# ĪSĀKĀ CEĻA MEKLĒŠANAS ALGORITMI
## *PATHFINDING ALGORITHMS*

Author: **Einārs BISTROVS**, e-mail: einars1997@gmail.com
Scientific supervisor: **Sergejs KODORS, Dr.sc.ing.,** e-mail: sergejs.kodors@rta.lv
Rēzeknes Tehnoloģiju akadēmija
Atbrīvošanas aleja 115, Rēzekne, Latvija

***Anotācija:*** *Darbā ir aprakstīts salīdzinājums starp trim īsākā ceļa meklēšanas algoritmiem: A*, Deikstras algoritma, Plašas pirmās meklēšanas algoritma (Breadth first search). Algoritmi tika salīdzināti aplikācijā, kura ģenerē labirintus pēc gadījuma principa. Algoritmu izpildes laikā tiek iegūti trīs veidu dati – izpildes laiks, veikto operāciju skaits, ceļa garums.*

***Atslēgvārdi:*** *algoritms, A*, Deikstras algoritms, operāciju skaits, īsākais ceļš.*

### Ievads

Loģistikas uzņēmumi tērē daudz laika, lai izplānotu maršrutus, kuri būtu ekonomiski izdevīgāki un aizņemtu pēc iespējas mazāk laika. Šo darbu ir iespējams veikt ar roku (rēķinot iespējamo vidējo pārvietošanās ātrumu katram ceļa posmam un veidot diagrammas), taču tas ir ļoti laikietilpīgs process, kurā pastāv samērā liela iespēja uz cilvēku pieļautajām kļūdām (piem., nepamanīti ceļi, neprecīzi izrēķināts ātrums, utt.).

Ļoti līdzīgā situācijā nonāk tūristi, kuri vēlas nokļūt galamērķī pēc iespējas ātrāk, pietam, ir jāņem vērā, ka ne visiem tūristiem ir spēja saplānot īsāku maršrutu var secināt, ka maršruta plānošana tūristiem aizņem daudz laika.

Darba mērķis: salīdzināt īsākā ceļa meklēšanas algoritmus.

Materiāli un metodes

Lai salīdzinātu ceļa meklēšanas algoritmus bija realizēta programma, balstoties uz pathfinding.js projekta [1], pielietojot šādas tehnoloģijas HTML, CSS, Javascript, jQuery, javascript state machine, kas salīdzināja algoritmu pēc trim parametriem: izpildes laika, ceļa garuma, veikto operāciju skaita; ģenerējot labirintu pēc gadījuma principa.

Rezultāti

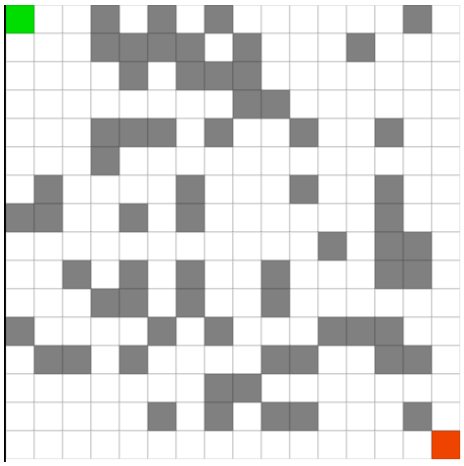
Pathfinding.js projektā īsākā ceļa meklēšanas algoritmi ir rakstīti JavaScript valodā, izmantojot prototipēšanu. Kā pamatalgoritms tiek lietos A* algoritms. [3] Deikstras algoritms [2] koda ziņā izpildās tā pat kā A* algoritms, tikai Deikstras algoritms neizmanto heiristisko metodi. Algoritmu izpildē tiek izmantotas kaudzes (heap), kas ir vērtību kolekciju, kur lielākā vērtībā vienmēr ir augšgalā. Kā heiristiskā metode tiek izmantota Manhetenas metode (Manhattan), kas izpaužas kā vertikālās un horizontālās komponentes summa.[1]

Pathfinding.js projektā ir izveidota funkcionalitāte, kura izdara 3 viedu mērījumus algoritma izpildes laikā, šie mērījumi ir

izpildes laiks;
veikto operāciju skaits;
ceļa garums.

Tika veikta algoritmu salīdzināšana pēc gadījuma principa ģenerētā labirintā, (skatīt 1. attēlu).

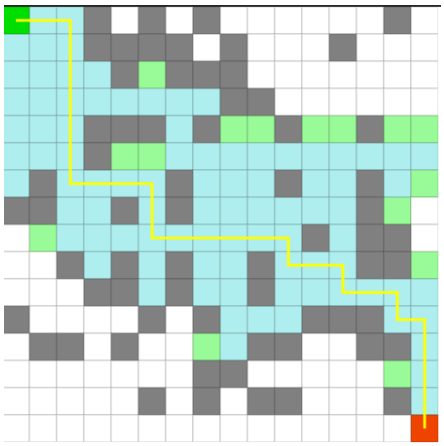
*https://doi.org/10.17770/het2020.24.6743*



*1. attēls. Ģenerētais labirints*

*A** algoritma izpildes rezultāts (skatīt 2. Attēlu) bija

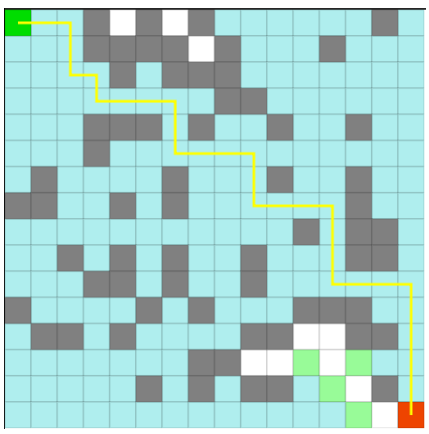
- Izpildes laiks 0,1 milisekundes;
- Veikto operāciju skaits – 197;
- Ceļa garums 30.



*2. attēls. A* algoritma izpilde ģenerētā labirintā*

Deikstras algoritma izpildes rezultāts(skatīt 3. Attēlu) bija

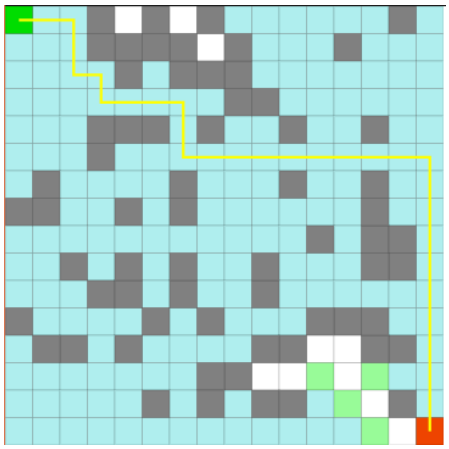
- Izpildes laiks 0,2 milisekundes;
- Operāciju skaits – 358;
- Ceļa garums 30.



*3. attēls. Deikstras algoritma izpilde ģenerētā labirintā*

Plašas pirmās meklēšanas (Breadth first search) algoritma [4] izpildes rezultāts (skatīt 4. attēlu) bija

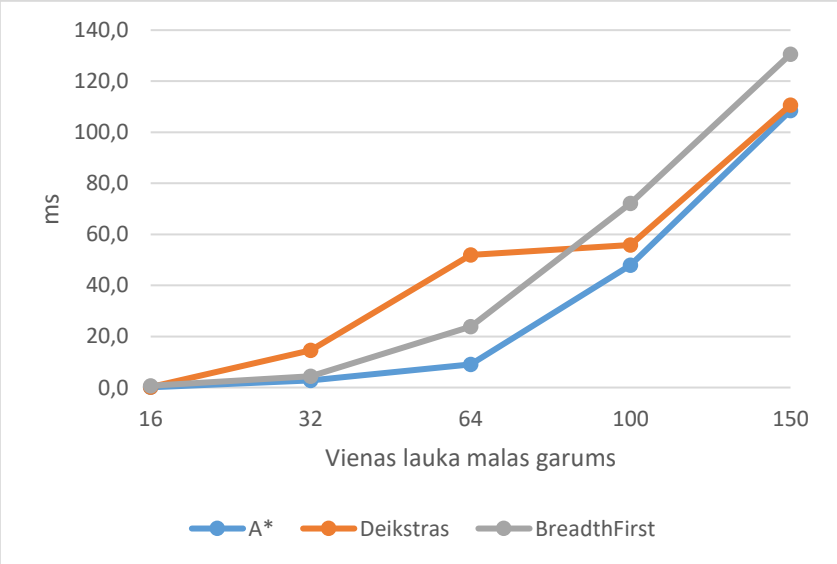
- Izpildes laiks 0,7 mili sekundes;
- Veikto operāciju skaits 358;
- Ceļa garums 30.



*4. attēls. Plašas pirmās meklēšanas algoritma izpilde ģenerētā labirintā*

Mērījumi tika atkārtoti 5 reizes, katru reizi palielinot labirinta izmērus. Tas tika veikts, lai novērotu lauka lieluma ietekmi uz algoritmu izpildes laikiem, kā arī lai pārliecinātos, ka pie citiem lauku izmēriem algoritmi pēc izpildes laika sarindojas tādā pašā secībā. Mērījumi tika veikti kvadrātiskā laukā, lai atvieglotu datu salīdzināšanu par lauka izmēra mērvienību tiek ņemts vienas malas garums.

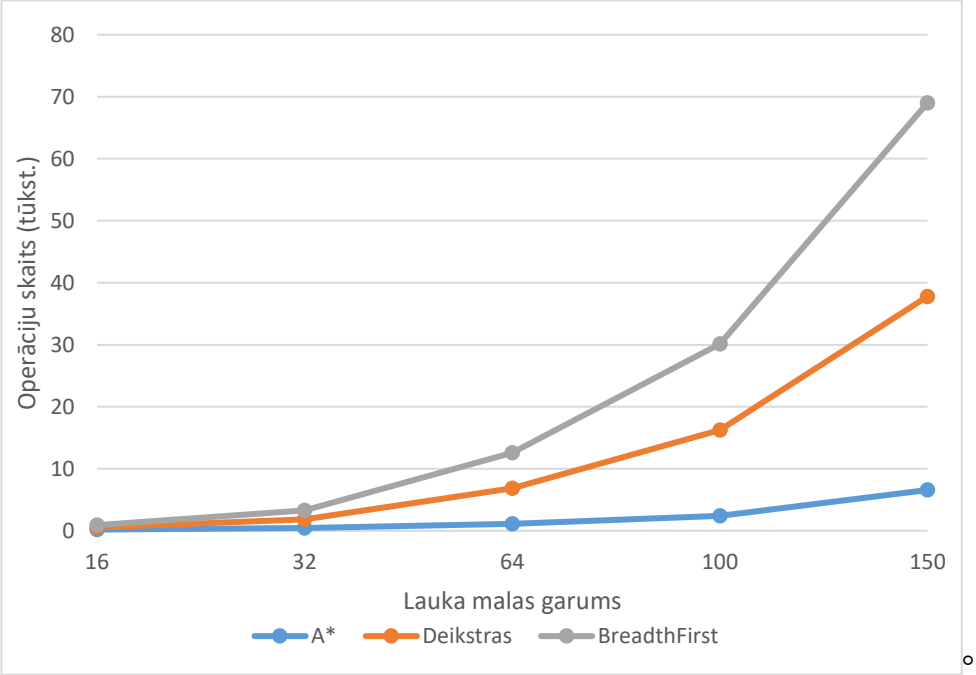
Salīdzinot algoritmu izpildes laikus visos izmēros var novērot, ka ātrākais algoritms pie visiem lauka izmēriem ir *A** algoritms. Līdz noteiktai robežai, kura ir aptuveni 80 vienības otrs ātrākais bija Breadth First Search algoritms, taču sasniedzot robežu  deikstras algoritms, kļūst ātrāks nekā Breadth First Search algoritms. Kā arī Deikstras algoritms ļoti pietuvojas a* algoritma ātrdarbībai, (skatīt 5. Attēlu).




*5. attēls. Algoritmu salīdzinājums pēc laika (ms)*

Salīdzinot algoritmus pēc izpildīto operāciju skaita, (skatīt 6. Attēlu), tika noskaidrots, ka neatkarīgi no lauka lieluma algoritmi sarindojas tādā pašā secībā. *A** algoritmam, salīdzinot pēc veikto operāciju skaita, ir nospiedošs pārsvars ar vidēji teju 4 reizes mazāk operāciju nekā

Deikstras algoritmam, savukārt *breadth first search* algoritms veic vēl daudz vairāk operāciju. Gadījumos, kur veikto operāciju skaits ir būtisks A* algoritms ir daudz efektīvāks nekā Deikstras un *breadth first seach* algoritms.




*6. attēls. Algoritmu salīdzinājums pēc veikto operāciju skaita*

## Secinājumi

- Algoritmi pēc darbības principa atšķiras ar to, ka Plašas pirmās meklēšanas algoritmā, mezgli tiek apskatīti pēc principa – kā pirmo apskatīt mezglu, kurš ir tuvāk sākuma punktam. Deikstras algoritmā – kā pirmo apskatīt mezglu, kuram ir mazākais svars. *A** algoritmā – kā pirmo apskatīt mezglu, kur heiristiskās metodes un svaru summa ir mazākā.
- Salīdzināšana tika veikta kā bāzi, izmantojot *pathfinding.js* projektu, to modificējot pievienojot tam labirinta ģenerēšanas funkcionalitāti. Labirinti tika ģenerēti pēc gadījuma principa.
- Salīdzinot algoritmus pēc īsākā ceļa garuma tika noskaidrots, ka visi trīs algoritmi pie pieciem dažādiem lauku izmēriem atrod vienādi garus ceļus, kuri ir aptuveni vienādi ar kvadrātiskā lauka malu reizinājumu.
- Savukārt, salīdzinot algoritmus pēc veikto operāciju skaita tika noskaidrots, ka šajā ziņā nospiedošs pārsvars ir *A** algoritmam, kas veic aptuveni 4 reizes mazāk operāciju nekā Deikstras algoritms un aptuveni 7 reizes mazāk kā Plašas pirmās meklēšanas algoritms.
- Salīdzinot algoritmu izpildes laikus tika konstatēts, ka vispārākais šajā ziņā ir A* algoritms, taču arī tika noskaidrots, ka Deikstras algoritms būtiski atpaliek no *A** algoritma tikai laukos, kuriem ir salīdzinoši mazs izmērs, jo lielāks laukus, jo mazāka šī atšķirība. Pie laukuma izmēra 150 vienības, tika konstatēts, ka laika atšķirība ir mazāka par 1 sekundi, kas norāda, to, ka Deikstras algoritms, kuru ir daudz vieglāk realizēt nekā *A** algoritmu pie lieliem lauku izmēriem var būt efektīvāks risinājums nekā *A** algoritms.

### Bibliography


1. https://github.com/qiao/PathFinding.js/ skatīts internetā 07.05.2018
2. Sergejs Kodors. “Deikstras algoritms”, ekursi.rta.lv skatīts internetā 03.05.2018
3. https://www.slideshare.net/dnatapov/a-path-finding skatīts internetā 03.05.2018
4. https://www.slideshare.net/jyothimonc/b-8742532 skatīts internetā 03.05.2018