SEMINARSKA NALOGA

Umetna inteligenca

Tilen Volk 63140336, David Borštner 63170059

Ljubljana 26.11.2018

1. Uvod

Za seminarsko nalogo sva se odločila, da bova uporabila programski jezik Java. Za preiskovanje labirintov pa sva uporabila algoritme: DFS, BFS ter A*.

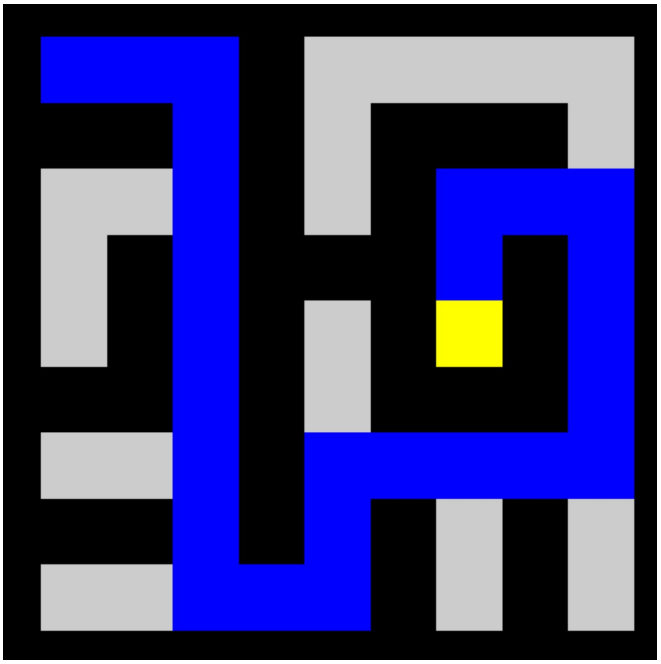
2. DFS algoritem

DFS(Depth-first search) se uporablja za iskanje čez drevesne strukture v globino. Začne se na vrhu drevesa, kjer se nato pomika do listov dokler ne najde pravilnega rezultata. Najden rezultat ni nujno idealen kar se pokaže tudi v iskanju skozi labirint kjer pride do izhoda ni pa nujno, da pride po najhitrejši oz. najučinkovitejši poti.

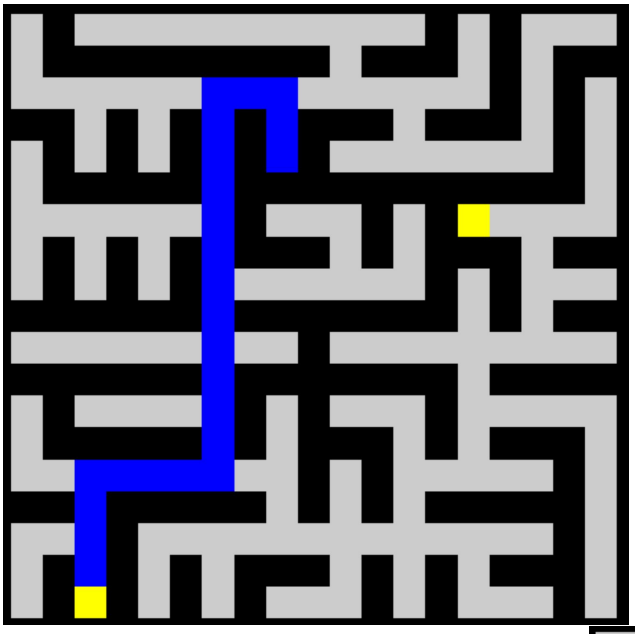
Najin algoritem deluje z uporabo labirinta, ki ga preberema iz datoteke ter pretvorima v 2D array. Za posamezne koordinate uporablja class node. Za samo iskanje pa glavni razred, ki vsebuje glavni metodi za preiskovanje labirinta: isci ter SosednjeKoordinate. Metoda SosednjeKoordinate vzame trenutno pozicijo oz. trenuten node ter vrne vse sosednje koordinate, če se te ne nahajajo na mestu z vrednostjo -1. Metoda isci pa z shranjuje node skozi katere hodimo dokler ne najdemo koordinate z vrednostjo -3.

Za izris se uporablja R.

Ime labirinta	Cena	Globina
Labirint_1	100	26
Labirint_2	92	24
Labirint_3	252	64
Labirint_4	412	104
Labirint_5	57	26
Labirint_6	49	24
Labirint_7	165	64
Labirint_8	260	104
Labirint_9	57	26
Labirint_10	55	24
Labirint_11	157	64
Labirint_12	259	104

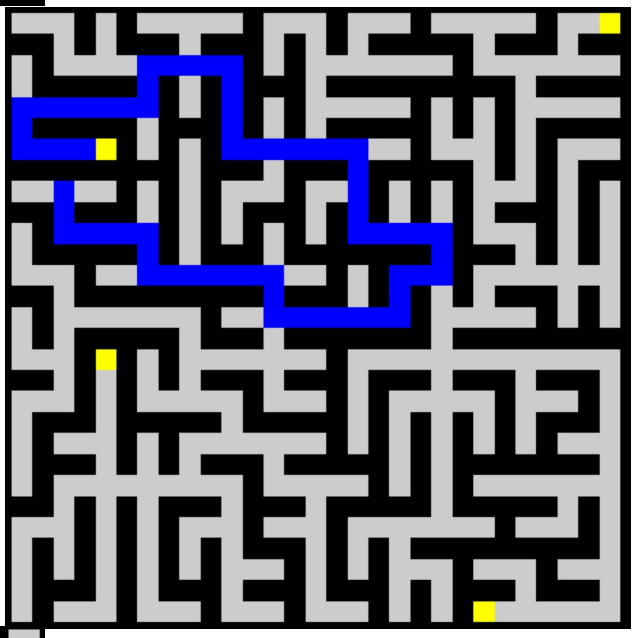


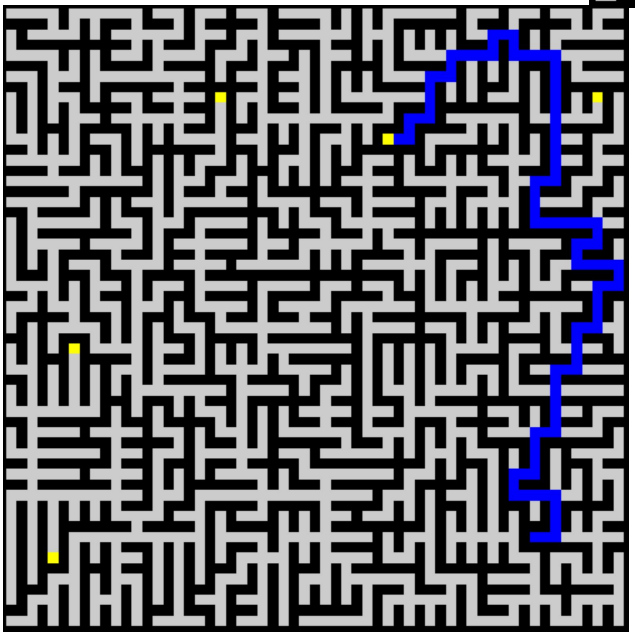
Izris poti za labirinte: 1, 5 in 9



Izris poti za labirinte: 2, 6 in 10

Izris poti za labirinte: 3, 7 in 11





Izris poti za labirinte: 4, 8 in 12

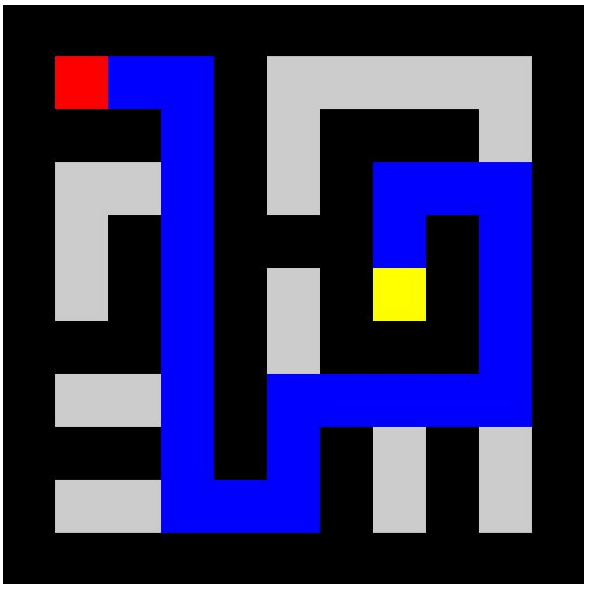
3. BFS algoritem(Dijkstra)

BFS(breadth-first-search) je algoritem z katerim iščemo najmanjše število korakov od začetne točke do končne točke v določeni povezani strukturi. Ker v naši nalogi so bili koraki ocenjeni z različnimi vrednostmi, prav takega algoritma ne moreš uporabiti, zato smo tega spremenili, da upošteva cene povezav. Na tak način, postane naš algoritem enak dijkstri. Dijkstra na podatkih, kjer so vse cene enake, je enaka algoritmu BFS. Algoritem dela tako, da imaš dve listi. Ena je lista obiskanih vozlišč, druga pa lista naslednjih kandidatov za obiskat. Na začetku je lista kandidatov prazna in v listi obiskanih pa je vozlišče začetka. Pogleda sosede začetka in jih sortirano vstavi na listo kandidatov, tako da je kandidat z najmanjšo ceno zmeraj prvi. Na koncu prvega kandidata izbriše iz liste naslednjih in ga vstavi v listo obiskanih. Tako dela dokler ne vstavi v listo obiskanih vozlišče označeno z -3.

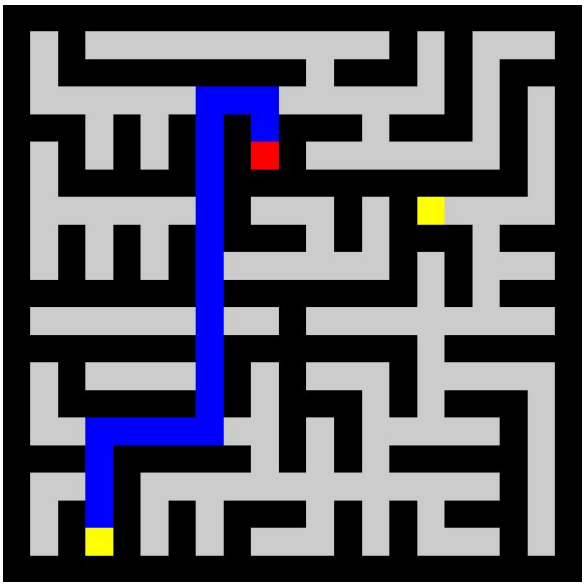
Rezultati algoritma:

Ime labirinta	Cena	Število korakov	Število preverjanj	Število dodanih vozlišč v listo obiskanih	Število dodanih vozlišč v listo kandidatov
labyrinth_1	100	27	172	44	44
labyrinth_2	92	25	508	128	134
labyrinth_3	164	43	1136	285	302
labyrinth_4	300	77	2824	707	730
labyrinth_5	57	27	176	45	45
labyrinth_6	49	25	508	128	133
labyrinth_7	124	43	1436	360	371
labyrinth_8	231	77	3588	898	933
labyrinth_9	57	27	168	43	43
labyrinth_10	55	25	500	126	131
labyrinth_11	99	43	1024	257	272
labyrinth_12	189	77	2884	722	740

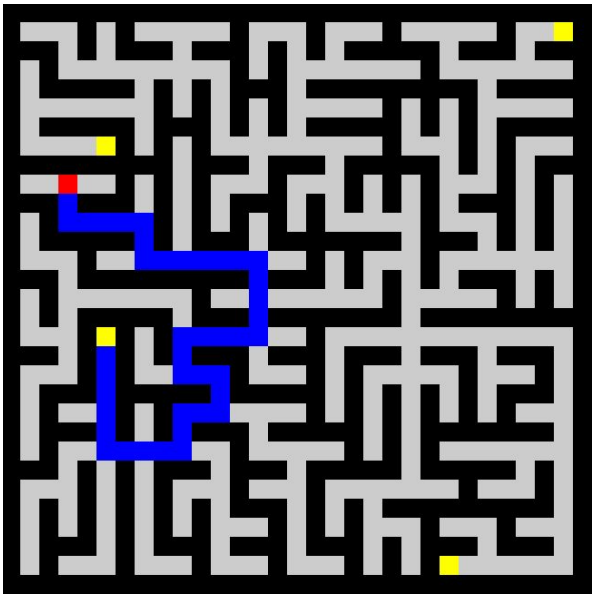
Slike najdenih rešitev:



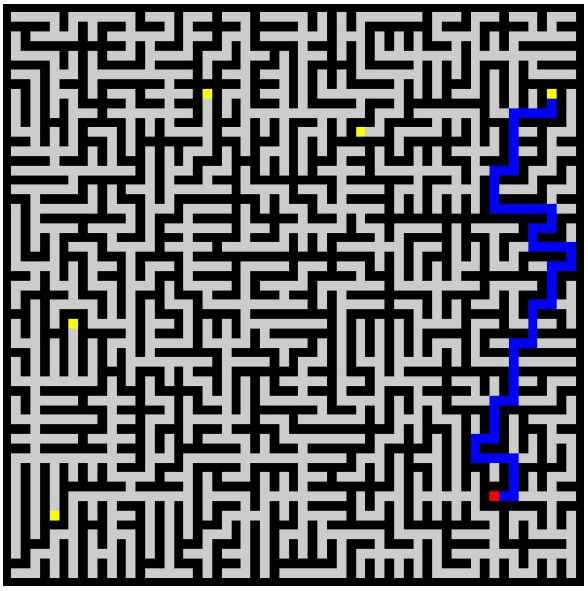
Izris poti za labirinte: 1, 5 in 9



Izris poti za labirinte: 2, 6 in 10



Izris poti za labirinte: 3, 7 in 11



Izris poti za labirinte: 4, 8 in 12

4. A* algoritem

A* je algoritem za iskanje najkrajše poti. Je zelo podoben algoritmu dijkstra, samo da še uporablja hevristiko. Predvideva v kakšno smer more it, glede na najbližji cilj. Ni nujno da je ta najbližje, če je med začetno točko in končno ovira. To uporablja tako, da dijkstra algoritmu dodamo še en parameter glede na katerega se odloča in ta parameter je razdalja od trenutnega vozlišča do končnega.

Rezultati algoritma:

Ime labirinta	Cena	Število korakov	Število preverjanj	Število dodanih vozlišč v listo obiskanih	Število dodanih vozlišč v listo kandidatov
labyrinth_1	100	27	168	43	43
labyrinth_2	92	25	476	120	124
labyrinth_3	164	43	952	239	255
labyrinth_4	300	77	2412	604	627
labyrinth_5	57	27	168	43	43
labyrinth_6	49	25	436	110	117
labyrinth_7	124	43	1116	280	299
labyrinth_8	231	77	3004	752	778
labyrinth_9	57	27	160	41	41
labyrinth_10	55	25	452	114	117
labyrinth_11	99	43	756	190	206
labyrinth_12	189	77	2296	575	596

Slike rešitev so enake kot pri dijkstri.