Složenost Dijkstra algoritma te primjena kod uređaja za podzemnu navigaciju u podzemnim prostorima

Ćesić, Bruno

SVEUČILIŠTE U ZAGREBU

RUDARSKO-GEOLOŠKO-NAFTNI FAKULTET

Diplomski studij rudarstva

SLOŽENOST DIJKSTRA ALGORITMA TE PRIMJENA KOD UREĐAJA ZA PODZEMNU NAVIGACIJU U PODZEMNIM PROSTORIMA

Diplomski rad

Bruno Ćesić

R-257

Zagreb, 2020.

KLASA: 602-04/20-01/238
URBROJ: 251-70-03-20-2
U Zagrebu, 23.11.2020.

Bruno Ćesić, student

RJEŠENJE O ODOBRENJU TEME

Na temelju Vašeg zahtjeva primljenog pod KLASOM 602-04/20-01/238, UR. BROJ: 251-70-11-20-1 od 02.11.2019. godine priopćujemo temu diplomskog rada koja glasi:

SLOŽENOST DIJKSTRA ALGORITMA TE PRIMJENA KOD UREĐAJA ZA PODZEMNU NAVIGACIJU U PODZEMNIM PROSTORIMA

Za voditelja ovog diplomskog rada imenuje se u smislu Pravilnika o diplomskom ispitu izv. prof. dr. sc. Dalibor Kuhinek, izvanredni profesor Rudarsko-geološko-naftnog fakulteta Sveučilišta u Zagrebu.

Voditelj

(potpis)

Izv. prof. dr. sc. Dalibor Kuhinek

(titula, ime i prezime)

Predsjednik povjerenstva za završne i diplomske ispite

(potpis)

Doc. dr. sc. Dubravko Domitrović

(titula, ime i prezime)

Prodekan za nastavu i studente

(potpis)

Izv. prof. dr. sc. Dalibor Kuhinek

(titula, ime i prezime)

Sveučilište u Zagrebu Diplomski rad

Rudarsko-geološko-naftni fakultet

SLOŽENOST DIJKSTRA ALGORITMA TE PRIMJENA KOD UREĐAJA ZA PODZEMNU NAVIGACIJU U PODZEMNIM PROSTORIMA

Bruno Ćesić

Rad izrađen: Sveučilište u Zagrebu
Rudarsko-geološko-naftni fakultet
Zavod za rudarstvo i geotehniku
Pierottijeva 6, 10000 Zagreb

Sažetak

Subterranean guidance system (SGS) je uređaj koji bi trebao zamijeniti konvencionalne načine orijentacije i navigacije pri spašavanju u podzemnim objektima. Unatoč zamisli osnovne primjene u podzemnim rudnicima, SGS je primjenjiv i u drugim okolnostima. SGS za svoj rad koristi Djikstra algoritam, jedan od najprepoznatijih algoritama u računalnim znanostima i operativnom istraživanju. Njegova upotreba važna je za problem najkraćeg puta, vrlo efektivno daje optimalna rješenja i pri velikim skupovima podataka. Napravljena je analiza kompleksnosti navedenog algoritma, te provedeni su testovi na nizu testnih mreža.

Ključne riječi: Subterranean guidance system (SGS), Djikstra algoritam (DA), Raspberry Pi 4 B, kompleksnost algoritma

Diplomski rad sadrži: 33 stranice, 15 slika, 5 tablica i 13 referenci.

Jezik izvornika: hrvatski

Pohrana rada: Knjižnica Rudarsko-geološko-naftnog fakulteta
Pierottijeva 6, 10000 Zagreb

Mentor: Izv.prof.dr.sc. Dalibor Kuhinek, izvanredni profesor RGNF

Ocjenjivači: Dr.sc. Dalibor Kuhinek, izvanredni profesor RGNF-a
Dr.sc. Mario Klanfar, docent RGNf-a
Dr.sc. Vinko Škrlec, docent RGNf-a

Datum obrane: 27.11.2020., Rudarsko-geološko-naftni fakultet, Sveučilište u Zagrebu

University of Zagreb
Faculty of
Mining, Geology
and Petroleum Engineering

Master'sThesis

DJIKSTRA ALGORITHM COMPLEXITY AND ITS APPLICATION IN UNDERGROUND NAVIGATION DEVICE FOR UNDERGROUND FACILITIES

Bruno Ćesić

Thesis completed at: University of Zagreb
Faculty of Mining, Geology and Petroleum Engineering
Department of Mining Engineering and Geotechnics,
Pierottijeva 6, 10 000 Zagreb

Abstract

Subterranean guidance system (SGS) is a device that should replace conventional methods of orientation and navigation during rescue in underground facilities. Despite the idea of basic application in underground mines, SGS is applicable in other circumstances as well. SGS uses the Djikstra algorithm for its work, one of the most recognized algorithms in computer science and operational research. Its use is important for the shortest path problem, for whom it provides very effectively optimal solutions even for large data sets. The complexity analysis of this algorithm was made and tests were performed on series of test networks.

Keywords: Subterranean guidance system (SGS), Djikstra algorithm (DA), Raspberry Pi 4 B, algorithm complexity

Thesis contains: 33 pages, 15 figures, 5 tables and 13 references

Original in: Croatian

Thesis is deposited at: Library of Faculty of Mining, Geology and Petroleum Engineering, Pierottijeva 6, Zagreb

Supervisor: Associate Professor Dalibor Kuhinek, PhD

Reviewers: Associate Professor Dalibor Kuhinek, PhD
Assistant Proffesor Mario Klanfar, PhD
Assistant Proffesor Vinko Škrlec, PhD

Date of defense: November 27, 2020, Faculty of Mining, Geology and Petroleum Engineering, University of Zagreb

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS KORIŠTENIH KRATICA

SGS – uređaj za podzemnu navigaciju pri spašavanju (*Subterranean Guidance System*)

GIS – Geografski informacijski sustav

GPIO – Ulaz/Izlaz opće namjene (*General Purpose Input/Output*)

MSHA – regulatorno tijelo Sjedinjenih Američkih Država za zaštitu na radu u industriji rudarstva (*Mine Safety and Health Administration*)

MINERS – međunarodni projekt financiran of Europskog Instituta za tehnologiju (EIT) za studente rudarstva (*Mine Emergency Response And Rescue School*)

DA – Dijkstrin algoritam

IT – Informatička tehnologija

LF – niska frekvencija (*low frequency*)

HF – visoka frekvencija (*high frequency*)

UHF – ultra visoka frekvencija (*ultra high frequency*)

MDS – višedimenzionalno skaliranje (*multidimensional scaling*)

SBL – naziv algoritma za segmentaciju podataka (*Sparse Bayesian Learning*)

GPS – globalni pozicijski sustav

CAD – dizajn s pomoću računala (*Computer Aided Design*)

VRML – jezik modeliranja virtualne stvarnosti (*Virtual Reality Modeling Language*)

NPS – Nacionalna parkovna agencija, SAD (*National Park Service*)

HDMI - Multimedijsko sučelje visoke definicije

POPIS KORIŠTENIH OZNAKA

OZNAKA	JEDINICA	OPIS
m	g	masa
l	m	duljina
f	kHz, MHz, GHz	frekvencija
-	Mb/s	brzina prijenosa podataka

1. UVOD

Tema ovog rada je ispitati prototip uređaja SGS (eng. Subterranean Guidance System – SGS), koji bi mogao postati dio osnovne opreme spasilačke službe u podzemnim prostorijama. Svrha SGS-a je jednostavna i brza navigacija kroz podzemne prostorije za sve članove spasilačke službe, neovisno o njihovom prethodnom poznavanju tlocrta podzemnog objekta i nepredvidivim uvjetima koji se događaju u takvim situacijama (Brozović i dr., 2020).

U ovom radu prikazan je razvoj uređaja i izrada prototipa (poglavlja preuzeta iz rada za rektorovu nagradu), prikazan primjer rada Dijkstra algoritma te je obrađena tematika složenosti algoritma i primjer procjene složenosti što je primijenjeno na Dijkstrin algoritam. Provedeno je testiranje SGS-a te su obavljeni testovi na mreži sa 20, 40, 80, 160 i 200 točaka te napravljena usporedba.

Ovaj diplomski rad predstavlja nastavak rada koji je predan na natječaj za rektorovu nagradu za akademsku godinu 2019./2020. studenata Brozović, Ćesić i Weiser pod nazivom *Razvoj uređaja za podzemnu navigaciju pri spašavanju (Development of Subterranean Guidance System - SGS)* (Brozović i dr., 2020).

2. RAZVOJ UREĐAJA

Nakon razrade zamisli svojstava i funkcionalnosti te sučelja uređaja, obavljena je analiza postojećih tehnologija i algoritama koji se mogu koristiti za izradu prototipa. Odabran je optimalan algoritam te je započet rad na izradi prototipa i nakon toga niz pokušaja programiranja i testiranja uređaja dok se nije dobila stabilna i zadovoljavajuća funkcija prototipa (Brozović i dr., 2020).

2.1. Opis željenih funkcionalnosti uređaja

Subterranean guidance system (SGS) je uređaj koji bi trebao zamijeniti konvencionalne načine orijentacije i navigacije pri spašavanju u podzemnim objektima. Osnovna primjena zamišljena je za rudnike, ali nije i ograničena samo za njih (Brozović i dr., 2020).

Cilj SGS-a je omogućiti navigaciju kroz nepoznate podzemne prostore, te praćenje lokacije spasilačke grupe u stvarnom vremenu bez ovisnosti o vanjskim i unutrašnjim čimbenicima. SGS može zamijeniti karte i slične metode navigacije u potpunosti. Također, mogao bi u kasnijim stadijima razvoja ukloniti čimbenik poznavanja podzemnog objekta, te omogućiti da se zapažanja tijekom spašavanja mogu zapisati u uređaju na interaktivnoj digitalnoj karti. U konačnoj inačici SGS-a sve potrebne informacije i karakteristike moći će se pohraniti u uređaju, te prikazati u sučelju po potrebi (Brozović i dr., 2020).

2.2. Dijkstrin algoritam u SGS-u

Dijkstrin algoritam (DA) je jedan od najpopularnijih algoritama u računalnim znanostima i operativnom istraživanju. 1959. godine Edsger W. Dijkstra objavio je rad u časopisu „*Numerische Mathematik*“. U tom radu Dijsktra je predložio algoritme za dva teoretska problema grafova: *problem minimalnog razapinjajućeg stabla* i *problem najkraćeg puta*. Upravo je Dijkstrin algoritam za problem najkraćeg puta jedan od najprepoznatijih algoritama u računalnim znanostima i operativnom istraživanju. Glavni razlog popularnosti

Dijkstrinog algoritma je taj da ga se smatra najbitnijim i najkorisnijim dostupnim algoritmom za generiranje optimalnih rješenja za veliku skupinu problema najkraćeg puta. Tradicionalne skupine problema najkraćeg puta su područja računalnih znanosti, umjetne inteligencije, operativnih znanosti i upravljačkih znanosti. Navedena područja smatraju se problemima najkraćeg puta jer se svaki problem kombinatoričke optimizacije može definirati kao problem najkraćeg puta (Brozović i dr., 2020).

Međutim, nove skupine problema zadnjih godina dobivaju na značaju zbog praktične povezanosti ove problematike i primjene na Geografskim Informacijskim Sustavima (GIS). Primjer takve primjene je *online* proračunavanje uputa u stvarnom vremenu tijekom vožnje (Brozović i dr., 2020).

Princip rada DA je da na zadanom grafu s poznatim točkama odredi najkraći put od zadane početne točke A do zadane konačne točke B. Poznate točke moraju biti definirane preko međusobne udaljenosti ukoliko između njih postoji put ili preko koordinata na apscisi i ordinati. Preko poznatih koordinata se može proračunati udaljenost točaka između kojih postoji put. Te udaljenosti mogu biti unaprijed proračunate ili izmjerene, a zatim i zapisane. Druga mogućnost je da se osigura način da s unosom koordinata točaka bude automatski proračunata vrijednost udaljenosti i zapisana. Algoritam prolazi sve točke na grafu, zbraja njihove međusobne udaljenosti, te potom razvrstava dobivene udaljenosti od najmanje do najveće. Kada se analiziraju sve vrijednosti, iskazuje se onaj put čija suma ima najmanji iznos (Sniedovich, 2006). Dijkstrin algoritam je prepoznat kao idealno rješenje i podloga za izradu SGS-a, obzirom da na jednostavan i pouzdan način pronalazi najkraći put, što je osnovna ideja principa rada SGS-a. Dijkstrin algoritam koji SGS koristi nalazi se u prilogu 3 (Brozović i dr., 2020).

DA ima dva glavna ograničenja. Prvo ograničenje je činjenica da svaki put pri pokretanju pretražuje sve čvorove. Za manje rudnike to nije problem, s obzirom da nemaju veliki broj čvorova i hodnika. Međutim, pri većem broju čvorova i hodnika pokretanje DA je zahtjevno s gledišta procesorske snage. Drugo ograničenje je nemogućnost uključivanja negativnih vrijednosti u algoritam. Ovo ograničenje nije presudno za rad SGS-a, ali otežava eventualna poboljšanja dodavanjem negativnih vrijednosti pojedinim točkama ili hodnicima za precizniju navigaciju i/ili neku drugu funkciju samog uređaja (Rehman i dr., 2019).

2.3. Logički princip rada uređaja

Kostur programa je Dijkstrin algoritam. Sve točke, odnosno čvorišta rudnika se moraju unaprijed unijeti u program putem koordinata u Kartezijevom koordinatnom sustavu. Oblik u kojem se koordinate točaka zapisuju je N-terac. N-terci su sljedovi vrijednosti koje mogu biti bilo koje vrste. Vrijednosti se razdvajaju, odnosno označavaju cijelim brojevima i prema tome su slični listama. N-terci se razlikuju od lista po tome što su nepromjenjivi (Downey, 2015).

Primjer N-terca koji se koristi u SGS-u:

'1' = ('78','jabuka')

Pomoću gore napisanog N-terca pod imenom *1* definirala se duljina hodnika *Jabuka*. Skup više takvih N-teraca koji definiraju točke rudnika nalaze se u rječniku koji će definirati rudnik za simulaciju u poglavlju 5. Takav rječnik koji sadrži više vrijednosti za jedan ključ naziva se multirječnik (Beazley i Jones, 2013). Svaki zapis u rječniku Pythona može se podijeliti na dva dijela:

ključ : vrijednost

Ključ je dio zapisa preko kojega se dohvaća njemu zadana vrijednost (Hruška, 2018).

Odabirom i potvrdom početne i krajnje točke na prozoru sa slike 4- program napravi nekoliko radnji:

- pohranjuje početnu točku u zasebnu varijablu za kasnije korištenje
- poziva DA da pronađe najkraći put i pohranjuje ga u novu varijablu
- uzima prvu točku i prvu slijedeću točku koja se nalazi na najkraćem putu, te poziva drugi algoritam koji generira dio karte s te dvije točke
- sprema generiranu kartu

Algoritam za generiranje karte uzima točke, pronalazi ih na x-y grafu i pridodaje im imena. Zatim algoritam uzima koordinate točaka, pridodaje im vrijednost od +0,2 ili –0,2 ovisno o poziciji u rudniku. Broj točaka koje se koriste pri generiranju ovisi o značaju za segment karte koji generira. Razlog promjene vrijednosti je preglednost generirane karte. Generirana karta se sprema u .jpg formatu, te ju onda program učitava i prikazuje u radnom prozoru (Brozović i dr., 2020).

Po završetku svih gore navedenih radnji korisniku se prikaže drugi prozor s generiranom kartom i uputama, Generirana karta prikazuje točku s koje je korisnik došao, točku na kojoj se nalazi i točku prema kojoj treba krenuti. Treba uzeti u obzir da je generirana karta sekundarni mehanizam orijentacije, a kao primarni mehanizam služi tekstualni dio ispod karte. Tekstualni dio opisuje korisniku sve informacije vidljive na karti, te mu daje uputu prema kojem hodniku treba krenuti uz naziv tog hodnika. Nazivi hodnika nisu nepoznanica u praksi, posebice ako se radi o većem rudniku gdje se to provodi radi jednostavnosti (Brozović i dr., 2020).

Primjer teksta:

Udaljenost koju trebate prijeći je 384 m.

Prošla točka je 1. Nalazite se na točki 2.

Krenite hodnikom Borovnica do točke 11.

Korisnik na ovome prozoru može obavijestiti program kako:

a) je moguće proći kroz navedeni hodnik – gumb „*Hodnik je prohodan*“
b) nije moguće proći kroz navedeni hodnik – gumb „*Hodnik ima prepreku*“

Odabirom opcije a) program uzima prošlu točku, točku na kojoj se korisnik nalazi i slijedeću točku i ponovo poziva algoritam za generiranje karte. Generirana karta ovaj put prikazuje tri točke. Korisniku se prikazuje radni prozor s novom generiranom kartom, novim uputama i informacijama u tekstualnom obliku, te opet ima iste dvije opcije (Brozović i dr., 2020).

Odabirom opcije b) program ponovo poziva DA radi pronalaska novog najkraćeg puta uzimajući u obzir raniju bilješku da je put prepriječen. Novi najkraći put se pohranjuje, te se ponovo poziva algoritam za generiranje karte (Brozović i dr., 2020).

Cijeli ciklus s ove dvije opcije se ponavlja sve dok:

1) korisnik nije došao do zadane krajnje točke
2) nisu svi hodnici do zadane krajnje točke prepriječeni

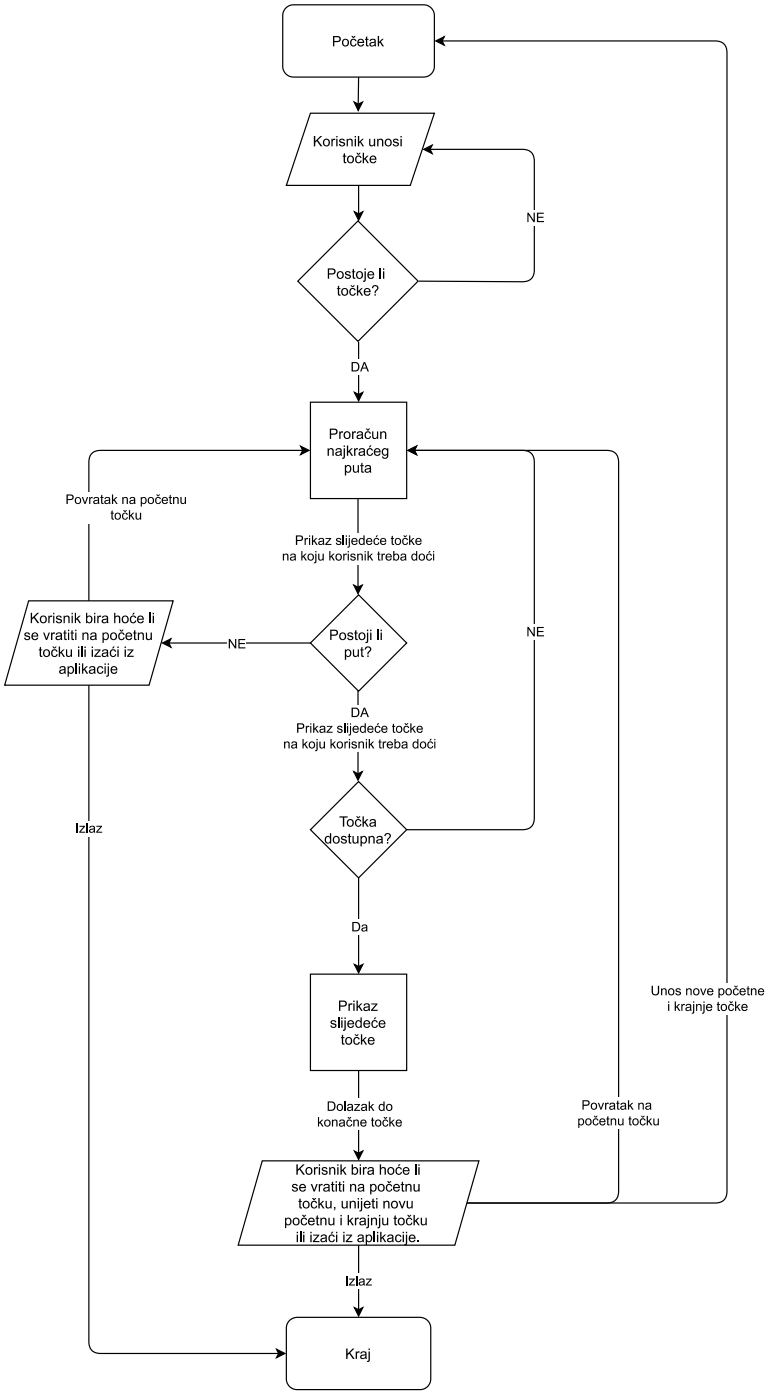
Ako je korisnik došao do zadnje krajnje točke (1), korisniku su ponuđene dvije opcije:

a) povratak na početnu točku (primjerice ulaz rudnika)
b) zatvoriti program

U slučaju opcije a) program očitava početnu točku iz varijable koja je spremljena pri početku rada programa i pamti ju kao krajnju. Točka na kojoj se korisnik nalazi uzima se kao početna, te se cijeli program pokreće ispočetka prema gore opisanom ciklusu. Povratkom na početnu točku korisnik može pokrenuti novi ciklus pritiskom na gumb „*Novi zadatak*“ ili zatvoriti program pritiskom na tipku „*Izlaz*“ (Brozović i dr., 2020).

Opcija b) zatvara program i gasi uređaj.

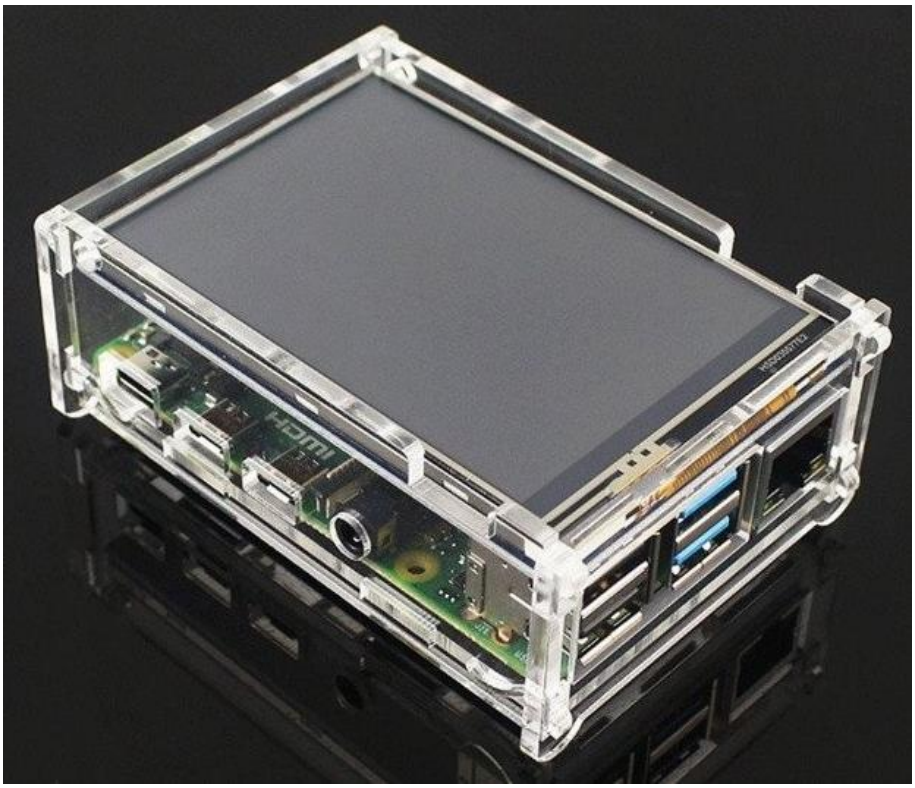
Ako su svi putevi prepriječeni (2), korisnik ima opcije vratiti se na početnu točku ili ugasiti uređaj. Početna točka tretira se kao krajnja, a trenutna lokacija (zadnja točka do koje je korisnik uspješno došao) učitava se kao početna. S tim ulaznim podatcima pokreće se ranije opisan ciklus. Slika 2-1 prikazuje dijagram tijeka rada programa SGS (Brozović i dr., 2020).



Slika 2- 1: Logička shema programa SGS-a (Brozović i dr., 2020)

3. IZVEDBA PROTOTIPA

Radi optimizacije troškova, lakše dostupnosti nekih komponenti te šire podrške za hardverske i softverske komponente budućeg uređaja odlučeno je koristiti mikroračunalo Raspberry Pi i programski jezik Python 3.7.0. Nabavljene su potrebne komponente i na slici 3-1 prikazan je sklopljen prototip SGS-a koji je korišten tijekom izrade programa testiranja (Brozović i dr., 2020).



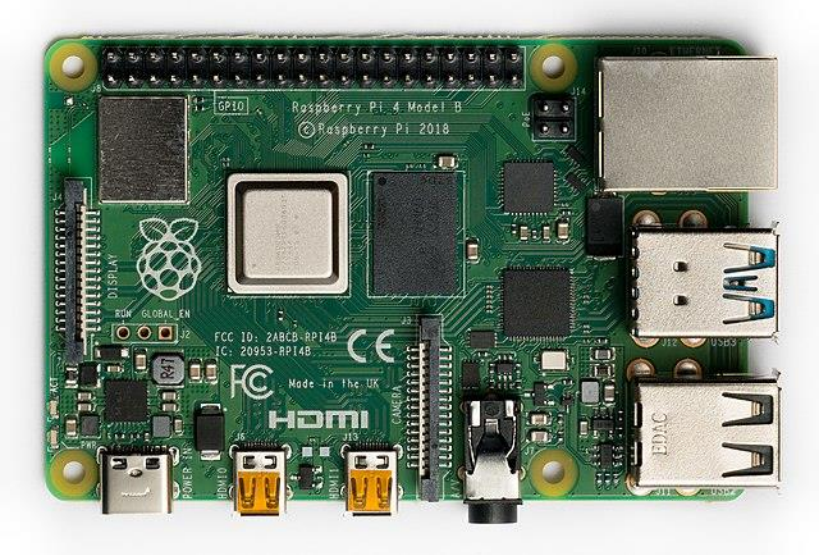
Slika 3- 1: Slika hardverskih komponenti uređaja SGS (Solectroshop, 2020)

3.1. Hardver uređaja

Odabrani hardware za prvu inačicu SGS-a je Raspberry Pi 4 model B koji je prikazan na slici 3-2. Raspberry Pi je serija jednopločnih računala proizvedenih u Ujedinjenom Kraljevstvu čija je primarna namjena omogućiti jeftino i pouzdano računalo za učenje osnova računalnih znanosti. Prednost korištenja Raspberry-a za prototip SGS-a je njegov operativni sustav Raspberry Pi OS (tzv. Raspbian) koji je baziran na otvorenom Linux OS-

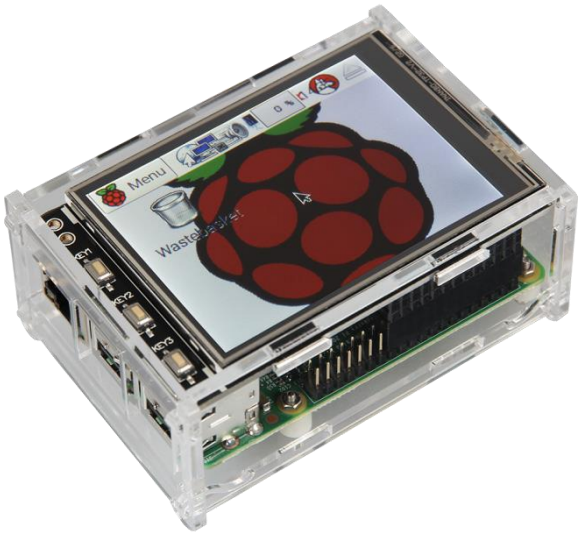
u. Raspbian dolazi s već unaprijed instaliranim programskim jezikom Pythonom što je znatno olakšalo povezivanje programa SGS-s s Raspberry-em (Raspberry Pi Foundation, 2020).

Na Raspberry Pi 4 model B računalu nalazi se Cortex A72 mobilni procesor koji je dio Broadcom BCM2711. Radni takt procesora je 1,5 GHz što je relativno sporo za današnje mobilne procesore čiji radni takt redovito prelazi 2.5 GHz.



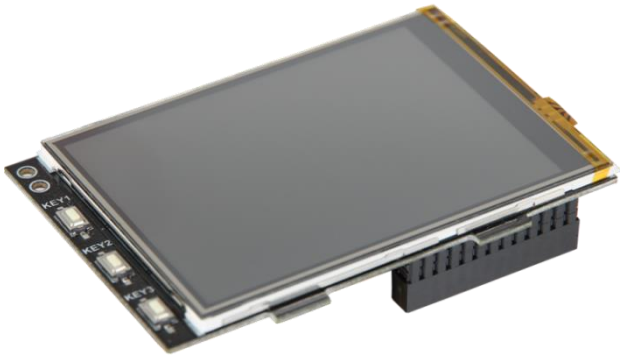
Slika 3- 2: Raspberry Pi 4 B (Wikipedia, 2019)

Korišteno je akrilno kućište tvrtke Joy-IT, prikazano na slici 3-3. Tvrtka Joy-IT izradila je ovaj tip kućišta specifično za Raspberry modele B+, 2B, 3B, 3B+ i 4, te se koristi u kombinaciji s njihovim zaslonom na dodir od 3,2 inča ili 3,5 inča. Po sastavu je od transparentnog akrila visoke kvalitete. S obzirom da različiti modeli Raspberry-a imaju drugačije dimenzije, kućište se može podesiti dodatcima koji dolaze u kompletu s kućištem. Masa kućišta je 70 g (Joy-IT, 2020).



Slika 3- 3: Joy-IT Akrilno kućište (Joy-IT, 2020)

Uređaj koristi zaslon na dodir prikazan na slici 3-4 koji je proizvela tvrtka Joy-IT. Dijagonala zaslona je 3,2 inča s maksimalnom rezolucijom od 320x240 pixela. Spaja ga se s Raspberry PI-em putem standardnih GPIO utora. Zaslon sadrži tri gumba koji se mogu programirati. Ovaj zaslon je kompatibilan s Raspberry Pi A, B, B+, 2B, 3B, 3B+ i 4B isto kao i kućište, što ih čini potpuno međusobno kompatibilnim za povezivanje software-a SGS-a na bilo koju drugu inačicu Raspberry Pi operativnog sustava. Masa zaslona je 52 g (Joy-IT, 2020).



Slika 3- 4: Joy-IT Zaslon na dodir (Joy-IT, 2020)

4. DIJKSTRA PRIMJER RADA

DA u osnovi započinje na čvoru koji je odabran (izvorni čvor) i analizira graf kako bi pronašao najkraći put između tog čvora i svih ostalih čvorova u grafu. Osnovna ideja DA je kontinuirano uklanjanje dužih puteva između početnog čvora i svih ostalih mogućih odredišta. Algoritam prati trenutno najkraću poznatu udaljenost od svakog čvora do izvornog čvora i ažurira te vrijednosti ako pronađe kraći put. Nakon što algoritam pronađe najkraći put između izvornog čvora i drugog čvora, taj čvor se označava kao "posjećen" i dodaje na put *S*. Proces se nastavlja dok se svi čvorovi na grafikonu ne dodaju na put *S*. Na taj način dobiven je put koji povezuje izvorni čvor sa svim ostalim čvorovima slijedeći najkraći mogući put do svakog čvora (Freecodecamp, 2020).

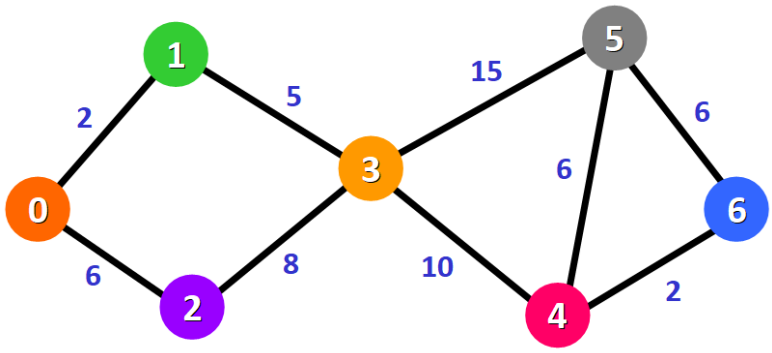
```
1  function Dijkstra(Graph, source):
2
3      create vertex set Q
4
5      for each vertex v in Graph:
6          dist[v] ← INFINITY
7          prev[v] ← UNDEFINED
8          add v to Q
10     dist[source] ← 0
11
12     while Q is not empty:
13         u ← vertex in Q with min dist[u]
14
15         remove u from Q
16
17         for each neighbor v of u:
18             alt ← dist[u] + length(u, v)
19             if alt < dist[v]:
20                 dist[v] ← alt
21                 prev[v] ← u
22
23     return dist[], prev[]
```

Slika 4- 1: Pseudokod DA(Wikipedia, 2020)

Na slici 4-1 prikazan je pseudokod DA. *Q* označava prazan skup, te se udaljenosti svih čvorova postavljaju u beskonačnost. Čvorovi *v* se dodaju u skup *Q*, dok se prethodni čvorovi postavljaju u nulu, sve dok je *Q* neprazan skup. Minimalna vrijednost čvora u skupu *Q* se postavlja u varijablu *u*. Dok ima čvorova u skupu *Q*, algoritam s njegovog početka uzima najmanju vrijednost čvora *u*. Zatim, u liniji 16 za svakog susjeda *v* od vrijednosti *u*, koja je trenutna najmanja vrijednost izvršava sljedeće: Prvo se varijabla *alt* postavlja na zbroj

trenutne minimalne vrijednosti *u* i udaljenosti *u* od susjeda *v*, ta udaljenost predstavlja brid, koja se razmatra. Ako je udaljenost *alt* manja od vrijednosti susjednog čvora *v*, tom susjednom čvoru se mijenja vrijednost na iznos *alt*, a samim time njegov prethodnik postaje najmanji čvor te se izbacuje iz skupa *Q*. Petlja se ponavlja sve dok se skup *Q* ne isprazni. (Žunić, 2018).

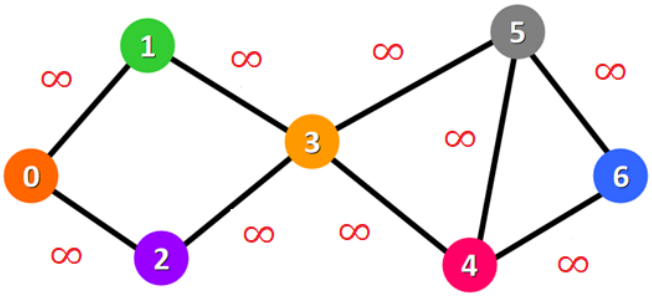
Na slici 4-2 nalazi se mreža, odnosno graf, s točkama koje su numerirane od 0 do 6 i stazama koje povezuju navedene. Svaka staza ima pripadnu vrijednost koja označava udaljenost odnosno težinu između dvije točke koje povezuje.



Slika 4- 2: Primjer grafa sa 7 točaka **(**Freecodecamp,2020**)**

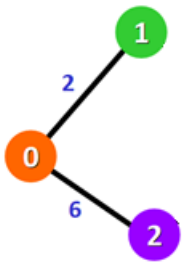
Prije nego što algoritam započne s proračunom svih putova na grafu, svi čvorovi su postavljeni na beskonačnu udaljenost, odnosno težina puta je beskonačna (slika 4-3). Osim izvora, odnosno točke 0, čija je udaljenost također 0. U varijabli put se nalazi samo točka 0.

S{0}



Slika 4- 3: Primjer DA korak 1 (Freecodecamp,2020)

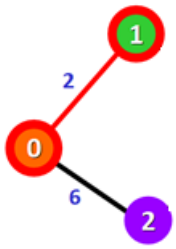
Na slici 4-4 DA započinje sa analizom, točka 0 je povezana sa točkama 1 i 2. Težina puta do točke 1 iznosi 2, a težina puta do točke 2 iznosi 6. Sve ostale točke su nepoznate, drugim riječima, beskonačne udaljenosti.



Slika 4- 4: Primjer DA korak 2 (Freecodecamp,2020)

Nakon uspoređivanja težine puta između točaka, odabrana je točka sa najmanjom udaljenosti, a to je točka 1. Točka 1 je označena kao početna (slika 4-5) i dodana je na put *S*.

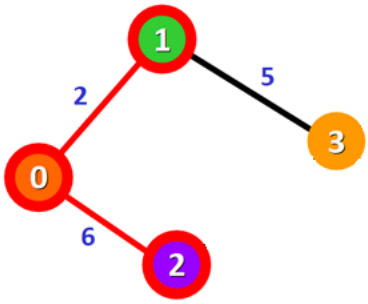
Najkraći put do točke 1 je S{0,1}



Slika 4- 5: Primjer DA korak 3 (Freecodecamp,2020)

Nadalje, točke 2 i 3 su sljedeće do kojih algoritam traži najkraći put. Težina puta od točke 0 do točke 3 iznosi 7, dok od točke 0 do točke 2 iznosi 6. Odabrana je točka 2, pošto je njezina udaljenost od točke 0 najmanja. Točka 2 sada je označena kao posjećena i dodana je na put *S* (slika 4-6).

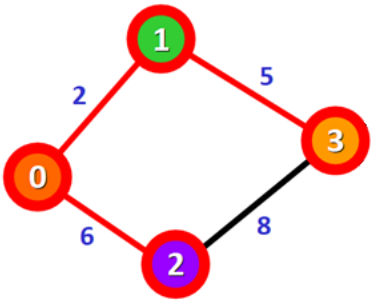
Najkraći put do točke 2 je S{0,2}.



Slika 4- 6: Primjer DA korak 4 (Freecodecamp,2020)

Sljedeća na redu je točka 3, do koje je moguće doći dvjema putanjama. Jedna je poznata iz prethodnog koraka, (0,1,3), gdje je težina puta 7, a nova putanja je (0,2,3) s težinom puta 14. DA odabire manju težinu, tj. kraći put. Točka 3 sada je označena kao posjećena te biva dodana na put *S* (slika 4-7).

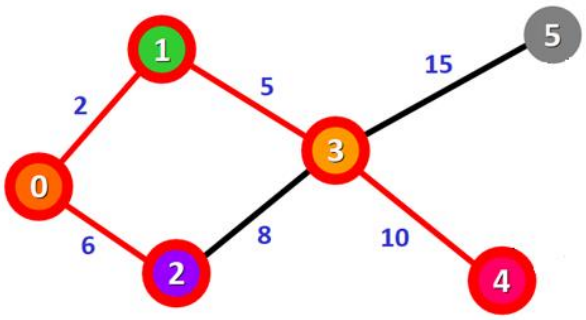
Najkraći put do točke 3 je S{0,1,3}



Slika 4- 7: Primjer DA korak 5 (Freecodecamp,2020)

Točke 4 i 5 su povezane s točkom 3. Težina puta od točke 0 do točke 4 iznosi 22, a do točke 4 iznosi 17. DA odabire kraći put a to je točka 4. Točka 4 sada je označena kao posjećena i dodana je na put *S* (slika 4-8).

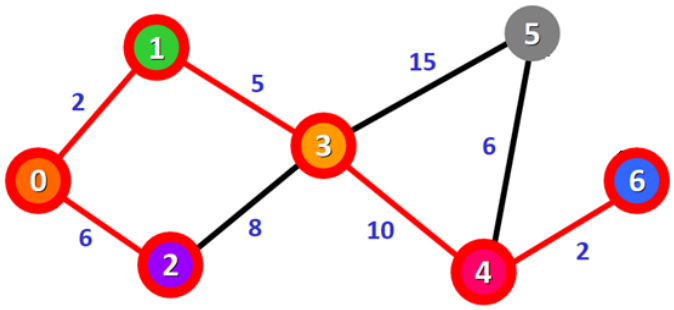
Najkraći put od točke 0 do točke 4 je *S*{0,1,3,4}.



Slika 4- 8: Primjer DA korak 6 (Freecodecamp,2020)

Točka 4 je povezana s točkama 5 i 6. Dvije su mogućnosti puta do točke 5 i jedna mogućnost puta do točke 6. Prva mogućnost puta za točku 5 je S(0,1,3,5) gdje je težina puta 22, a druga mogućnost je S(0,1,3,4,5) čija težina iznosi 23. Težina puta do točke 6 S(0,1,3,4,6) iznosi 19, stoga algoritam odabire točku 6 kao najjeftiniju. Točka 6 sada je označena kao posjećena i dodana je na put *S* (slika 4-9).

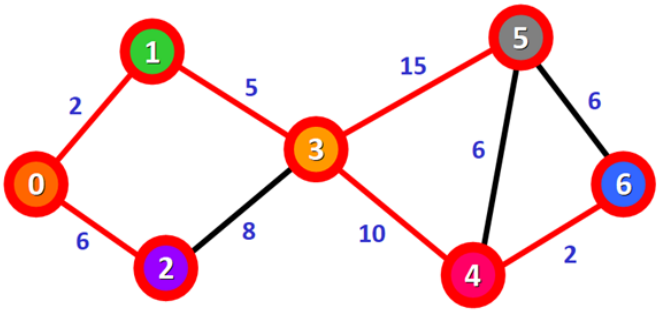
Najkraći put od točke 0 do točke 6 sada je S{0,1,3,4,6}



Slika 4- 9: Primjer DA korak 7 (Freecodecamp,2020)

Zadnja neposjećena točka je 5 , za koju postoji još jedna dodatna mogućnost puta. Kao što je spomenuto, težine puta do točke 5 iznose 22 i 23. Nova mogućnost je S(0,1,3,4,6,5) čija težina iznosi 25. Odabire se najmanja težina puta za točku 5, koja je sada označena kao posjećena te dodana na put *S* (slika 4-10).

Najkraći put od točke 0 do točke 5 S{0,1,3,5}.



Slika 4- 10: Primjer DA korak 8 (Freecodecamp,2020)

5. SLOŽENOST ALGORITMA

5.1. Vremenska složenost.

Analiza algoritma važan je dio šire teorije računske složenosti, koja daje teorijske procjene resursa (prostora i vremena) potrebnih bilo kojem algoritmu koji rješava zadani računski problem. U računalnim znanostima vremenska složenost algoritma daje količinu vremena koja je potrebna da algoritam ili program dovrši njegovo izvršavanje, a obično se izražava oznakom Big-O (O). Big-O, koji se obično zapisuje kao O, asimptotska je notacija u najgorem slučaju, što je ujedno i gornja granica rasta za zadanu funkciju. Vremenska složenost algoritma označava ukupno vrijeme koje je potrebno programu da odradi određeni zadatak (Introprogramming, 2020).

Vremenska složenost najčešće se procjenjuje brojanjem elementarnih koraka izvedenih bilo kojim algoritmom do kraja izvršavanja zadatka. Budući da se izvedba algoritma može razlikovati vrstama ulaznih podataka, stoga se za algoritam obično koristi najgora vremenska složenost algoritma, jer je to maksimalno vrijeme potrebno za bilo koju ulaznu veličinu (Studytonight, 2020).

5.2. Tipične složenosti algoritma:

5.2.1. Konstantna kompleksnost O(1)

Potreban je konstantan broj koraka za izvođenje zadane operacije (na primjer 1, 5, 10 ili neki drugi broj) i taj broj ne ovisi o veličini ulaznih podataka (Introprogramming, 2020).

5.2.2. Logaritamska kompleksnost O(log(N))

Za izvršavanje zadane operacije nad N elemenata potreban je redoslijed dnevničkih (N) koraka, gdje je baza logaritma najčešće 2. Na primjer, ako je N = 1 000 000, algoritam složenosti O (log (N)) učinio bi oko 20 koraka (s konstantnom preciznošću). Budući da baza

logaritma nije od vitalne važnosti za redoslijed brojanja operacija, obično se izostavlja (Introprogramming, 2020).

5.2.3. Linearna kompleksnost O(N)

Potrebna je gotovo ista količina koraka kao i broj elemenata za izvođenje operacije nad N elementima. Na primjer, ako imamo 1 000 elemenata, potrebno je oko 1 000 koraka. Linearna složenost znači da broj elemenata i broj koraka međusobno ovise linearno, primjerice, broj koraka za N elemenata može biti N / 2 ili 3 * N (Introprogramming, 2020).

5.2.4. Kvadratna kompleksnost O(N^2)

Potreban je redoslijed N^2 broja koraka, gdje je N veličina ulaznih podataka, za izvođenje zadane operacije. Na primjer, ako je N = 100, potrebno je oko 10 000 koraka. Zapravo imamo kvadratnu složenost kada je broj koraka u kvadratnoj vezi s veličinom ulaznih podataka. Primjerice, za N elemenata koraci mogu biti reda 3 * N2 / 2 (Introprogramming, 2020).

Navedene složenosti najčešće predstavljaju prosječan broj koraka koji je potreban za izvođenje algoritma. U praksi postoje i veće složenosti od ovih koje su spomenute, ali nisu navedene pošto kompleksnost DA nije toliko visoka.

Brzina izvršavanja programa ovisi o složenosti algoritma koji se izvršava. Ako je složenost mala, program će se brzo izvršiti čak i za velik broj elemenata. Ako je složenost velika, program će se izvoditi polako ili čak neće raditi za velik broj elemenata (Introprogramming, 2020).

U praksi složenost algoritama obično se procjenjuje u najgorem slučaju (najnepovoljniji scenarij). Drugim riječima, u prosjeku algoritmi mogu raditi brže, ali u najgorem slučaju rade s procijenjenom složenošću, a ne sporije.

5.3. Primjer procjene složenosti

Za slučaj pretraživanja u nizu, u prosjeku možemo očekivati da će se provjeriti polovica njegovih elemenata, dok se ne pronađe onaj koji se traži. U prosjeku je složenost O (N / 2) = O (N) - linearna, jer se pri procjeni složenosti ne uzimaju u obzir konstante. U najboljem slučaju imamo konstantnu složenost O (1), jer je napravljen samo jedan korak i element je izravno pronađen. Za najgori slučaj, svi elementi niza moraju biti provjereni, stoga je spomenutome složenost O (N) – linearna (Introprogramming, 2020).

5.4. Vremenska složenost DA

Vremenska složenost DA na grafu sa putevima E i čvorovima V može se izraziti povezanošću |E| i |V|. Vremenska složenost DA u najgorem slučaju iznosi O (V^2), dok je u najboljem slučaju O (E + Vlog(V)), što vrijedi samo za binarne hrpe. Pošto se kod testiranja algoritama uvijek računa s najgorim mogućim slučajem, može se reći da je kompleksnost DA O (V^2).

6. TESTIRANJE SGS-A

Prethodnim testiranjem SGS-a primijećeno je da uređaj ne radi fluidno odnosno da postoje smetnje u radu, no za vrijeme rada uređaj se poprilično zagrijava. Cilj ovog testiranja je pokušaj dosezanja limita uređaja, ali i utvrđivanje razloga njegova usporenog rada odnosno utvrditi je li problem u algoritmu, GUI-ju ili nečemu trećem.

Radi jednostavnosti testiranja test se izvodi izvan GUI-a, odnosno bez grafičkog sučelja i bez pozivanja karte. Testiranje se izvodi na osnovnom algoritmu koji pronalazi najkraći put između dvije točke i koji generira alternativne rute u slučaju da je jedan od puteva neprohodan. Izvorno, ovaj način testiranja je procesorski puno manje zahtjevan jer uređaj vrti algoritam vrlo kratko vrijeme, odnosno dok ne ostane bez puteva. Kod rada sa GUI-em uređaj cijelo vrijeme ima pokrenuto korisničko sučelje, tj. ne radi periodički što zahtjeva više procesorskih resursa.

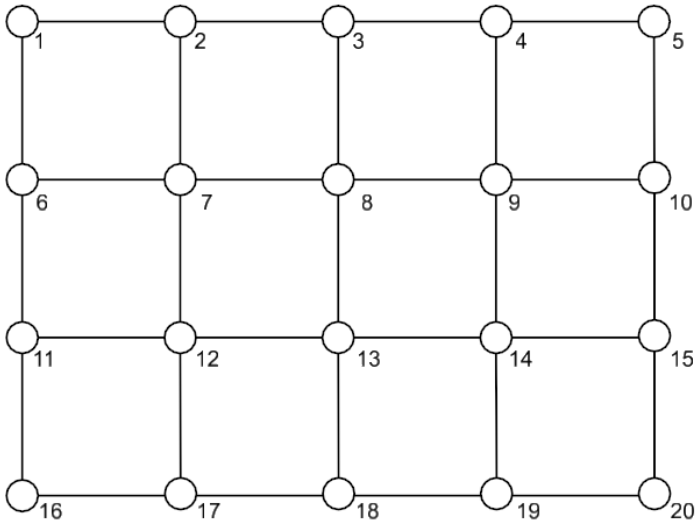
Pošto u većini slučajeva raskrižja u podzemnim prostorijama nemaju više od 4 hodnika, testiranje se izvodi na kvadratnoj mreži gdje su točke međusobno povezane sa 2, 3 ili 4 susjedne točke čime se najvjernije simulira stanje podzemnih hodnika. Dimenzije mreže koja se koristi su 20, 40, 80, 160 i 200 točaka, gdje se mjere vremena potrebna kako bi algoritam pronašao najkraći put od zadane do krajnje točke. U dobivena vremena će biti uključen najmanje jedan pokušaj traženja alternativne rute.

Svaka mreža se testira 10 puta, na način da se jednom traži put između najudaljenijih točaka, a ostali metodom nasumičnog odabira. Nakon testiranja svake mreže najvažniji je najlošiji rezultat, odnosno vrijeme, i njega se uzima kao mjerodavnog.

Test završava nakon što SGS da informaciju 'nije moguće doći do odredišta', odnosno kada se iscrpe sve mogućnosti pronalaženja alternativnih ruta, jer je na taj način napravljeno najveće opterećenje na uređaj.

6.1. Test 1

Testiranje se izvodi na mreži od 20 točaka slika 6-1. Prvo mjerenje se izvodi traženjem najkraćeg puta od točke 1 do točke 20, a ostala mjerenja metodom slučajnog odabira.



Slika 6- 1: Mreža 20 točaka

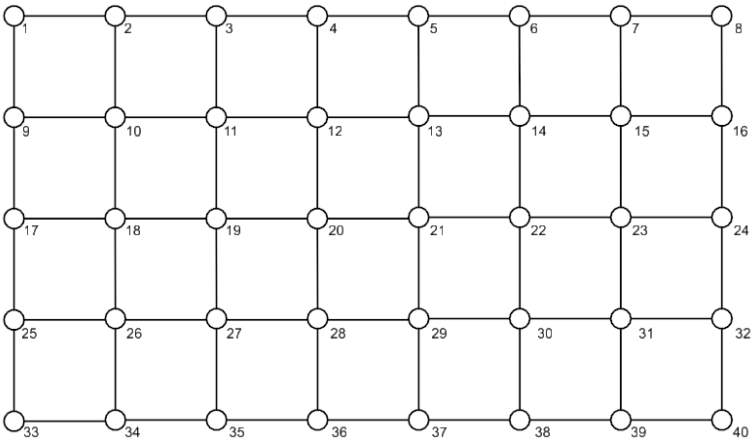
Tablica 6- 1: Rezultati testa 1

Redni broj testa	Vrijeme *t* (ms)
1.	1,19
2.	1,18
3.	1,15
4.	1,17
5.	1,18
6.	1,18
7.	1,15
8.	1,19
9.	1,17
10.	1,15
t_{max}	1,19

Rezultati mjerenja su prikazani u tablici 6-1. Sva dobivena vremena su oko 1 ms, što je ujedno i donja granica mogućnosti mjerenja, jer za svako vrijeme koje je ispod 1 ms program ne daje nikakve vrijednosti mjerenja.

6.2. Test 2

Testiranje se izvodi na mreži od 40 točaka (slika 6-2). Prvo mjerenje se radi tako da se traži udaljenost od točke 1 do točke 40, a ostala mjerenja metodom slučajnog odabira.



Slika 6- 2: Mreža 40 točaka

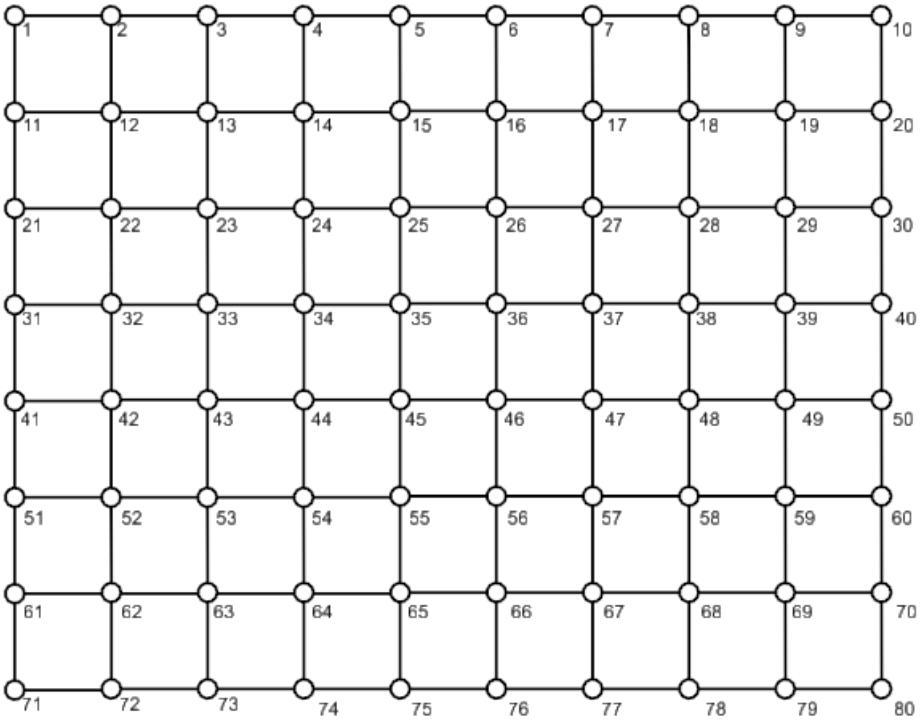
Tablica 6- 2: Rezultati testa 2

Redni broj testa	Vrijeme *t* (ms)
1.	1,82
2.	1,78
3.	1,88
4.	1,85
5.	1,84
6.	1,80
7.	1,81
8.	1,81
9.	1,89
10.	1,88
t_{max}	1,89

Rezultati svih mjerenja su ispod 2 ms, odnosno najlošije dobiveno vrijeme je 1,89 ms (tablica 6-2). Uređaj nema nikakvih problema pri radu. Dobivena vremena su bolja od očekivanih obzirom na dvostruko veći broj točaka, gdje je i sam broj alternativnih ruta koje algoritam može pronaći puno veći nego što je to bilo u slučaju prvog testa.

6.3. Test 3

Testiranje se izvodi na mreži od 80 točaka slika 6-3. Prvo mjerenje će opet biti između najudaljenijih točaka koje su u ovom slučaj 1 i 80. Broj alternativnih ruta na mreži je velik i vrijeme rada uređaja je duže, stoga se obraća pozornost i na radnu temperaturu uređaja.



Slika 6- 3: Mreža 80 točaka

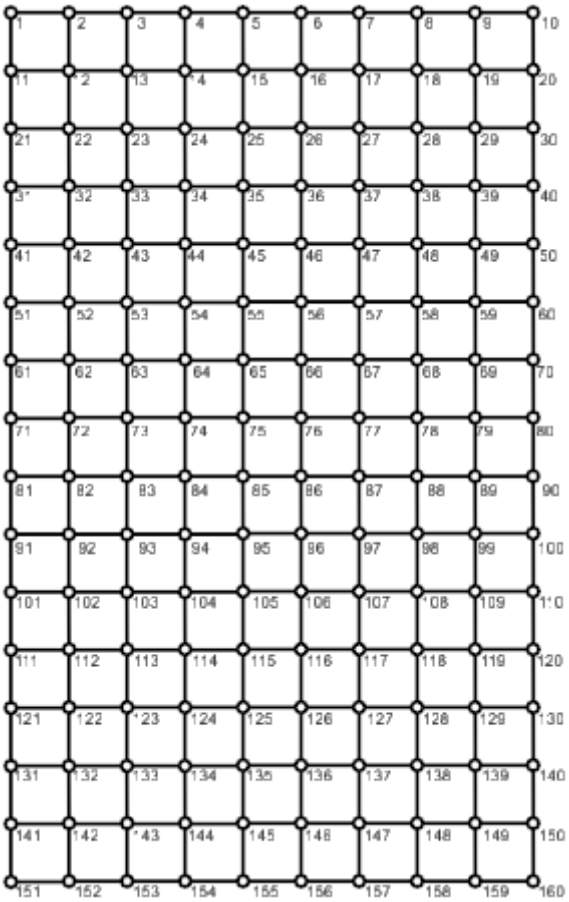
Tablica 6- 3: Rezultati testa 3

Redni broj testa	Vrijeme *t* (ms)
1.	4,10
2.	3,94
3.	3,93
4.	3,98
5.	3,89
6.	4,09
7.	4,14
8.	4,04
9.	3,92
10.	3,95
t_{max}	4,14

Najlošije dobiveno vrijeme u ovom testu iznosi 4,14 ms (tablica 6-3). Zbog toga što je izvedeno svega 10 mjerenja, postoji mogućnost da se najlošije vrijeme nije uspjelo postići. Za vrijeme testa nisu uočeni problemi u radu.

6.4. Test 4

Testiranje se izvodi na mreži od 160 točaka (slika 6-4). Prvo mjerenje sa izvodi između točaka 1 i 160. Broj alternativnih ruta je višestruko veći nego u prijašnjim testovima. U ovom testu također se obraća pozornost na temperaturu uređaja.



Slika 6- 4: Mreža 160 točaka

Rezultati

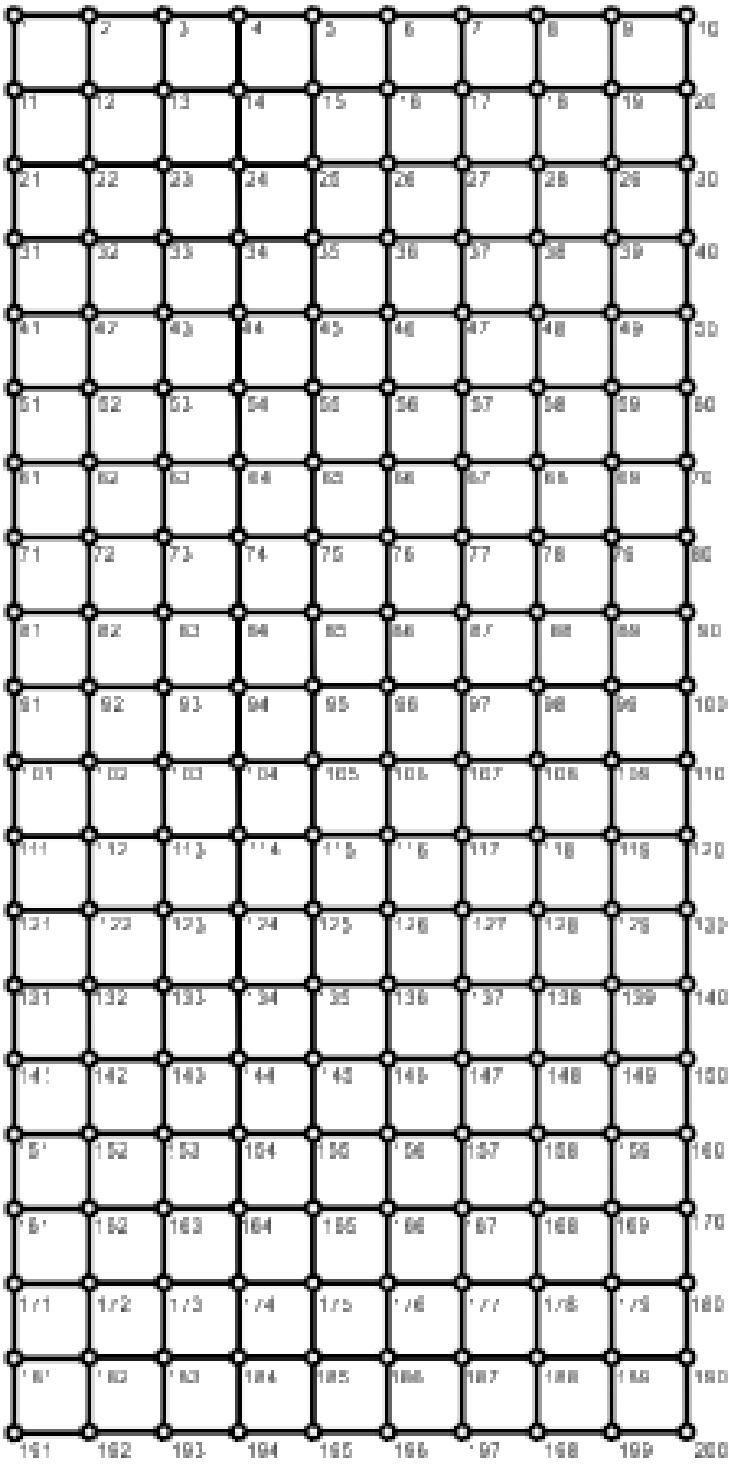
Tablica 6- 4: Rezultati testa 4

Redni broj testa	Vrijeme *t* (ms)
1.	11,50
2.	11,60
3.	11,49
4.	10,67
5.	11,84
6.	13,76
7.	11,44
8.	12,08
9.	12,38
10.	11,66
t_{max}	13,76

Najlošije postignuto vrijeme je 13,76 ms (tablica 6-4). U ovom testu primijećena su prva veća odstupanja dobivenih vremena. Najveće odstupanje između mjerenja je oko 2 ms. Za vrijeme testiranja uređaja nisu primijećene nikakve smetnje pri radu.

6.5. Test 5

Peti test se izvodi na mreži od 200 točaka slika 6-5. Prvo mjerenje se izvodi između točaka 1 i 200, kao i u testu 4 imamo puno alternativnih ruta. Opet se obraća pozornost i na temperaturu samog uređaja.



Slika 6- 5: Mreža 200 točaka

Tablica 6- 5: Rezultati testa 5

Redni broj testa	Vrijeme *t* (ms)
1	16,75
2	17,92
3	17,48
4	15,81
5	17,37
6	17,71
7	17,12
8	17,91
9	17,88
10	16,38
t_{max}	17,92

Najlošije postignuto vrijeme iznosi 17,92 ms (tablica 6-5). Gledajući razlike između dobivenih rezultata, najveće odstupanje iznosi oko 2 ms. Uređaj radi fluidno i nema nikakvih problema pri radu.

Analizom rezultata dobivena je linearna ovisnost između vremena i broja točaka na mreži, što znači da najgori slučaj nije postignut. Granice uređaja, odnosno sve mreže koje su testirane nisu dovoljno velike i komplekse da bi zadale probleme u radu SGS-a. Obzirom na dobivena vremena broj točaka s kojim bi se dosegnuo limit uređaja vjerojatno se broji u tisućama. Za vrijeme testiranja raspberry pi je bio spojen na vanjski monitor preko HDMI izlaza iz razloga što je zaslon na dodir malih dimenzija i niske rezolucije. Ono što je primijećeno za vrijeme rada sa vanjskim monitorom je malo poboljšanje u radu sa GUI-em, smetnje su bile i dalje prisutne ali u manjoj mjeri nego kod rada sa zaslonom na dodir.

Iz dobivenih podataka se može zaključiti kako problem pregrijavanja dolazi od GUI-a, dok su smetnje u radu odnosno usporenost dijelom rezultat zaslona na dodir. Problem temperature se najvjerojatnije može riješiti optimizacijom koda GUI-a dok se smetnje u radu odnosno sporost mogu riješiti ugradnjom boljeg zaslona na dodir.

Sljedeća inačica testiranja, koju bi bilo dobro napraviti, uključuje korištenje ove mreže sa grafičkim sučeljem uređaja i generiranjem karte čime bi se dodatno opteretio uređaj.

7. ZAKLJUČAK

U ovom radu je prikazan Dijkstrin algoritam za pronalazak najkraćeg puta u podzemnim prostorima gdje može postojati blokada prolaza kroz pojedine hodnike što je primijenjeno za izradu uređaja Subterranean guidance system

Prikazan je Dijsktrin algoritam, analizirana je njegova kompleksnost te je napravljen niz testiranja koliko dugo vremena traje pronalazak najkraćeg puta. Mreže od 20 čvorova imale su vrijeme reda veličine milisekunde a mreže veličine 200 čvorova imale su vrijeme reda veličine 18 ms. To su sve vremena koja su jako mala te se može zaključiti da algoritam praktički ne opterećuje procesor, već se većina resursa razvijenog uređaja troši na sučelje prema korisniku što je vjerojatni uročnik povremenog kašnjenja u radu uređaja. To znači da bi se uređaj lako nosio i sa nekoliko tisuća čvorova a sučelje prema korisniku terba pokušati optimirati.

8. LITERATURA

1. BEAZLEY D., JONES B. K. (2013): *Python Cookbook*, Third Edition, O'Reilly Media, Inc., Sebastopol, California, Sjedinjene Američke Države.
2. BROZOVIĆ, J., ĆESIĆ, B., WEISER, K. (2020): *Razvoj uređaja za podzemnu navigaciju pri spašavanju*. Rad za rektorovu nagradu. RGNF. Sveučilište u Zagrebu.
3. DOWNEY, A. (2015): *Think Python: How to Think Like a Computer Scientist*, 2nd Edition, Version 2.4.0., Green Tea Press, Needham, Massachusetts, Sjedinjene Američke Države.
4. HRUŠKA, M. (2018): *Osnove programiranja (Python)*, Sveučilište u Zagrebu, Sveučilišni računski centar, Zagreb.
5. REHMAN A. U., AWUAH-OFFEI K., BAKER D.A., BRISTOW D. (2019): *Emergency evacuation guidance system for underground miners*, Conference paper, SME Annual Meeting, Denver, Colorado, Feb. 24-27, 2018.
6. SNIEDOVICH, M. (2006): *Dijkstra's algorithm revisited: the dynamic programming connexion*, Melbourne, Australia, The University of Melbourne, Department of Mathematics and Statistics.
7. ŽUNIĆ, S. (2018): *Pregled algoritama za traženje najkraćeg puta u grafu.* Završni rad. Fakultet prometnih znanosti. Sveučilište u Zagrebu.

WWW izvori

1. Freecodecamp,2020., URL: https://www.freecodecamp.org/news/dijkstras-shortest-path-algorithm-visual-introduction/
2. Introprogramming,2020., URL: https://introprogramming.info/english-intro-csharp-book/read-online/chapter-19-data-structures-and-algorithm-complexity/
3. Joy-IT, 2020., URL: https://joy-it.net/en/
4. Raspberry Pi Foundation, 2020., URL: https://www.raspberrypi.org/
5. Solectroshop,2020.,URL: https://solectroshop.com/en/otros-accesorios-raspberry-pi/1898-caja-acrilica-para-raspberry-pi-4-modelo-b.html
6. Studytonight,2020., URL: https://www.studytonight.com/data-structures/time-complexity-of-algorithms