', 'A', ' ', ' ', ' '] (4, 0)
Število korakov	5	9	8	12	8

2.1. BFS

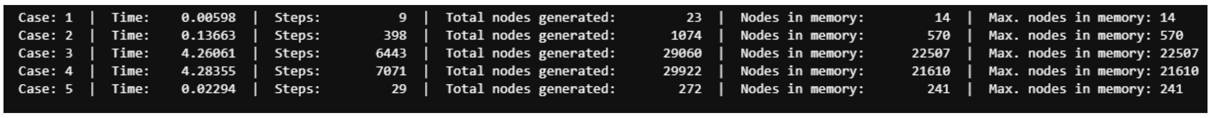
Algoritem sem implementiral iterativno z uporabo vrste (queue). Razvijanje novih stanj sem omejil tako, da se stanje ki je že bilo razvito ne sme več razviti. S takim pristopom sem dobil sledeče rezultate:



```
Case: 1  |  Time:    0.00798  |  Steps:          5  |  Total nodes generated:       54  |  Nodes in memory:        20  |  Max. nodes in memory: 20
Case: 2  |  Time:    0.13863  |  Steps:          9  |  Total nodes generated:      797  |  Nodes in memory:       242  |  Max. nodes in memory: 241
Case: 3  |  Time:     5.8743  |  Steps:          8  |  Total nodes generated:    21825  |  Nodes in memory:     10904  |  Max. nodes in memory: 10904
Case: 4  |  Time:   16.00621  |  Steps:         12  |  Total nodes generated:    32399  |  Nodes in memory:        21  |  Max. nodes in memory: 10691
Case: 5  |  Time:   40.08436  |  Steps:          8  |  Total nodes generated:    99361  |  Nodes in memory:     47870  |  Max. nodes in memory: 47872
```

2.2. DFS

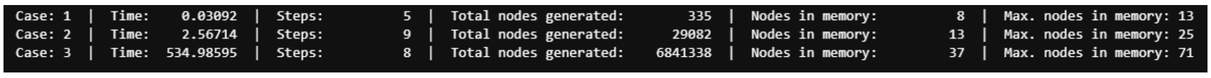
Algoritem sem implementiral iterativno z uporabo sklada (stack). Razvijanje novih stanj sem omejil tako, da se stanje ki je že bilo razvito ne sme več razviti, zavedam da to ni ravno najboljša rešitev saj se tako lahko zgodi da se stanje najprej razvije na večji globini in se zaradi tega ne more več razviti na manjši globini, a glede na to da DFS ne išče optimalne rešitve se mi je zdela pridobljena časovna pohitritev vredna teg. S takim pristopom sem dobil sledeče rezultate:



```
Case: 1  |  Time:    0.00598  |  Steps:          9  |  Total nodes generated:       23  |  Nodes in memory:        14  |  Max. nodes in memory: 14
Case: 2  |  Time:    0.13663  |  Steps:        398  |  Total nodes generated:     1074  |  Nodes in memory:       570  |  Max. nodes in memory: 570
Case: 3  |  Time:    4.26061  |  Steps:       6443  |  Total nodes generated:    29060  |  Nodes in memory:     22507  |  Max. nodes in memory: 22507
Case: 4  |  Time:    4.28355  |  Steps:       7071  |  Total nodes generated:    29922  |  Nodes in memory:     21610  |  Max. nodes in memory: 21610
Case: 5  |  Time:    0.02294  |  Steps:         29  |  Total nodes generated:      272  |  Nodes in memory:       241  |  Max. nodes in memory: 241
```

2.3. IDDFS

Algoritem sem implementiral iterativno kliče DFS z omejeno globino in jo po vsaki iteraciji poveča za 1. Razvijanje novih stanj sem omejil tako, da se stanje ki je v trenutni poti, ne sme več razviti, ko DFS zaide v slepo ulico in se vrne na manjšo globino se lahko že nastala stanja na večji globini spet ponovijo. Časovna zahtevnost tega algoritma seveda eksponentno narašča, zato sem uspel izvesti algoritem samo na prvih treh primerih in dobil sledeče rezultate:



```
Case: 1 |  Time:    0.03092 |  Steps:          5 |  Total nodes generated:       335 |  Nodes in memory:          8 |  Max. nodes in memory: 13
Case: 2 |  Time:    2.56714 |  Steps:          9 |  Total nodes generated:     29082 |  Nodes in memory:         13 |  Max. nodes in memory: 25
Case: 3 |  Time:  534.98595 |  Steps:          8 |  Total nodes generated:   6841338 |  Nodes in memory:         37 |  Max. nodes in memory: 71
```

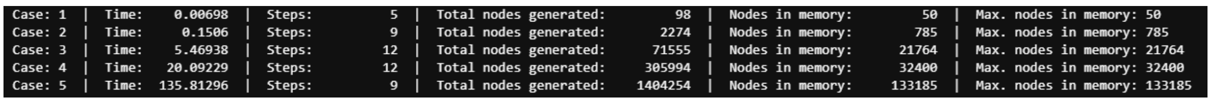
2.4. A*

Pri algoritmu A* in tudi pri naslednjih dveh informiranih algoritmih sem uporabil sledeče hevristike:

- row: za vsako škatlo upošteva absolutno razliko vertikalne pozicije škatle v trenutnem stanju in končnem/začetnem
- column: za vsako škatlo upošteva absolutno razliko horizontalne pozicije škatle v trenutnem stanju in končnem/začetnem
- manhattan: Manhatanska razdalja med škatlama v trenutnem in končnem/začetnem stanju
- euclidean: Evklidska razdalja med škatlama v trenutnem in končnem/začetnem stanju
- misplaced: za vsako škatlo ki ni na enakem mestu upošteva njeno vertikalno pozicijo

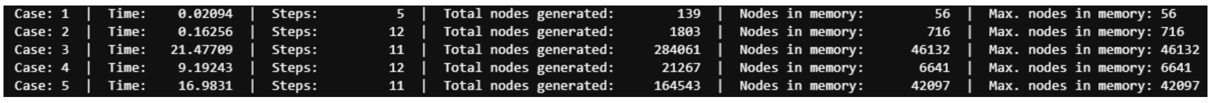
Algoritem A* sem implementiral s pomočjo prioritetne vrste (priority queue), ki sem jo uporabil kot »open« in množico (set), ki sem jo uporabil kot »closed«. Algoritem sem pognal z vsemi hevristikami, a samo z hevristiko »misplaced« sem dobil optimalno pot na vseh primerih.

Uporabljena hevristika: row



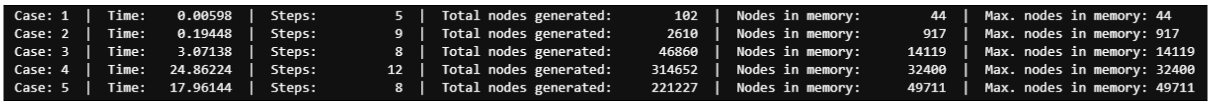
```
Case: 1 |  Time:    0.00698 |  Steps:          5 |  Total nodes generated:        98 |  Nodes in memory:         50 |  Max. nodes in memory: 50
Case: 2 |  Time:     0.1506 |  Steps:          9 |  Total nodes generated:      2274 |  Nodes in memory:        785 |  Max. nodes in memory: 785
Case: 3 |  Time:    5.46938 |  Steps:         12 |  Total nodes generated:     71555 |  Nodes in memory:      21764 |  Max. nodes in memory: 21764
Case: 4 |  Time:   20.09229 |  Steps:         12 |  Total nodes generated:    305994 |  Nodes in memory:      32400 |  Max. nodes in memory: 32400
Case: 5 |  Time:  135.81296 |  Steps:          9 |  Total nodes generated:   1404254 |  Nodes in memory:     133185 |  Max. nodes in memory: 133185
```

Uporabljena hevristika: column



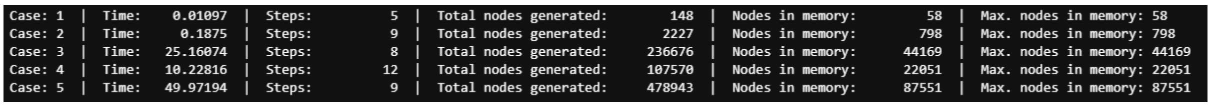
```
Case: 1 |  Time:    0.02094 |  Steps:          5 |  Total nodes generated:       139 |  Nodes in memory:         56 |  Max. nodes in memory: 56
Case: 2 |  Time:    0.16256 |  Steps:         12 |  Total nodes generated:      1803 |  Nodes in memory:        716 |  Max. nodes in memory: 716
Case: 3 |  Time:   21.47709 |  Steps:         11 |  Total nodes generated:    284061 |  Nodes in memory:      46132 |  Max. nodes in memory: 46132
Case: 4 |  Time:    9.19243 |  Steps:         12 |  Total nodes generated:     21267 |  Nodes in memory:       6641 |  Max. nodes in memory: 6641
Case: 5 |  Time:    16.9831 |  Steps:         11 |  Total nodes generated:    164543 |  Nodes in memory:      42097 |  Max. nodes in memory: 42097
```

Uporabljena hevristika: misplaced



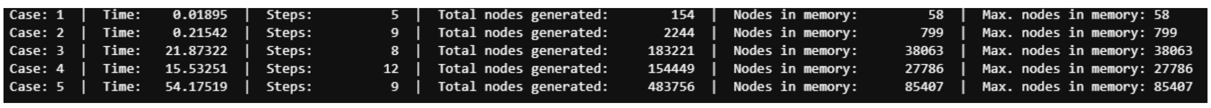
```
Case: 1 |  Time:    0.00598 |  Steps:          5 |  Total nodes generated:       102 |  Nodes in memory:         44 |  Max. nodes in memory: 44
Case: 2 |  Time:    0.19448 |  Steps:          9 |  Total nodes generated:      2610 |  Nodes in memory:        917 |  Max. nodes in memory: 917
Case: 3 |  Time:    3.07138 |  Steps:          8 |  Total nodes generated:     46860 |  Nodes in memory:      14119 |  Max. nodes in memory: 14119
Case: 4 |  Time:   24.86224 |  Steps:         12 |  Total nodes generated:    314652 |  Nodes in memory:      32400 |  Max. nodes in memory: 32400
Case: 5 |  Time:   17.96144 |  Steps:          8 |  Total nodes generated:    221227 |  Nodes in memory:      49711 |  Max. nodes in memory: 49711
```

Uporabljena hevristika: manhattan



```
Case: 1 |  Time:    0.01097 |  Steps:          5 |  Total nodes generated:       148 |  Nodes in memory:         58 |  Max. nodes in memory: 58
Case: 2 |  Time:     0.1875 |  Steps:          9 |  Total nodes generated:      2227 |  Nodes in memory:        798 |  Max. nodes in memory: 798
Case: 3 |  Time:   25.16074 |  Steps:          8 |  Total nodes generated:    236676 |  Nodes in memory:      44169 |  Max. nodes in memory: 44169
Case: 4 |  Time:   10.22816 |  Steps:         12 |  Total nodes generated:    107570 |  Nodes in memory:      22051 |  Max. nodes in memory: 22051
Case: 5 |  Time:   49.97194 |  Steps:          9 |  Total nodes generated:    478943 |  Nodes in memory:      87551 |  Max. nodes in memory: 87551
```

Uporabljena hevristika: euclidean

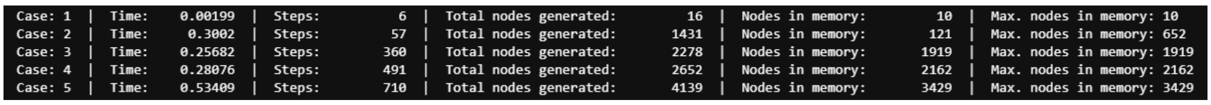


```
Case: 1 |  Time:    0.01895 |  Steps:          5 |  Total nodes generated:       154 |  Nodes in memory:         58 |  Max. nodes in memory: 58
Case: 2 |  Time:    0.21542 |  Steps:          9 |  Total nodes generated:      2244 |  Nodes in memory:        799 |  Max. nodes in memory: 799
Case: 3 |  Time:   21.87322 |  Steps:          8 |  Total nodes generated:    183221 |  Nodes in memory:      38063 |  Max. nodes in memory: 38063
Case: 4 |  Time:   15.53251 |  Steps:         12 |  Total nodes generated:    154449 |  Nodes in memory:      27786 |  Max. nodes in memory: 27786
Case: 5 |  Time:   54.17519 |  Steps:          9 |  Total nodes generated:    483756 |  Nodes in memory:      85407 |  Max. nodes in memory: 85407
```

2.5. Hill Climbing

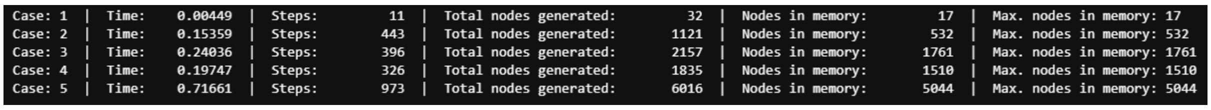
Algoritem sem implementiral z uporaba sklada (stack) na katerega sem dodajal glede na hevristično oceno urejene naslednike, tako kot A* sem tudi ta algoritem izvedel z vsemi hevristikami in tudi tukaj dobil različne rezultate.

Uporabljena hevristika: row



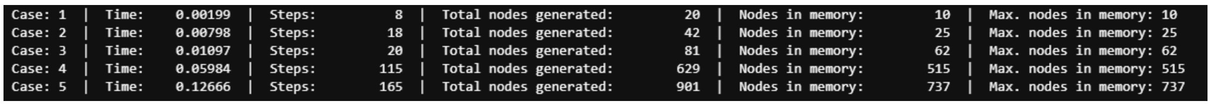
```
Case: 1 | Time:    0.00199 | Steps:         6 | Total nodes generated:       16 | Nodes in memory:        10 | Max. nodes in memory: 10
Case: 2 | Time:     0.3002 | Steps:        57 | Total nodes generated:     1431 | Nodes in memory:       121 | Max. nodes in memory: 652
Case: 3 | Time:    0.25682 | Steps:       360 | Total nodes generated:     2278 | Nodes in memory:      1919 | Max. nodes in memory: 1919
Case: 4 | Time:    0.28076 | Steps:       491 | Total nodes generated:     2652 | Nodes in memory:      2162 | Max. nodes in memory: 2162
Case: 5 | Time:    0.53409 | Steps:       710 | Total nodes generated:     4139 | Nodes in memory:      3429 | Max. nodes in memory: 3429
```

Uporabljena hevristika: column



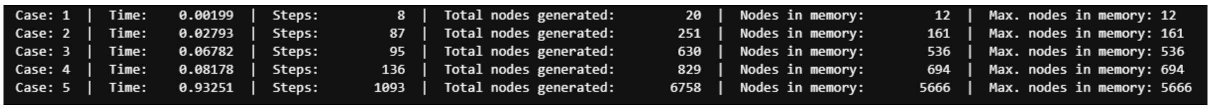
```
Case: 1 | Time:    0.00449 | Steps:        11 | Total nodes generated:       32 | Nodes in memory:        17 | Max. nodes in memory: 17
Case: 2 | Time:    0.15359 | Steps:       443 | Total nodes generated:     1121 | Nodes in memory:       532 | Max. nodes in memory: 532
Case: 3 | Time:    0.24036 | Steps:       396 | Total nodes generated:     2157 | Nodes in memory:      1761 | Max. nodes in memory: 1761
Case: 4 | Time:    0.19747 | Steps:       326 | Total nodes generated:     1835 | Nodes in memory:      1510 | Max. nodes in memory: 1510
Case: 5 | Time:    0.71661 | Steps:       973 | Total nodes generated:     6016 | Nodes in memory:      5044 | Max. nodes in memory: 5044
```

Uporabljena hevristika: misplaced



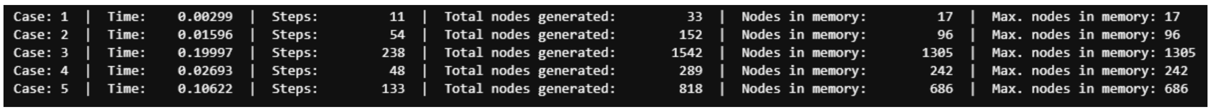
```
Case: 1 | Time:    0.00199 | Steps:         8 | Total nodes generated:       20 | Nodes in memory:        10 | Max. nodes in memory: 10
Case: 2 | Time:    0.00798 | Steps:        18 | Total nodes generated:       42 | Nodes in memory:        25 | Max. nodes in memory: 25
Case: 3 | Time:    0.01097 | Steps:        20 | Total nodes generated:       81 | Nodes in memory:        62 | Max. nodes in memory: 62
Case: 4 | Time:    0.05984 | Steps:       115 | Total nodes generated:      629 | Nodes in memory:       515 | Max. nodes in memory: 515
Case: 5 | Time:    0.12666 | Steps:       165 | Total nodes generated:      901 | Nodes in memory:       737 | Max. nodes in memory: 737
```

Uporabljena hevristika: manhattan



```
Case: 1 | Time:    0.00199 | Steps:         8 | Total nodes generated:       20 | Nodes in memory:        12 | Max. nodes in memory: 12
Case: 2 | Time:    0.02793 | Steps:        87 | Total nodes generated:      251 | Nodes in memory:       161 | Max. nodes in memory: 161
Case: 3 | Time:    0.06782 | Steps:        95 | Total nodes generated:      630 | Nodes in memory:       536 | Max. nodes in memory: 536
Case: 4 | Time:    0.08178 | Steps:       136 | Total nodes generated:      829 | Nodes in memory:       694 | Max. nodes in memory: 694
Case: 5 | Time:    0.93251 | Steps:      1093 | Total nodes generated:     6758 | Nodes in memory:      5666 | Max. nodes in memory: 5666
```

Uporabljena hevristika: euclidean

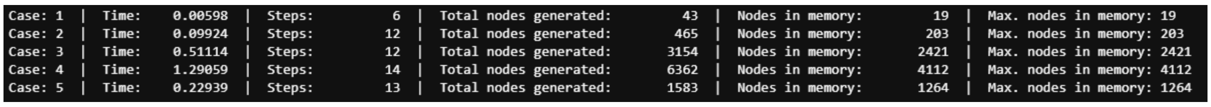


```
Case: 1 | Time:    0.00299 | Steps:        11 | Total nodes generated:       33 | Nodes in memory:        17 | Max. nodes in memory: 17
Case: 2 | Time:    0.01596 | Steps:        54 | Total nodes generated:      152 | Nodes in memory:        96 | Max. nodes in memory: 96
Case: 3 | Time:    0.19997 | Steps:       238 | Total nodes generated:     1542 | Nodes in memory:      1305 | Max. nodes in memory: 1305
Case: 4 | Time:    0.02693 | Steps:        48 | Total nodes generated:      289 | Nodes in memory:       242 | Max. nodes in memory: 242
Case: 5 | Time:    0.10622 | Steps:       133 | Total nodes generated:      818 | Nodes in memory:       686 | Max. nodes in memory: 686
```

2.6. Best First

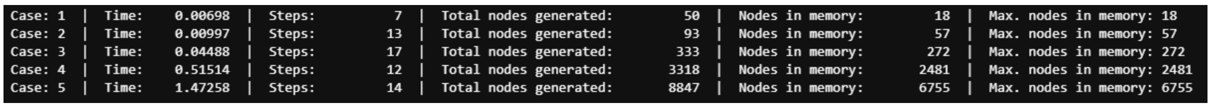
Algoritem sem implementiral z uporaba prioritetne vrste (priority queue)v katero dodajam naslednike, njihovo prioriteto pa določa hevristična ocena, tako kot A* sem tudi ta algoritem izvedel z vsemi hevristikami in tudi tukaj dobil različne rezultate.

Uporabljena hevristika: row



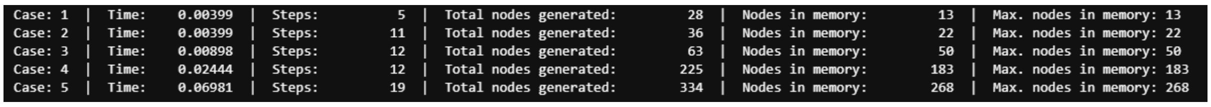
```
Case: 1  |  Time:    0.00598  |  Steps:          6  |  Total nodes generated:       43  |  Nodes in memory:       19  |  Max. nodes in memory: 19
Case: 2  |  Time:    0.09924  |  Steps:         12  |  Total nodes generated:      465  |  Nodes in memory:      203  |  Max. nodes in memory: 203
Case: 3  |  Time:    0.51114  |  Steps:         12  |  Total nodes generated:     3154  |  Nodes in memory:     2421  |  Max. nodes in memory: 2421
Case: 4  |  Time:    1.29059  |  Steps:         14  |  Total nodes generated:     6362  |  Nodes in memory:     4112  |  Max. nodes in memory: 4112
Case: 5  |  Time:    0.22939  |  Steps:         13  |  Total nodes generated:     1583  |  Nodes in memory:     1264  |  Max. nodes in memory: 1264
```

Uporabljena hevristika: column



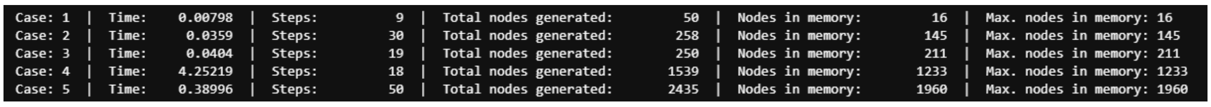
```
Case: 1  |  Time:    0.00698  |  Steps:          7  |  Total nodes generated:       50  |  Nodes in memory:       18  |  Max. nodes in memory: 18
Case: 2  |  Time:    0.00997  |  Steps:         13  |  Total nodes generated:       93  |  Nodes in memory:       57  |  Max. nodes in memory: 57
Case: 3  |  Time:    0.04488  |  Steps:         17  |  Total nodes generated:      333  |  Nodes in memory:      272  |  Max. nodes in memory: 272
Case: 4  |  Time:    0.51514  |  Steps:         12  |  Total nodes generated:     3318  |  Nodes in memory:     2481  |  Max. nodes in memory: 2481
Case: 5  |  Time:    1.47258  |  Steps:         14  |  Total nodes generated:     8847  |  Nodes in memory:     6755  |  Max. nodes in memory: 6755
```

Uporabljena hevristika: misplaced



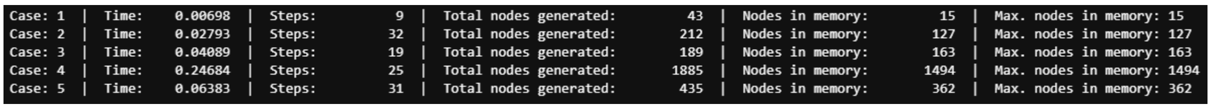
```
Case: 1  |  Time:    0.00399  |  Steps:          5  |  Total nodes generated:       28  |  Nodes in memory:       13  |  Max. nodes in memory: 13
Case: 2  |  Time:    0.00399  |  Steps:         11  |  Total nodes generated:       36  |  Nodes in memory:       22  |  Max. nodes in memory: 22
Case: 3  |  Time:    0.00898  |  Steps:         12  |  Total nodes generated:       63  |  Nodes in memory:       50  |  Max. nodes in memory: 50
Case: 4  |  Time:    0.02444  |  Steps:         12  |  Total nodes generated:      225  |  Nodes in memory:      183  |  Max. nodes in memory: 183
Case: 5  |  Time:    0.06981  |  Steps:         19  |  Total nodes generated:      334  |  Nodes in memory:      268  |  Max. nodes in memory: 268
```

Uporabljena hevristika: manhattan



```
Case: 1  |  Time:    0.00798  |  Steps:          9  |  Total nodes generated:       50  |  Nodes in memory:       16  |  Max. nodes in memory: 16
Case: 2  |  Time:     0.0359  |  Steps:         30  |  Total nodes generated:      258  |  Nodes in memory:      145  |  Max. nodes in memory: 145
Case: 3  |  Time:     0.0404  |  Steps:         19  |  Total nodes generated:      250  |  Nodes in memory:      211  |  Max. nodes in memory: 211
Case: 4  |  Time:    4.25219  |  Steps:         18  |  Total nodes generated:     1539  |  Nodes in memory:     1233  |  Max. nodes in memory: 1233
Case: 5  |  Time:    0.38996  |  Steps:         50  |  Total nodes generated:     2435  |  Nodes in memory:     1960  |  Max. nodes in memory: 1960
```

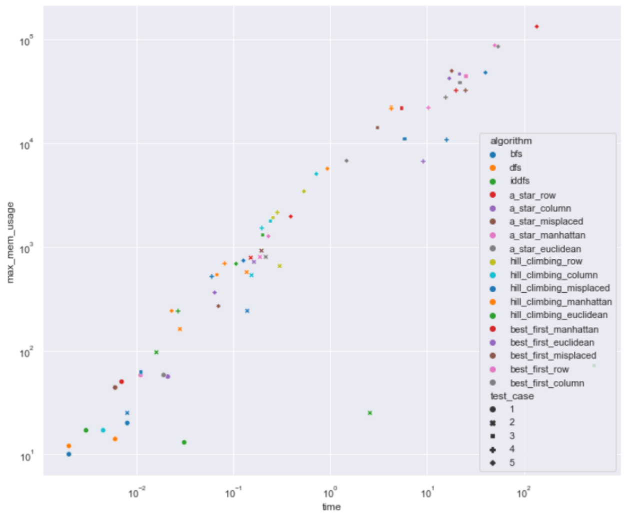
Uporabljena hevristika: euclidean



```
Case: 1  |  Time:    0.00698  |  Steps:          9  |  Total nodes generated:       43  |  Nodes in memory:       15  |  Max. nodes in memory: 15
Case: 2  |  Time:    0.02793  |  Steps:         32  |  Total nodes generated:      212  |  Nodes in memory:      127  |  Max. nodes in memory: 127
Case: 3  |  Time:    0.04089  |  Steps:         19  |  Total nodes generated:      189  |  Nodes in memory:      163  |  Max. nodes in memory: 163
Case: 4  |  Time:    0.24684  |  Steps:         25  |  Total nodes generated:     1885  |  Nodes in memory:     1494  |  Max. nodes in memory: 1494
Case: 5  |  Time:    0.06383  |  Steps:         31  |  Total nodes generated:      435  |  Nodes in memory:      362  |  Max. nodes in memory: 362
```

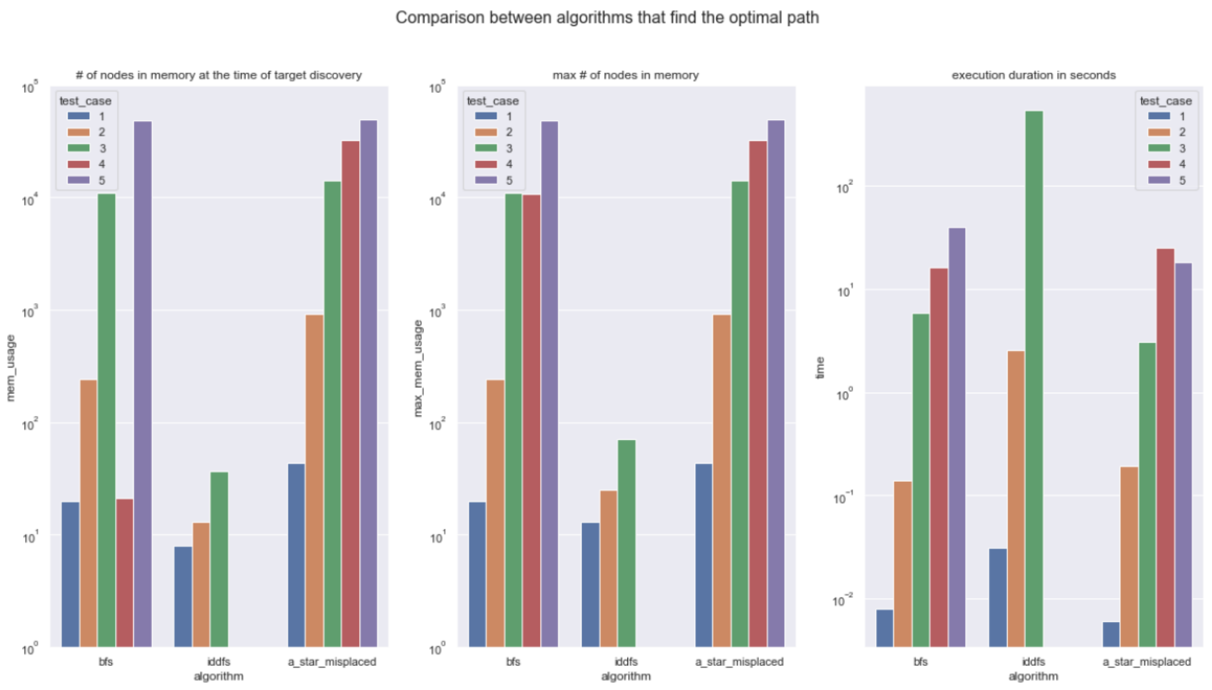
3. Primerjava učinkovitosti algoritmov

Na spodnjem grafu lahko približno razberemo, da sta maksimalna poraba pomnilnika in časovna zahtevnost algoritmov linearno povezana.



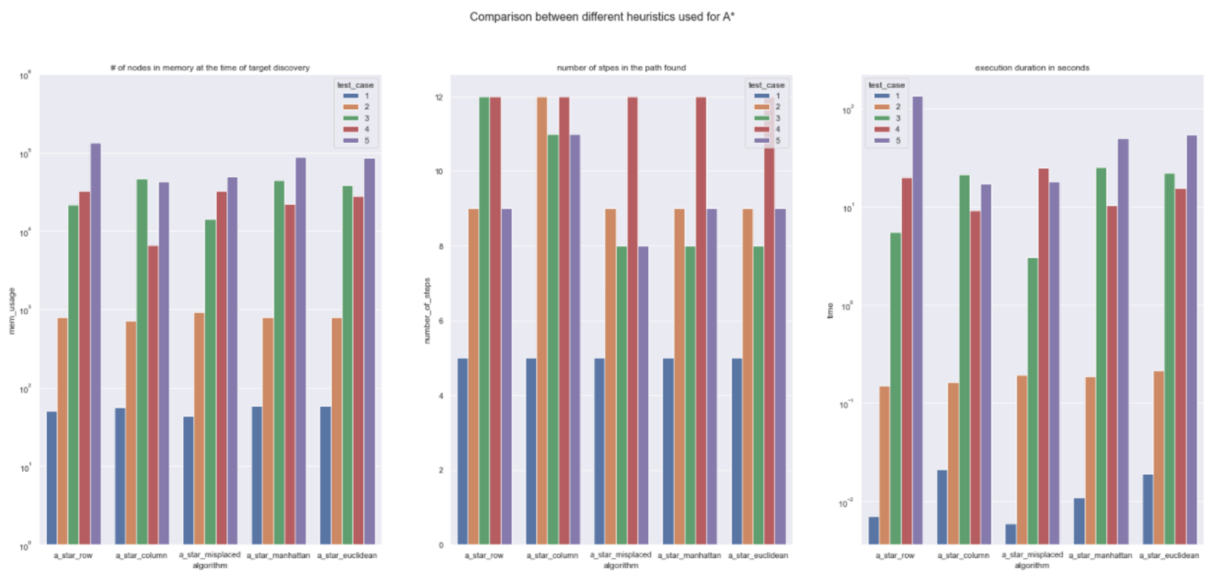
3.1. Primerjava algoritmov, ki dobijo optimalno pot

Iz spodnjih grafov (skale so logaritemske zaradi velikega razpona vrednosti) lahko razberemo, da je prostorska zahtevnost najboljša (najmanjša) pri algoritmu IDDFS, časovna zahtevnost je pa odvisno od primera boljša pri BFS ali pa A*, sklepam da bi A* pri bolj kompleksnih (večjih) primerih imel veliko boljšo časovno zahtevnost v primerjavi z BFS, pri primerih, kjer pa je rešitev na dokaj majhni globini, pa je BFS boljši, zardi manjšega »overhead-a« (računanje hevristike, premikanje med open in closed, …), to razliko bi se mogoče dalo rešiti z izbiro boljše hevristike, a se sam nisem spomnil boljše.



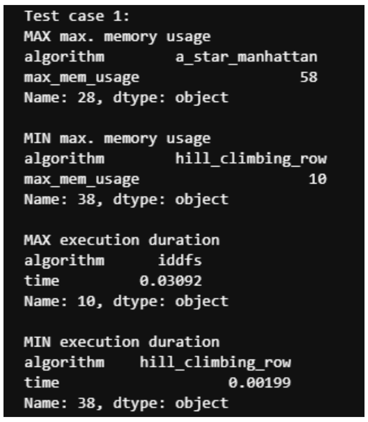
3.2. Primerjava učinkovitosti algoritma A* z uporabo različnih hevristik

Iz spodnjih grafov (skale so logaritemske zaradi velikega razpona vrednosti) lahko razberemo, da izbira dobre hevristike pripomore k boljši prostorski in časovni zahtevnosti, poleg tega pa samo ena od hevristik (»misplaced«) najde optimalno rešitev. Različne hevristike sicer na večini primerih dobijo optimalno pot, ne pa na vseh.



3.1. Časovno in prostorsko najbolj učinkoviti algoritmi (optimalni in neoptimalni) na posameznih primerih

Primer 1:



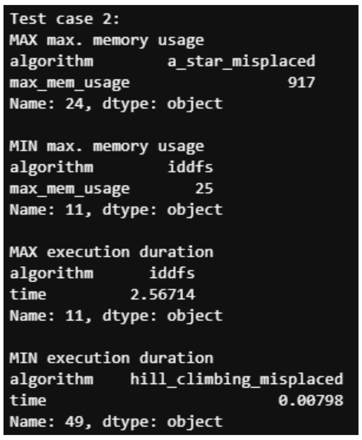
```
Test case 1:
MAX max. memory usage
algorithm        a_star_manhattan
max_mem_usage                  58
Name: 28, dtype: object

MIN max. memory usage
algorithm        hill_climbing_row
max_mem_usage                   10
Name: 38, dtype: object

MAX execution duration
algorithm      iddfs
time         0.03092
Name: 10, dtype: object

MIN execution duration
algorithm    hill_climbing_row
time                   0.00199
Name: 38, dtype: object
```

Primer 2:



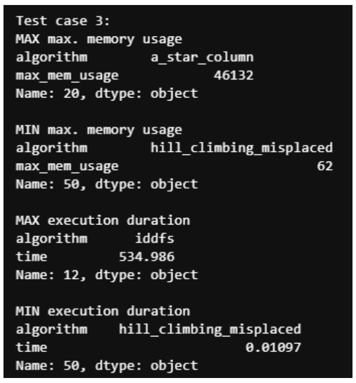
```
Test case 2:
MAX max. memory usage
algorithm        a_star_misplaced
max_mem_usage                 917
Name: 24, dtype: object

MIN max. memory usage
algorithm        iddfs
max_mem_usage       25
Name: 11, dtype: object

MAX execution duration
algorithm      iddfs
time         2.56714
Name: 11, dtype: object

MIN execution duration
algorithm    hill_climbing_misplaced
time                         0.00798
Name: 49, dtype: object
```

Primer 3:



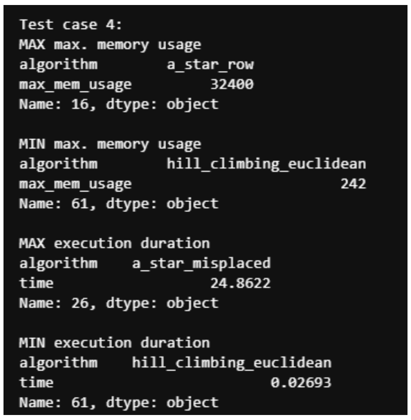
```
Test case 3:
MAX max. memory usage
algorithm        a_star_column
max_mem_usage            46132
Name: 20, dtype: object

MIN max. memory usage
algorithm        hill_climbing_misplaced
max_mem_usage                         62
Name: 50, dtype: object

MAX execution duration
algorithm      iddfs
time         534.986
Name: 12, dtype: object

MIN execution duration
algorithm    hill_climbing_misplaced
time                         0.01097
Name: 50, dtype: object
```

Primer 4:



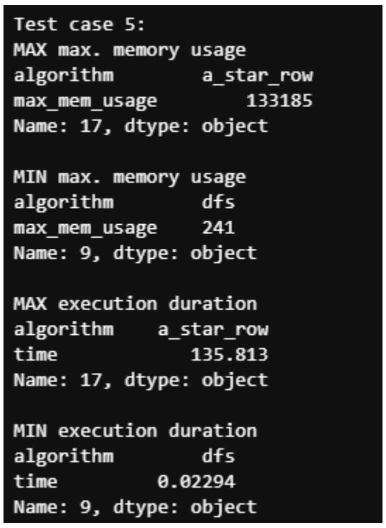
```
Test case 4:
MAX max. memory usage
algorithm        a_star_row
max_mem_usage         32400
Name: 16, dtype: object

MIN max. memory usage
algorithm        hill_climbing_euclidean
max_mem_usage                        242
Name: 61, dtype: object

MAX execution duration
algorithm    a_star_misplaced
time                  24.8622
Name: 26, dtype: object

MIN execution duration
algorithm    hill_climbing_euclidean
time                         0.02693
Name: 61, dtype: object
```

Primer 5:



```
Test case 5:
MAX max. memory usage
algorithm        a_star_row
max_mem_usage        133185
Name: 17, dtype: object

MIN max. memory usage
algorithm        dfs
max_mem_usage    241
Name: 9, dtype: object

MAX execution duration
algorithm    a_star_row
time            135.813
Name: 17, dtype: object

MIN execution duration
algorithm        dfs
time         0.02294
Name: 9, dtype: object
```