Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Seminarska naloga iz preiskovalnih algoritmov
Umetna Inteligenca

Nejc Vrčon Zupan (63200327)

Luka Šveigl (63200301)

1. Uvod

Za seminarsko nalogo iz preiskovalnih algoritmov je bilo potrebno poiskati pot, ki najde vse zaklade v podanem labirintu. Za iskanje poti je bilo potrebno uporabiti preiskovalne algoritme, ter rezultate primerjati oz. analizirati. Odločila sva se, da bova nalogo naredila v progamskem jeziku Python.

2. Labirinti

Podanih je bilo 9 labirintov različnih težavnosti in velikosti. Tloris labirinta je bil podan v obliki 2D matrike z celoštevilskimi vrednostmi v tekstovni datoteki. Vsaka celoštevilska vrednost je predstavljala svoj pomen. Vsak labirint vsebuje začetno in ciljno polje, ter eno polje z zakladom.

Vrednost	**Pomen**
-1	Zid
>= 0	Hodnik
-2	Začetno polje
-3	Zaklad
-4	Ciljno polje

3. Preiskovalni algoritmi

Naloga preiskovalnih algoritmov je bila poiskati čim cenejšo pot od začetnega polja do enega izmed ciljnih polj tako, da na poti pobere vse zaklade v labirintu. Prehodi skozi zid seveda niso bili dovoljeni. Implementirala sva 5 algoritmov:

- BFS
- DFS
- IDDFS
- A*
- IDA*

3.1 BFS (Breadth-first search)

3.1.1 Delovanje

Algoritem išče primerno vozlišče po drevesni strukturi in sicer začne pri korenu, ter išče po globinskih nivojih v širino. Dokler ne preišče trenutnega globinskega nivoja se ne spušča globje. Potrebuje dodaten pomnilnik, da si pomni vozlišča, ki jih sreča ampak jih še ni raziskal. Algoritem najprej najde najkrajšo pot.

3.1.2 Prostorska in časovna kompleksnost

Časovno zahtevnost lahko opišemo, kot $O(b^d)$, prostorsko zahtevnost pa kot $O(b^d)$.
b - faktor vejanja grafa, d – globina najbljižjega končnega stanja

3.1.3 Uporaba

BFS se uporablja v mnogo algoritmih (Cheney, Aho–Corasick, Ford–Fulkerson, Cuthill–McKee), lahko pa tudi ugotavljamo dvodelnost grafa, serializacijo binarnega drevesa in pa seveda iščemo najkrajšo pot med dvemi vozlišči v grafu, kjer je merjena pot število obiskanih vozlišč.

3.2 DFS (Depth-first search)

3.2.1 Delovanje

Algoritem išče primerno vozlišče po drevesni strukturi in sicer začne pri korenu, ter razišče pot po vsaki veji do konca, preden se vrača nazaj. Algoritem se lahko hitro ujame v zanko. Ni nujno, da algoritem najprej najde najkrajšo pot.

3.2.2 Prostorska in časovna kompleksnost

Časovno zahtevnost lahko opišemo, kot $O(b^m)$, prostorsko zahtevnost pa kot $O(bm)$.
b - faktor vejanja grafa, m – maksimalna globina prostora stanj

3.2.3 Uporaba

DFS se uporablja pri topološkem sortiranju, iskanju povezanih komponent v grafih, iskanje mostov v grafih, reševanje in generiranje labirintov.

3.3 IDDFS (Iterative deepening depth-first search)

3.3.1 Delovanje

Algoritem deluje identično, kot DFS z razliko, da se iterativno poglablja. Torej ne razišče pot po vsaki veji takoj do konca ampak se spusti do neke globine, glede na stanje oz. stopnjo iteracije. Algoritem tako zavzame bistveno manj prostora.

3.3.2 Prostorska in časovna zahtevnost

Prostorska zahtevnost IDDFS-ja je precej manjša od DFS-ja in sicer $O(bd)$. Časovna zahtevnost pa je ista, kot pri BFS-ju in sicer $O(b^d)$.

b - faktor vejanja grafa, d – globina najbljižjega končnega stanja

3.3.3 Uporaba

IDDFS se uporablja pri avtomatiziranem razporejanju in načrtovanju.

3.4 A*

3.4.1 Delovanje

A* izbira premik glede na rezultat f, ki je seštevek g funkcije in h funkcije, v najini implementaciji je bila ta manhattanska razdalja od vozlišča do končnega vozilšča oz. zaklada. Ko je algoritem prišel do enega izmed zakladov je ponovno izračunal h funkcijo. Slabost algoritma je, da zasede ogromno prostora in ima s tem veliko prostorsko kompleksnost.

3.4.2 Prostorska in časovna zahtevnost

Prostorsko in časovna zahtevnost algoritma, je težko določiti saj je odvisna od h funkcije, ki si jo programer izbere sam.

3.4.3 Uporaba

A* se uporablja, pri iskanju poti v aplikacijah (npr. video igre) in procesiranju naravnega jezika.

3.5 IDA* (Iterative deepening A*)

3.5.1 Delovanje

IDA* deluje z pomočjo f,. g in h funckije z razliko od A*, pa ta izločuje vozlišča, ki ne dosežejo nastavljene meje funkcije. Meja se pri vsakem obhodu poveča na min. vrednost sosednjih vozlišč, ki jih še nismo obiskali vendar so povezana z temi, ki smo jih.

3.5.2 Prostorska in časovna zahtevnost

Kot pri A* je prostorsko in časovno zahtevnost algoritma, težko določiti saj je odvisna od h funkcije.

3.5.3 Uporaba

IDA* se uporablja pri avtomatiziranem razporejanju in načrtovanju.

4. Rezultati

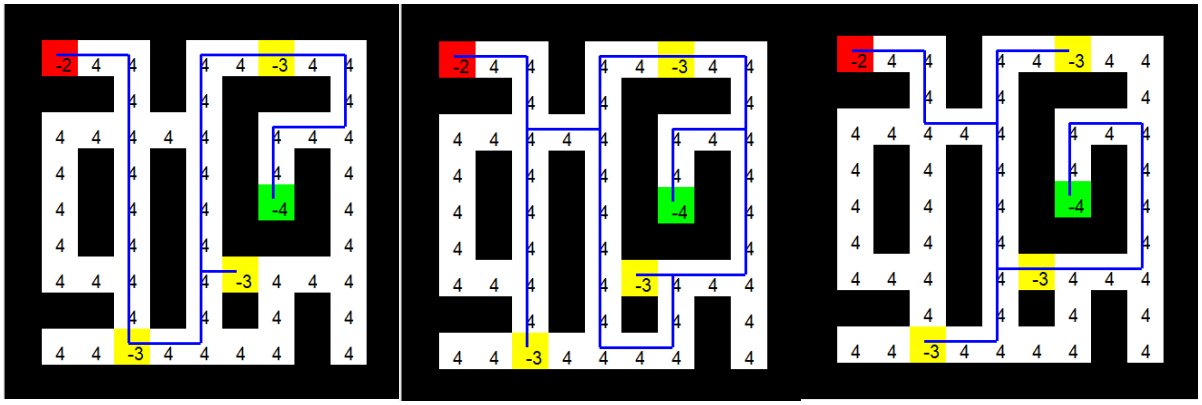
4.1 Labirint 1

Algoritem	Cena	Dolžina poti	Št. Preverjenih vozlišč
BFS	112	33	114
DFS	268	73	132
IDDFS	112	33	925
A*	140	41	215
IDA*	140	41	1681

Najboljša algoritma sta BFS in IDDFS, ki sta našla isto pot, ker imajo v tem labirintu vsa vozlišča isto ceno. Druga najboljša algoritma sta A* in IDA*, ki sta našla isto pot, a slabšo kot BFS in IDDFS. Najslabši algoritem je DFS.

Po številu preverjenih vozlišč je najboljši algoritem BFS, nato DFS in nato A*. Najslabša sta seveda IDDFS in IDA*, ki sta iterativna algoritma. Kar pridobita s tem, da preiščeta več vozlišč je boljša prostorska kompleksnost.

Maksimalna globina iskanja IDDFS: 9



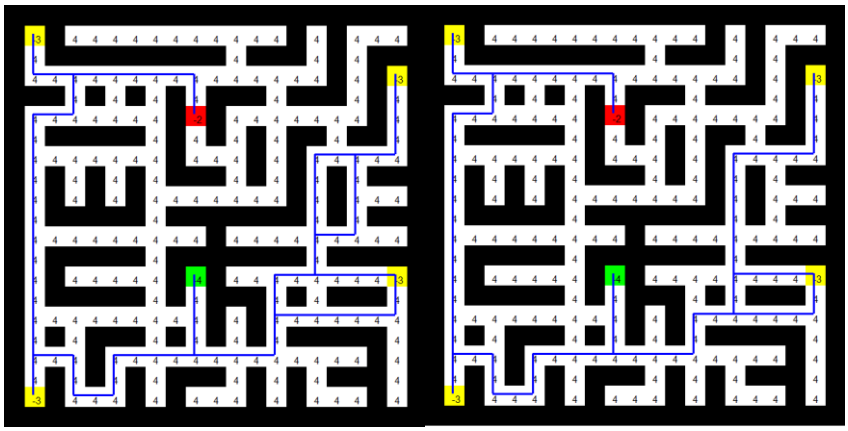
Levo: BFS in IDDFS, Sredina: DFS, Desno: A in IDA**

4.2 Labirint 2

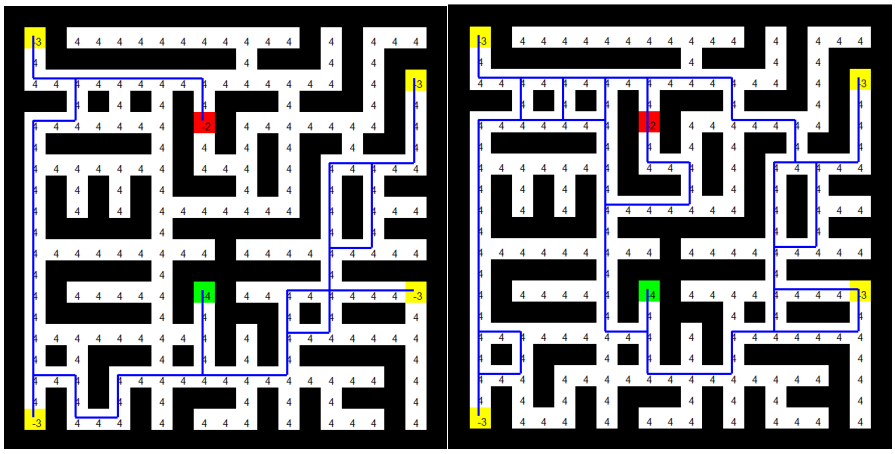
Algoritem	Cena	Dolžina poti	Št. Preverjenih vozlišč
BFS	412	109	628
DFS	952	247	460
IDDFS	412	109	31083
A*	412	109	2075
IDA*	412	109	69517

Algoritmi BFS, IDDFS, A* in IDA* so našli isto pot z isto ceno, najslabši je bil zopet algoritem DFS, a je preiskal najmanj vozlišč. Zopet sta po pričakovanjih iterativna algoritma preiskala največ vozlišč, najslabši po tej metriki je bil algoritem IDA*.

Maksimalna globina iskanja IDDFS: 32



*Levo: BFS, Desno: IDDFS in A**



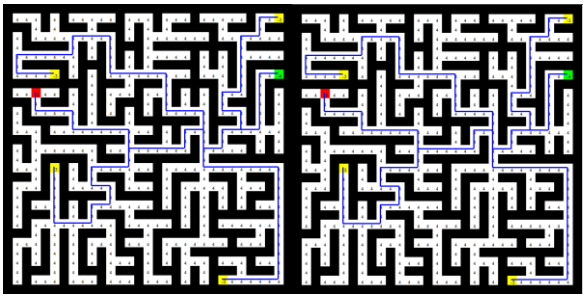
Levo: IDA, Desno: DFS*

4.3 Labirint 3

Algoritem	Cena	Dolžina poti	Št. Preverjenih vozlišč
BFS	1116	285	1732
DFS	1196	305	1254
IDDFS	1116	285	74793
A*	1116	285	3094
IDA*	1116	285	44061

Ponovno so algoritmi BFS, IDDFS, A* in IDA* našli isto pot, DFS je bil zopet najslabši. Po številu preverjenih vozlišč je najboljši DFS, najslabša pa sta zopet A* in IDA*.

Maksimalna globina iskanja IDDFS: 66



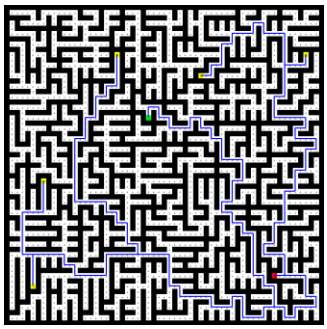
Levo: BFS, IDDFS, A, IDA*, Desno: DFS (na pogled zgledata poti enaki, a nista)*

4.4 Labirint 4

Algoritem	Cena	Dolžina poti	Št. Preverjenih vozlišč
BFS	2496	631	4820
DFS	2496	631	5230
IDDFS	2496	631	641333
A*	2496	631	8571
IDA*	2496	631	805965

Pri tem labirintu opazimo, da vsi algoritmi najdejo isto pot z isto ceno, edina razlika med njimi je v številu preverjenih vozlišč. Seveda so tudi tu veliko boljši neiterativni algoritmi, razlika med iterativnimi in neiterativnimi pa je ogromna.

Maksimalna globina iskanja IDDFS: 213



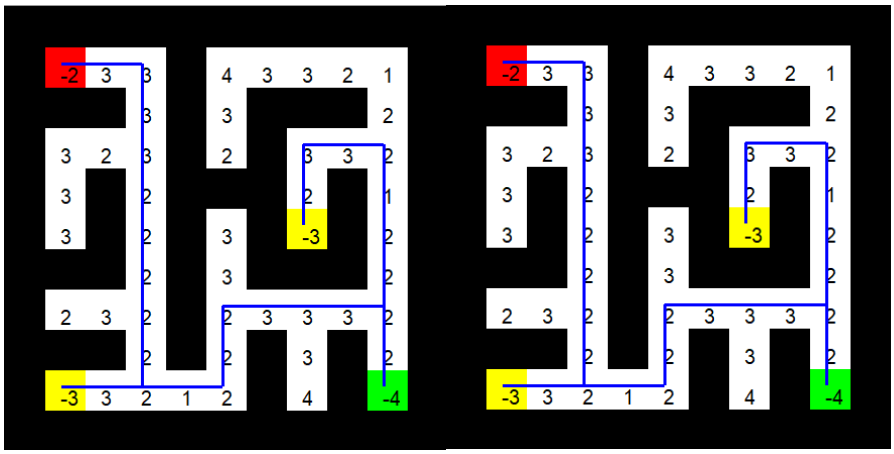
Vsi algoritmi

4.5 Labirint 5

Algoritem	Cena	Dolžina poti	Št. Preverjenih vozlišč
BFS	84	41	82
DFS	120	57	131
IDDFS	84	41	1220
A*	84	41	89
IDA*	84	41	632

Pri tem labirintu imajo zopet vsi algoritmi, razen DFS isto pot, zopet je najslabši DFS. Glavno razliko opazimo v številu preverjenih vozlišč, kjer je najboljši DFS, najslabša pa sta iterativna algoritma, s tem da je IDA* pregledal pol manj polj kot IDDFS.

Maksimalna globina iskanja IDDFS: 11



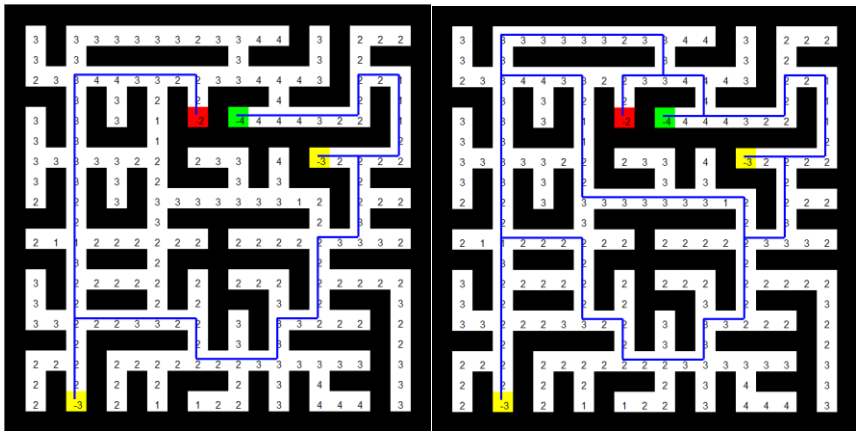
Levo: BFS, IDDFS, A, IDA*, Desno: DFS (na pogled zgledata poti enaki, a DFS najprej najde desni zaklad)*

4.6 Labirint 6

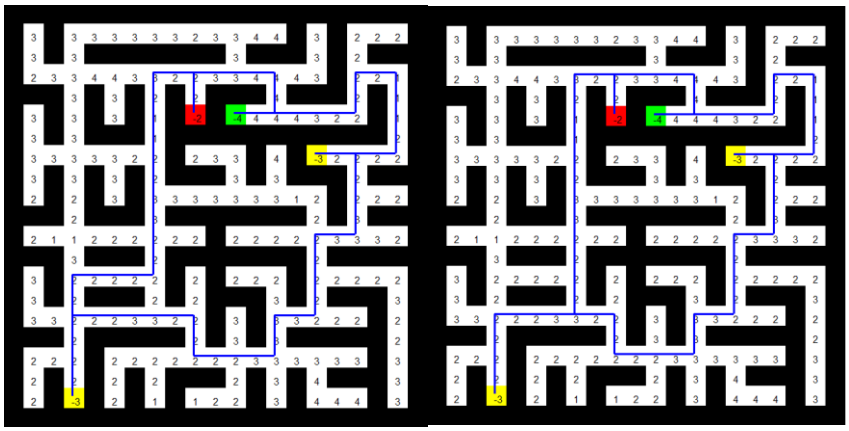
Algoritem	Cena	Dolžina poti	Št. Preverjenih vozlišč
BFS	168	75	435
DFS	407	163	426
IDDFS	199	87	28025
A*	199	87	796
IDA*	201	87	11297

Pri tem labirintu je bil najboljši algoritem BFS, saj je našel najkrajšo in najcenejšo pot. Druga najboljša algoritma sta bila IDDFS in A*, ki se razlikujeta le po številu preverjenih vozlišč. Naslednji algoritem je IDA*, ki je našel pot iste dolžine kot IDDFS in IDA*, le da je cena te poti rahlo slabša. Najslabši algoritem je zopet DFS.

Maksimalna globina iskanja IDDFS: 40



Levo: BFS, Desno: DFS



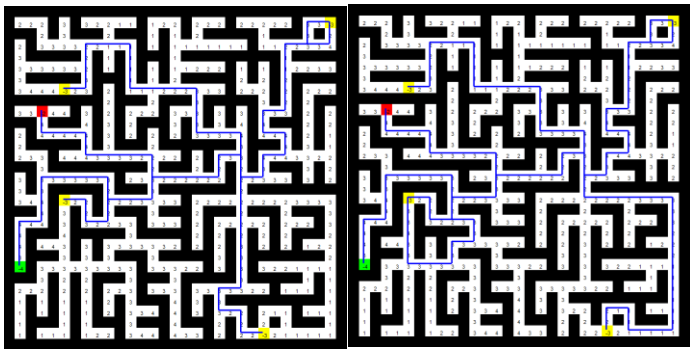
Levo: IDDFS in A, Desno: IDA**

4.7 Labirint 7

Algoritem	Cena	Dolžina poti	Št. Preverjenih vozlišč
BFS	612	245	1598
DFS	720	305	1064
IDDFS	612	245	110815
A*	612	245	2983
IDA*	612	245	70368

Pri tem labirintu so vsi algoritmi, razen DFS, našli isto pot z isto ceno, zopet je glavna razlika v številu preiskanih vozlišč. Po pričakovanjih sta po tej metriki najslabša iterativna algoritma.

Maksimalna globina iskanja IDDFS: 73



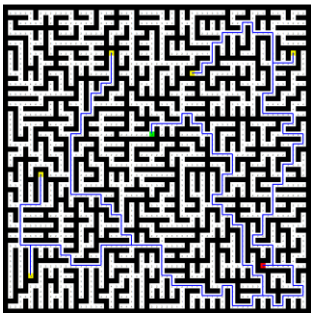
Levo: BFS, IDDFS, A*, IDA*, Desno: DFS

4.8 Labirint 8

Algoritem	Cena	Dolžina poti	Št. Preverjenih vozlišč
BFS	1560	625	4787
DFS	1560	625	5240
IDDFS	1560	625	618174
A*	1560	625	6659
IDA*	1560	625	510557

Pri tem labirintu so vsi algoritmi našli isto pot z isto ceno, razlikovali so se le po številu preiskanih vozlišč. Zopet sta bila po tej metriki najslabša iterativna algoritma.

Maksimalna globina iskanja IDDFS: 204



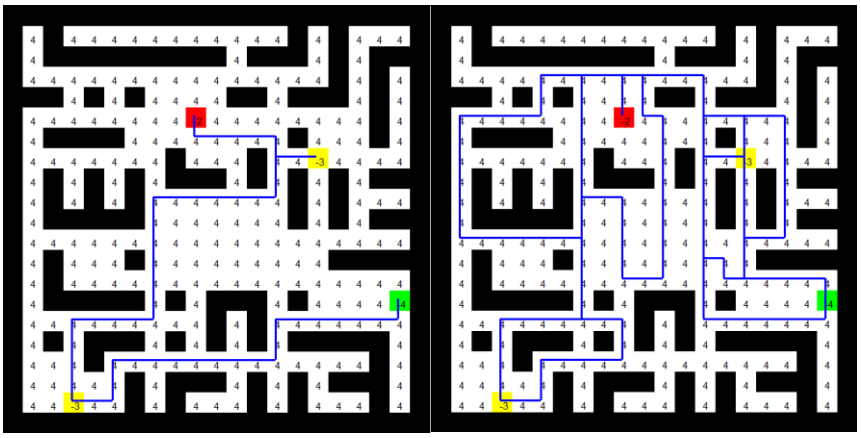
Vsi algoritmi

4.9 Labirint 9

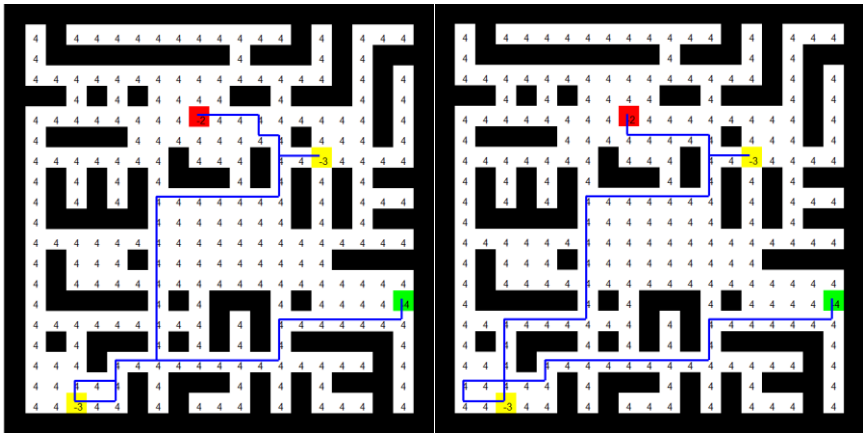
Algoritem	Cena	Dolžina poti	Št. Preverjenih vozlišč
BFS	200	54	495
DFS	720	186	325
IDDFS	200	54	841119
A*	200	54	34636
IDA*	216	58	696723970

Pri tem labirintu so algoritmi BFS, IDDFS in A* našli isto pot, za njimi je bil algoritem IDA*, zadnji pa je bil algoritem DFS. Po številu preverjenih vozlišč je najboljši algoritem DFS, daleč najslabši pa je algoritem IDA*.

Maksimalna globina iskanja IDDFS: 26



Levo: BFS in A, Desno: DFS*



*Levo: IDDFS, Desno: IDA**

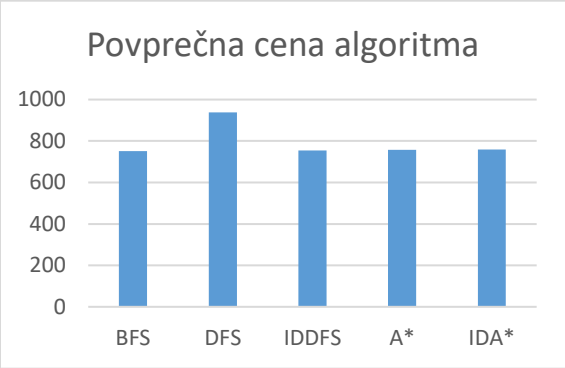
4.10 Ugotovitve

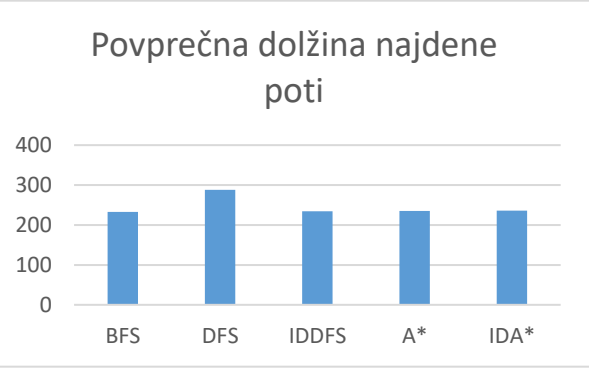
Po analizi rezultatov sva ugotovila, da je v večini primerov najslabšo pot našel algoritem DFS. V dveh primerih (labirint 4 in labirint 8), so vsi algoritmi našli enako dobro pot. V večini primerov so algoritmi BFS, IDDFS, A* in IDA* našli enako dobro pot, v enem primeru (labirint 6) pa je BFS našel boljšo pot od vseh ostalih algoritmov.

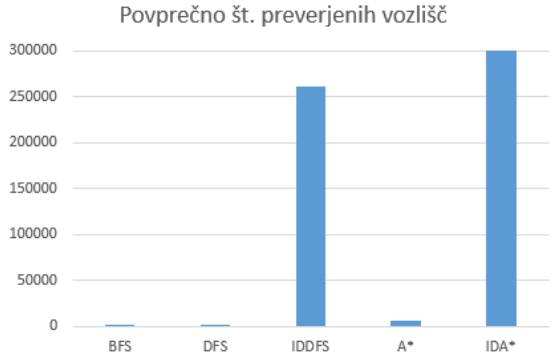
Pri analiziranju števila preverjenih vozlišč sva potrdila pričakovano, in sicer, da iterativni algoritmi preiščejo bistveno več vozlišč kakor neiterativni, zaradi tega so tudi počasnejši, a s tem pridobijo na prostorski zahtevnosti, saj ne shranjujejo že obiskanih vozlišč.

Predvidevava, da so rezultati DFS tako slabi ker sva izvajanje algoritma zaključila takoj, ko je našel validno pot. Ker pa je značilnost algoritma DFS, da ne garantira da je prva pot ki jo najde, najbol optimalna, torej bi lahko rezultat algoritma izboljšala tako, da bi zbrala vse poti ki jih generira in vrnila najbolj optimalno.

Prav tako lahko iz značilnosti BFS algoritma, ki pravi da vedno najprej najde najkrajšo pot, sklepamo, da je zato v vseh labirintih dosegal najboljše rezultate.







Algoritem	Cena	Dolžina	Št. Preverjenih vozlišč
BFS	751,1111	233,1111	1632,333333
DFS	937,6667	288	1584,666667
IDDFS	754,5556	234,4444	260831,8889
A*	757,6667	235,3333	6568,666667
IDA*	759,6667	235,7778	7915338,667

5. Zaključek

Pri izdelavi seminarske naloge sva se izredno zabavala, sploh zato ker sva uporabila Python. Ugotovila sva, da ne obstaja en algoritem, ki bi zagotovil vsem našim potrebam, zato moramo pri izbiri primernega algoritma upoštevati več faktorjev, kot so npr. velikost problema, strojna oprema, željena hitrost pridobitve rešitve, natančnost itd.