Seminarska naloga 2

Created	@January 8, 2022 11:59 AM
Class	UI
Type	Exam
Letnik	2
Semester	1

Povezave

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/99cfb413-0ff2-4314-b90d-e03d1f0d6b01/Umetna_inteligenca_Seminarska_2.pdf

https://github.com/bartolomej/fri-artificial-intelligence/tree/main/seminar-2

https://bartolomej.github.io/fri-artificial-intelligence/seminar-2/scripts/labyrinth.html

Opis pristopa

Podan labirint lahko pretvorimo v graf za katerega definiramo tudi zacetno in koncno vozlisce, ter seznam vmesnih vozlisc (zakladov), ki jih moramo obiskati na podi od zacetnega do koncnega vozlisca.

Cilj naloge je torej najti optimalno zaporedje vmesnih vozlisc (oz. pot v grafu), ki minimizira ceno sprehoda.

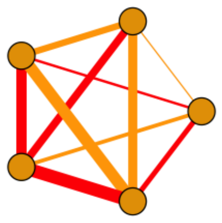
K problemu sem se odlocil pristopiti tako, da sem razdelil celoten problem na dva dela:

1. iskanje najkrejse poti med posameznimi vozlisci, ki nas zanimajo (zacetno, koncno in vmesna)
2. iskanje optimalnega zaporedja vmesnih vozlisc, glede na dobljeno mnozico najkrajsih poti med vozlisci

Prvi problem lahko resimo z metodami iskanja najkrajse poti v grafih (bfs, dfs, A*,..), drug problem pa je v resnici problem trgovskega potnika za katerega obstaja nekaj hevristicnih nacinov resevanja.

Vseh moznih poti med vsemi vmesnimi vozlisci n je enako $(n+1)! + 1$, saj v prvem koraku izbiramo optimalno pot od zacetnega vozlisca do enega izmed n-tih vozlisc, v zadnjem pa iz enega od notranjih do koncnega vozlisca (+ 1).

Ko izracunamo vse mozne najkrajse povezave med notranjimi vozlisci, dobimo nov polno povezan podgraf, katerega lahko vizualiziramo na spodnji nacin.



V tem grafu moramo najti hamiltonovo pot, ki minimizira skupno ceno vseh povezav.

Rezultati

Testiral sem na dveh algoritmih za iskanje poti v grafih (bfs, dfs), ter na dveh algoritmih za iskanje optimalnega zaporedja povezav med vmesnimi vozlisci (lokalno optimalno iskanje, pozresno iskanje).

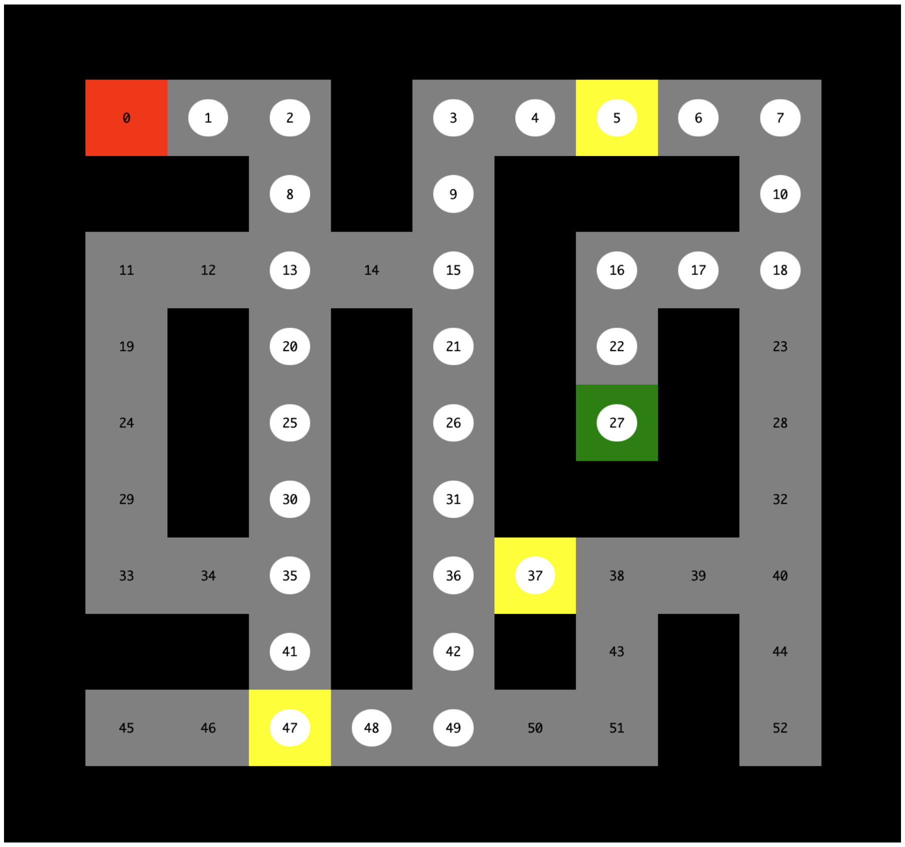
Vse vizualizacije rezultatov so generirane z zgrajeno spletno stranjo, na povezavi: https://bartolomej.github.io/fri-artificial-intelligence/seminar-2/scripts/labyrinth.html

💡 V spodnjih vizualizacijah je najdena pot oznacena z belimi krogi, stevilke na vozliscih pa predstavljajo oznako vozlisca v matriki sosednjosti, ki predstavlja dolocen graf labirinta.

Najboljo resitev dosezemo pri kombinaciji DFS algoritma in metode lokalne optimizacije.

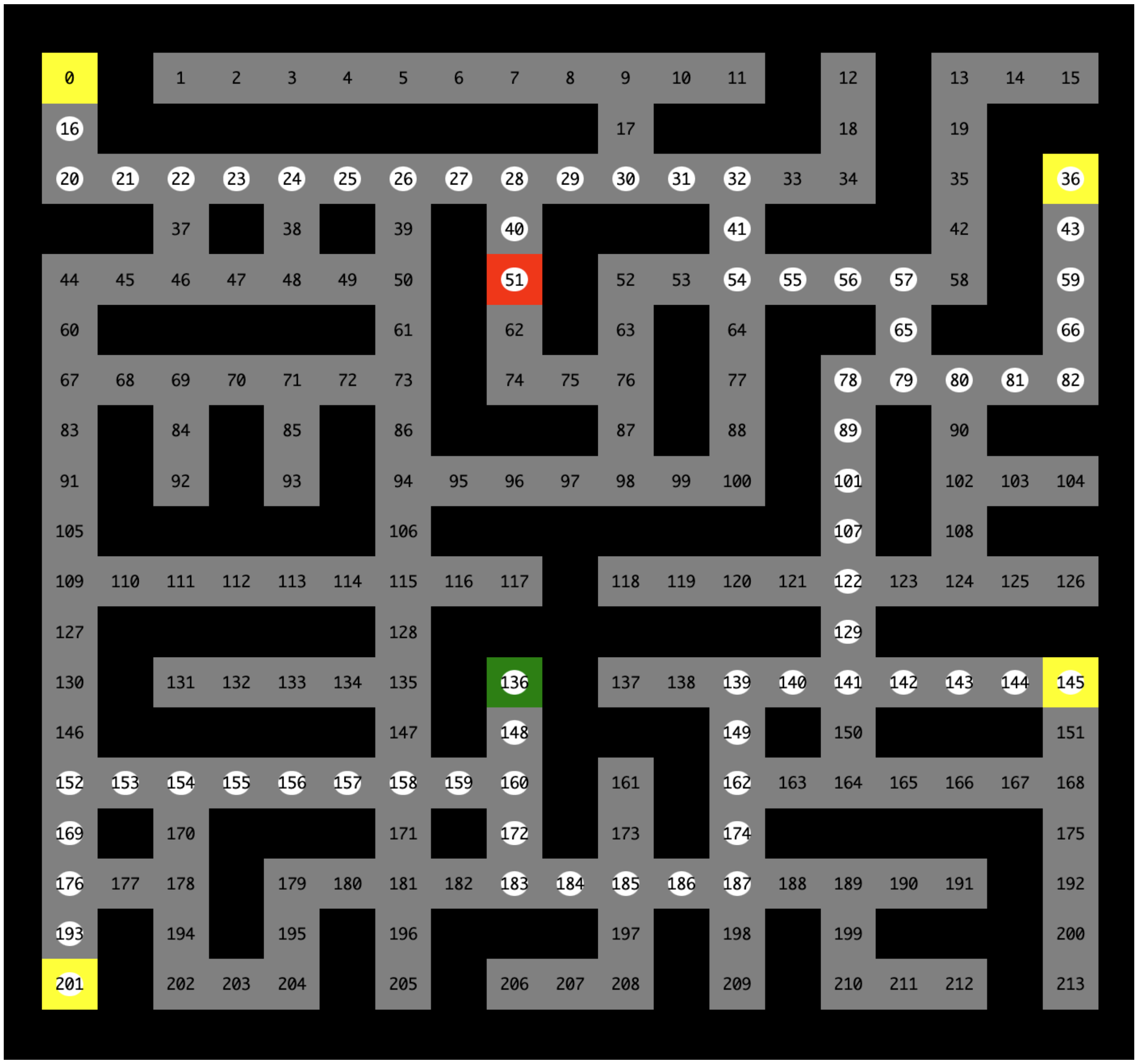
▼ Labirint 1 (min cena=116)

```
ALGORITHM: dfs
GREEDY (cost=299, path=0,1,2,8,13,12,11,19,24,29,33,34,35,41,47,47,41,35,34,33,29,24,19,11,12,13,14,15,9,3,4,5,5,4,3,9,15,14,13,12,11,1
LOCAL OPTIMUM (cost=254, path=0,1,2,8,13,12,11,19,24,29,33,34,35,41,47,47,48,49,50,51,43,38,39,40,32,28,23,18,10,7,6,5,4,3,9,15,21,26,3

ALGORITHM: bfs
GREEDY (cost=145, path=0,1,2,8,13,14,15,9,3,4,5,5,4,3,9,15,21,26,31,36,37,37,36,42,49,48,47,47,48,49,42,36,37,38,39,40,32,28,23,18,17,1
LOCAL OPTIMUM (cost=116, path=0,1,2,8,13,20,25,30,35,41,47,47,48,49,42,36,37,37,36,31,26,21,15,9,3,4,5,5,6,7,10,18,17,16,22,27)
```



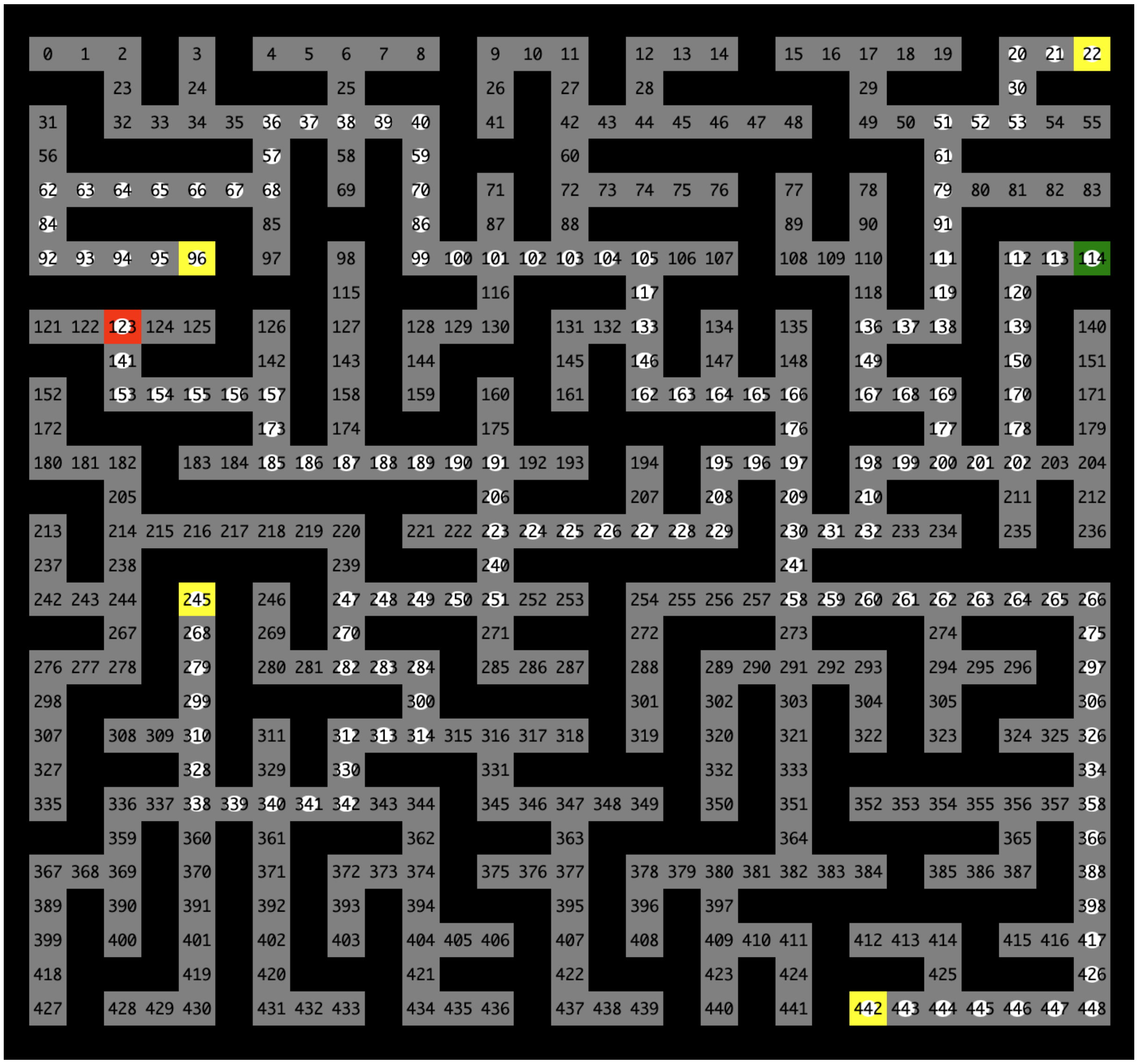
▼ Labirint 2 (min cena=385)

```
ALGORITHM: dfs
GREEDY (cost=1121, path=51,40,28,27,26,25,24,23,22,21,20,16,0,0,16,20,21,22,23,24,25,26,27,28,29,30,31,32,41,54,53,52,63,76,87,98,97,96
LOCAL OPTIMUM (cost=1121, path=51,40,28,27,26,25,24,23,22,21,20,16,0,0,16,20,21,22,23,24,25,26,27,28,29,30,31,32,41,54,53,52,63,76,87,9

ALGORITHM: bfs
GREEDY (cost=417, path=51,40,28,27,26,25,24,23,22,21,20,16,0,0,16,20,21,22,37,46,45,44,60,67,83,91,105,109,127,130,146,152,169,176,193,
LOCAL OPTIMUM (cost=385, path=51,40,28,27,26,25,24,23,22,21,20,16,0,0,16,20,21,22,23,24,25,26,27,28,29,30,31,32,41,54,55,56,57,65,79,80
```



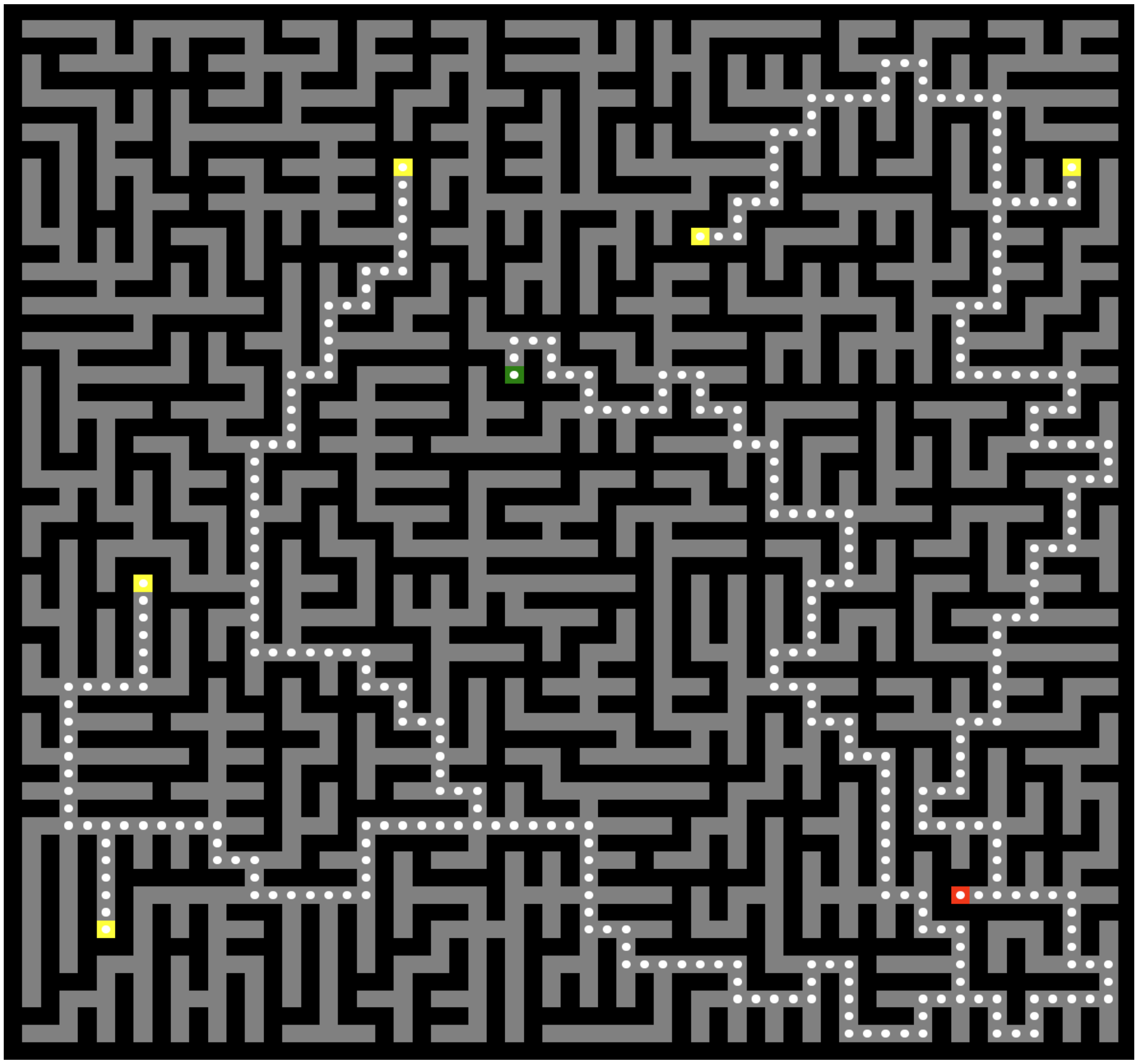
▼ Labirint 3 (min cena=1057)

```
ALGORITHM: dfs
GREEDY (cost=1121, path=123,141,153,154,155,156,157,173,185,186,187,188,189,190,191,206,223,240,251,250,249,248,247,270,282,283,284,300
LOCAL OPTIMUM (cost=1057, path=123,141,153,154,155,156,157,173,185,186,187,188,189,190,191,206,223,240,251,250,249,248,247,270,282,283,

ALGORITHM: bfs
GREEDY (cost=1121, path=123,141,153,154,155,156,157,173,185,186,187,188,189,190,191,206,223,240,251,250,249,248,247,270,282,283,284,300
LOCAL OPTIMUM (cost=1057, path=123,141,153,154,155,156,157,173,185,186,187,188,189,190,191,206,223,240,251,250,249,248,247,270,282,283,
```



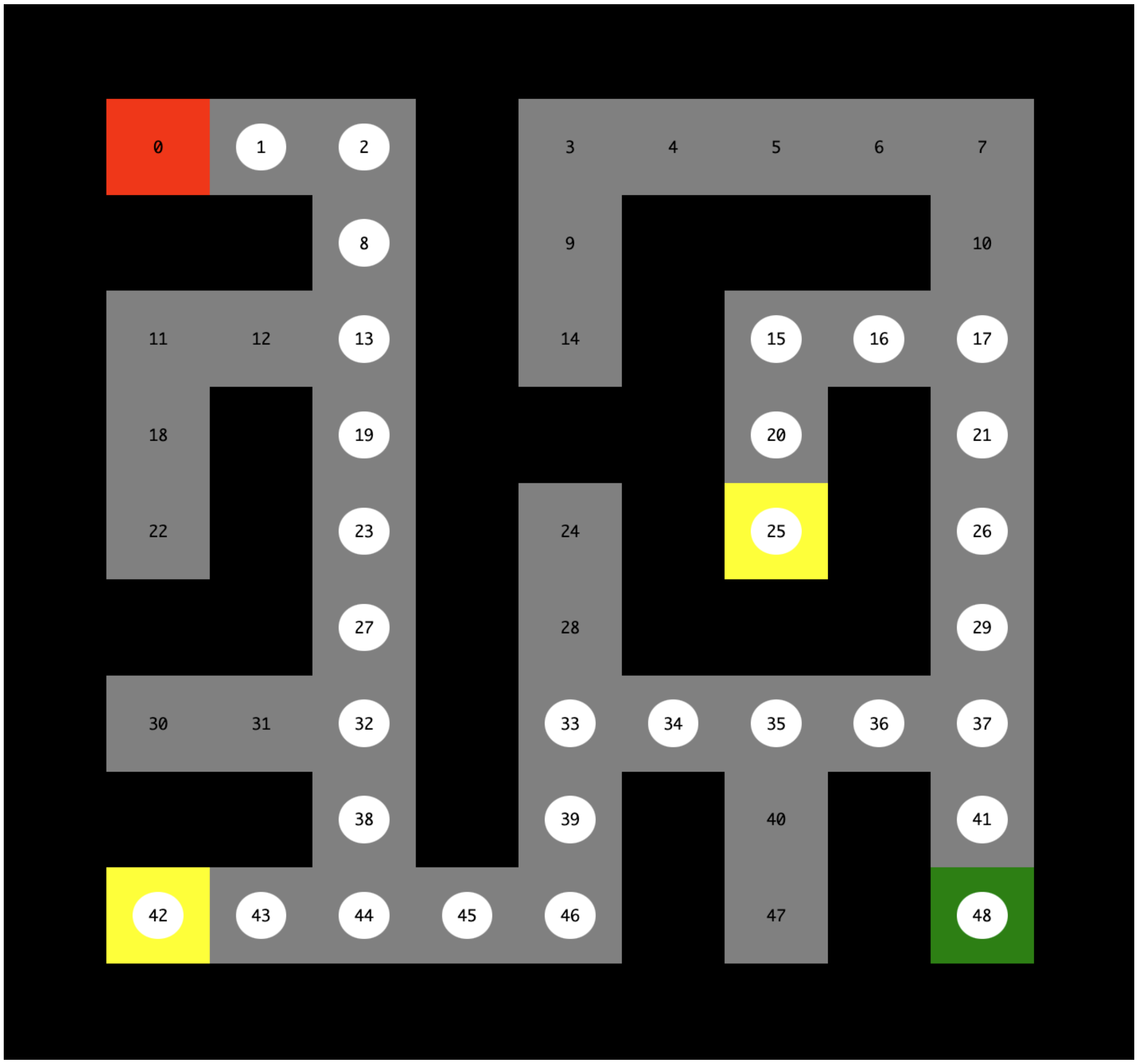
▼ Labirint 4 (min cena=2502)

```
ALGORITHM: dfs
GREEDY (cost=2502, path=1545,1546,1547,1501,1482,1448,1427,1426,1425,1424,1423,1377,1359,1360,1361,1320,1303,1261,1240,1241,1242,1199,1
LOCAL OPTIMUM (cost=2502, path=1545,1546,1547,1501,1482,1448,1427,1426,1425,1424,1423,1377,1359,1360,1361,1320,1303,1261,1240,1241,1242

ALGORITHM: bfs
GREEDY (cost=2502, path=1545,1546,1547,1501,1482,1448,1427,1426,1425,1424,1423,1377,1359,1360,1361,1320,1303,1261,1240,1241,1242,1199,1
LOCAL OPTIMUM (cost=2502, path=1545,1546,1547,1501,1482,1448,1427,1426,1425,1424,1423,1377,1359,1360,1361,1320,1303,1261,1240,1241,1242
```



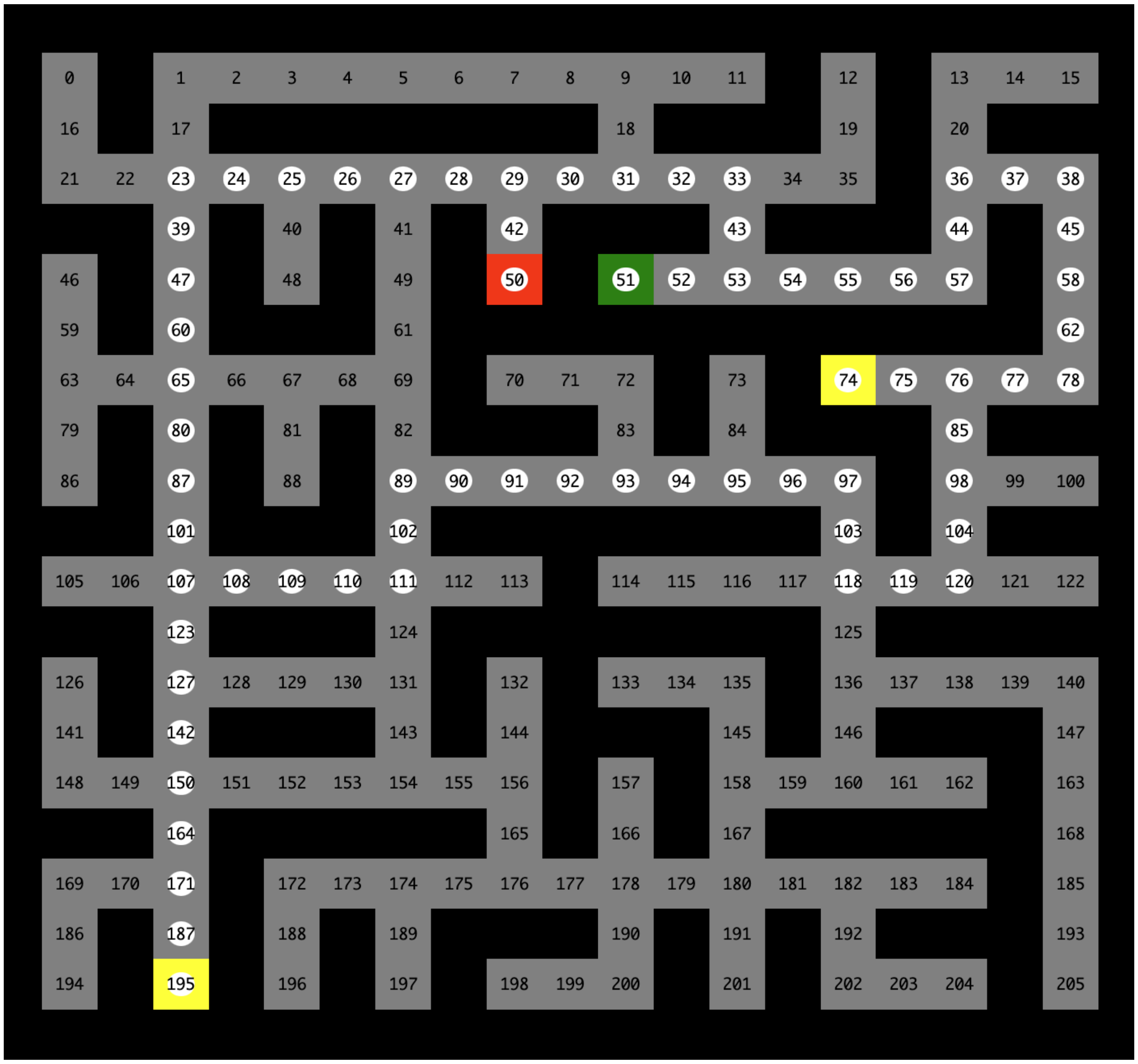
▼ Labirint 5 (min cena=87)

```
ALGORITHM: dfs
GREEDY (cost=87, path=0,1,2,8,13,19,23,27,32,38,44,43,42,42,43,44,45,46,39,33,34,35,36,37,29,26,21,17,16,15,20,25,25,20,15,16,17,21,26,
LOCAL OPTIMUM (cost=87, path=0,1,2,8,13,19,23,27,32,38,44,43,42,25,20,15,16,17,21,26,29,37,36,35,34,33,39,46,45,44,43,42,48,41,37,29,26

ALGORITHM: bfs
GREEDY (cost=87, path=0,1,2,8,13,19,23,27,32,38,44,43,42,42,43,44,45,46,39,33,34,35,36,37,29,26,21,17,16,15,20,25,25,20,15,16,17,21,26,
LOCAL OPTIMUM (cost=87, path=0,1,2,8,13,19,23,27,32,38,44,43,42,25,20,15,16,17,21,26,29,37,36,35,34,33,39,46,45,44,43,42,48,41,37,29,26
```



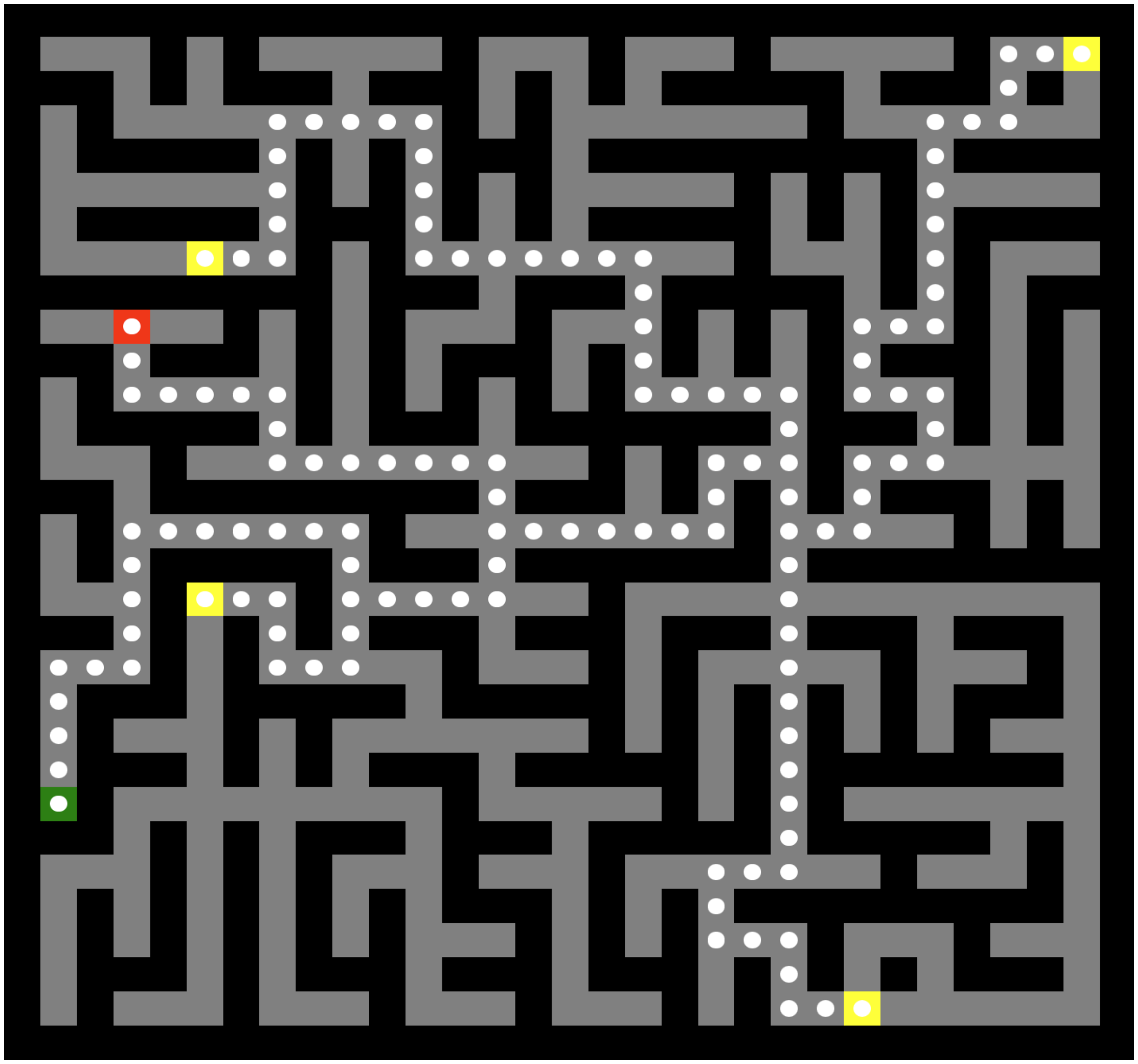
▼ Labirint 6 (min cena=171)

```
ALGORITHM: dfs
GREEDY (cost=525, path=50,42,29,28,27,26,25,24,23,17,1,2,3,4,5,6,7,8,9,18,31,32,33,43,53,54,55,56,57,44,36,37,38,45,58,62,78,77,76,75,7
LOCAL OPTIMUM (cost=525, path=50,42,29,28,27,26,25,24,23,17,1,2,3,4,5,6,7,8,9,18,31,32,33,43,53,54,55,56,57,44,36,37,38,45,58,62,78,77,

ALGORITHM: bfs
GREEDY (cost=213, path=50,42,29,30,31,32,33,43,53,54,55,56,57,44,36,37,38,45,58,62,78,77,76,75,74,74,75,76,85,98,104,120,119,118,103,97
LOCAL OPTIMUM (cost=171, path=50,42,29,28,27,26,25,24,23,39,47,60,65,80,87,101,107,123,127,142,150,164,171,187,195,195,187,171,164,150,
```



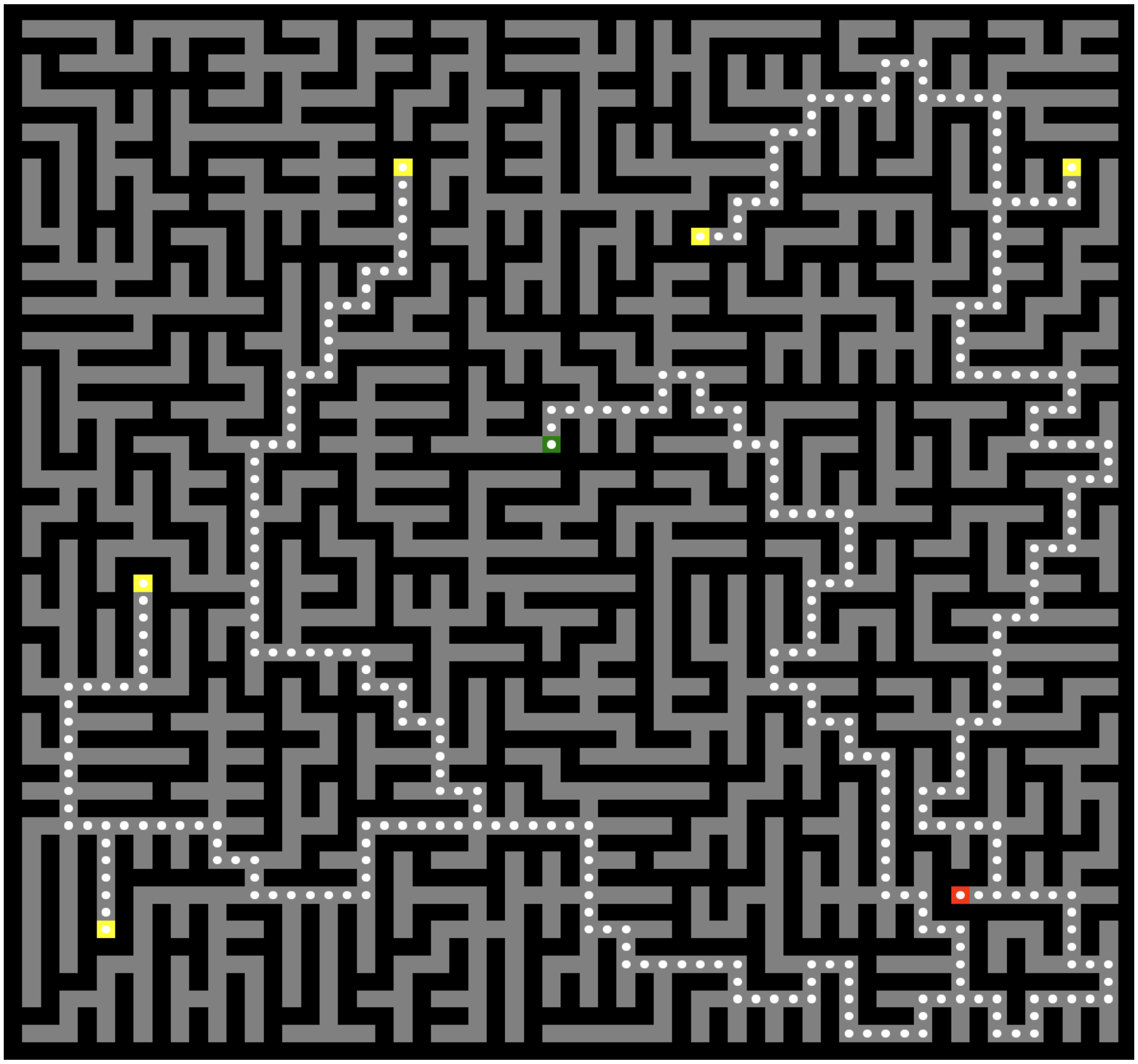
▼ Labirint 7 (min cena=586)

```
ALGORITHM: dfs
GREEDY (cost=687, path=125,143,155,156,157,158,159,175,187,188,189,190,191,192,193,208,225,242,254,253,252,251,250,273,285,284,283,272,
LOCAL OPTIMUM (cost=656, path=125,143,155,156,157,158,159,175,187,188,189,190,191,192,193,208,225,226,227,228,229,230,231,210,197,198,1

ALGORITHM: bfs
GREEDY (cost=626, path=125,143,155,156,157,158,159,175,187,188,189,190,191,192,193,208,225,242,254,253,252,251,250,273,285,284,283,272,
LOCAL OPTIMUM (cost=586, path=125,143,155,156,157,158,159,175,187,188,189,190,191,192,193,208,225,226,227,228,229,230,231,210,197,198,1
```



▼ Labirint 8 (min cena=1566)

```
ALGORITHM: dfs
GREEDY (cost=1566, path=1545,1546,1547,1501,1482,1448,1427,1426,1425,1424,1423,1377,1359,1360,1361,1320,1303,1261,1240,1241,1242,1199,1
LOCAL OPTIMUM (cost=1566, path=1545,1546,1547,1501,1482,1448,1427,1426,1425,1424,1423,1377,1359,1360,1361,1320,1303,1261,1240,1241,1242

ALGORITHM: bfs
GREEDY (cost=1566, path=1545,1546,1547,1501,1482,1448,1427,1426,1425,1424,1423,1377,1359,1360,1361,1320,1303,1261,1240,1241,1242,1199,1
LOCAL OPTIMUM (cost=1566, path=1545,1546,1547,1501,1482,1448,1427,1426,1425,1424,1423,1377,1359,1360,1361,1320,1303,1261,1240,1241,1242
```



▼ Labirint 9 (min cena=203)

```
ALGORITHM: dfs
GREEDY (cost=1279, path=54,41,28,27,26,25,24,23,22,37,48,47,46,64,77,78,79,80,81,82,65,51,52,39,40,53,67,66,83,97,105,106,107,108,109,9
LOCAL OPTIMUM (cost=1167, path=54,41,28,27,26,25,24,23,22,37,48,47,46,64,77,78,79,80,81,82,65,51,52,39,40,53,67,66,83,97,105,106,107,10

ALGORITHM: bfs
GREEDY (cost=203, path=54,55,56,57,58,59,60,73,89,89,88,87,99,111,110,109,108,107,106,105,117,132,149,163,177,191,190,189,188,187,203,2
LOCAL OPTIMUM (cost=203, path=54,55,56,57,58,59,60,73,89,89,88,87,99,111,110,109,108,107,106,105,117,132,149,163,177,191,190,189,188,18
```