УНИВЕРЗИТЕТ У НОВОМ САДУ
ПРИРОДНО-МАТЕМАТИЧКИ ФАКУЛТЕТ
ДЕПАРТМАН ЗА МАТЕМАТИКУ И ИНФОРМАТИКУ

# Ердошев број

Пројекат из предмета Вештачка интелигенција

**Име и презиме: Милош Марић**

**Професор: др Милош Радовановић**

**Нови Сад, август 2023.**

# Садржај

# Увод

Пол Ердош (**Pál Erdős**) је био мађарски математичар јеврејског порекла који је дао огроман допринос у разним областима математематике: теорије бројева, теорије графова, комбинаторике, верватноће, теорије скупова… Рођен 26. марта 1913. у Будимпешти. Од својих родитеља, средњошколских професора математике, наследио је страствену љубав према математици. За свога живота објавио је око 1500 радова са 507 коаутора, број далеко већи од било ког другог математичара, научника или истраживача. 20. септембра 1996. преминуо је у Варшави.

Услед великог броја коауторстава, Пол Ердош представља битно чвориште када се посматра социјална мрежа сарадње у научним радовима. Појам *Ердошев број* представља дистанцу сарадње између Пола Ердоша и задатог аутора. Аутору који је директно сарађивао са Ердошем додељен број 1, аутору који је сарађивао са ауторем који има број 1, а притом он сам није сарађивао са Ердошем је додељен број 2, аутор који је сарађивао са ауторима чији је Ердошев број најмање 2 има Ердошев број 3. Аналогно, бројеви се додељују свим осталим ауторима. Уколико не постоји путања од Пола Ердоша и одређеног аутора, том аутору се додељује број бесконачно.

У овом пројекту имплементирана су три алгоритма за проналажење најкраће путање између два чвора социјалне мреже:

1. Breadth-first search (BFS) – Претрага у ширину
2. Iterative Deepening Depth First Search (IDDFS или IDS) – Претрага у дубину итеративним продубљивањем
3. Bidirectional Breadth-first Search (BBFS) – Двосмерна претрага у ширину

Као извор података коришћен је dblp[1] скуп података. За реализацију пројекта коришћена је *NEO4J* база података и програмски језик *Python*.

1 dblp - Digital Bibliography & Library Project

# Припрема података

## Чишћење података

Скуп података *dblp*[2] је доступан у *xml* фајлу величине 3,5 GB. Како би проверили валидност самог фајла потребно је преузети *dblp.dtd*[3] фајл и валидирати га.

Након тога, потребно је очистити све штампарске грешке. У оквиру *dblp.dtd* фајла налазе се дефинисани ентитети – специајлни знакови који не припадају енглеском алфабету. То су:

1. Ç
2. ï
3. ö
4. ü
5. ä

На многим местима они су записани на нестандардан начин. На пример, карактер Ç је у *xml* фајлу записан на следећи начин:

"{C}

те се он мора испиравити и записати онако како је то дефинисано у *dblp.dtd* фајлу, иначе касније при учитавању података у базу *Cypher Query Language* неће моћи да га обради. Такође, због каснијег пребацивања податка потребно је заменити и знакове наводника (“”) са шифром за знак наводника (&quot).

За потребе распакивања и чишћења *xml* фајла покрећемо *prepare_xml.sh*[4] Shell скрипта.

## Парсирање података

Скуп података над којим радимо је огроман (3,5 GB) и садржи мноштво различитих информација о самим радовима. Нас интересује само податак о томе ко је са киме сарађивао, те су ауторства једини тип информација који треба да извучемо из *xml* fajla и пребацимо у *csv* фајл (*dblp_authors.csv*).

*Pyhton* скрипта које покрећемо за парсирање *xml* фајла и пребацивање података у *csv* фајл је *transform_xml2csv.py*[5].

2 dblp - https://www.kaggle.com/datasets/dheerajmpai/dblp2023
3 dblp.dtd - https://dblp.org/xml/
4 prepare_xml.sh - https://github.com/momcilol/Erdos-number/blob/main/prepare_xml.sh
5 transform_xml2csv.py - https://github.com/momcilol/Erdos-number/blob/main/transform_xml2csv.py

## Унос података у базу

Када креирамо *NEO4J* базу можемо кренути да у њу уносимо податке. Најпре пребацимо *dblp_authors.csv* фајл у *import* директоријум наше базе, а затим покренемо упит који се налази у фајлу *import_cooperations.cyp*[6]. Сви чворови ће бити типа *Author*, а све везе типа *WORKED_WITH*[7].

Због величине података трајање учитавања може да потраје 1,5h – 2h. На крају добијамо граф са ~3,15 милиона чворова (аутора) и ~39,75 милиона грана (сарадњи).

6 import_cooperations.cyp - https://github.com/momcilol/Erdos-number/blob/main/import_cooperations.cyp

7 Везе се у NEO4J бази података увек чувају као усмерене, али касније можемо да игноришемо усмерење и посматрамо их као неусмерене.

# Имплементација

## Приступ бази података

Класа *DataAccess*[8] служи нам за везу са базом података. Преко драјвера из класе *GraphDatabase* из пакета *neo4j* класа *DataAccess* са своја 3 метода нам добавља информације које су потребне за извршавање претраге:

```
def check_author(self, author: str) -> bool:
    result = self.driver.execute_query("MATCH (n:Author {name: $author}) RETURN n.name", author=author)
    authors = [r.value() for r in result[0]]
    # print(authors)
    return len(authors) == 1
```

1. check_author – проверава да ли постоји аутор са датим именом

```
def find_author(self, author_name: str):
    substrings = author_name.split(" ")
    sub_dict = {f"name{i}":substrings[i] for i in range(len(substrings))}
    if len(substrings) == 0:
        return None

    sub_list = []
    where_clause = "WHERE "
    for i in range(len(substrings)):
        sub_list.append(f"TOLOWER(a.name) CONTAINS TOLOWER($name{i})")

    where_clause += " AND ".join(sub_list)
    query = DataAccess._find_author_template.replace("$where_clause", where_clause)
    result = self.driver.execute_query(query,parameters_=sub_dict)
    authors = [r.value() for r in result[0]]
    # print(authors)
    return authors
```

2. find_author – враћа све ауторе чије је име слично са датим

```
def find_colleagues(self, author):
    result = self.driver.execute_query("""
        MATCH (:Author {name: $author})-[:WORKED_WITH]-(n:Author)
        RETURN n.name
    """, author=author, database_= "neo4j")
    colleagues = [r.value() for r in result[0]]
    # print(colleagues)
    return colleagues
```

3. find_colleagues – враћа све ауторе који су сарађивали са датим аутором

Приликом покретања програма успоставља се веза са базом, и објекат *DataAccess* класе је доступан у читавом програму.

8 DataAccess.py - https://github.com/momcilol/Erdos-number/blob/main/DataAccess.py

# Алгоритми за претрагу

## Breadth-first search (BFS) – Претрага у ширину

Претраживање у ширину је врста неинформисаног претраживања. Идеја овог претраживања је да се оно врши по нивоима графа. Најпре се креће од задатог чвора и сви суседи чвора се додају у ред за претрагу (*FIFO*). Затим се узима следећи чвор из реда за претрагу и у ред се додају сви његови суседи који нису обиђени или нису већ у реду. Поступак се понавља све док се не пронађе тражени чвор, или док се не испразни ред опслуживања. Просторна и временска сложеност овог алгоритма је $O(b^d)$, где је $b$ фактор гранања а $d$ дубина претраге.

У овом пројекту, услед потребе да се као резултат претраге добију све могуће путање (не само једна), претрага у ширину (метода *find_paths_bfs*) је имплементирана мало другачије.

```
def find_paths_bfs(author: str):
    if author == erdos_name:
        return [[erdos_name]]

    # Mapa je strukture {<element>: {"level": <broj>, "parents": [<parent1>, ...]}}
    path_map = {}
    path_map[author] = {"level": 0, "parents": []}
    queue = []
    queue.append([author])
    queue.append([])
    i = 0
    j = 0
    path_exists = False
```

Најпре проверавамо тривијални случај, тј. да ли је наш аутор управо Пол Ердош. Затим, иницијализујемо потребне променљиве. Ред претраживања *queue* је овде дефинисан по нивоима, тј. први ниво садржи само почетног аутора, у другом нивоу се налазе сви његови суседи, претпоследњи ниво је онај кроз који се итерира и том приликом се последњи ниво пуни суседима чвора који је тренутно посећен. Како бисмо касније могли да реконструишемо путеве, чувамо и посебну мапу *path_map* која чувао податке о томе из којих чворова (*“parents”*) се деспело у неки одређени чвор, као и који ниво (*“level”*) је тај чвор.

```
while True:

    colleagues = db.find_colleagues(queue[i][j])
    if not path_exists:
        for coll in colleagues:
            if coll == erdos_name:
                path_map[erdos_name] = {"level": i+1, "parents": [queue[i][j]]}
                path_exists = True
                break

            # Check if coll is in that level in queue or above (smaller index),
            # making sure we don't go backward
            if coll not in path_map:
                queue[i+1].append(coll)
                path_map[coll] = {"level": i+1, "parents": [queue[i][j]]}
            elif path_map[coll]["level"] == i+1:
                path_map[coll]["parents"].append(queue[i][j])
    else:
        for coll in colleagues:
            if coll == erdos_name:
                path_map[erdos_name]["parents"].append(queue[i][j])
                break

    j += 1

    if j >= len(queue[i]):
        j = 0
        i += 1
        queue.append([])
        if path_exists or len(queue[i]) == 0:
            break
```

Када смо иницијализовали променљиве крећемо у претрагу.

Узимамо први чвор (нека га овде назовемо *parent*, у коду је то *queue[i][j]*) који је на реду и проналазимо његове сараднике (*find_colleagues*). За сваког сарадника *coll* најпре проверавамо да ли је он Пол Ердош. Уколико није настављамо даље и проверавамо да ли се налази у *path_map*. Уколико се *coll* не налази у *path_map* знамо да је то први пут да је чвор посећен у овој претрази, те га додајемо у последњи ниво у *queue* и у *path_map*. Уколико се *coll* налази у *path_map* знамо да смо га већ обишли. Тада проверавамо да ли је он последњи ниво[9] (ниво који се пуни) и ако јесте додамо чвор parent у листу *"parents"* и настављамо даље са претрагом. Уколико *coll* на који смо наишли је уствари Пол Ердош одмах га додајемо у *path_map*, постављамо *path_exists* на *True* и прекидамо петљу јер знамо да сви наредни суседи за наш parent чвор нису Пол Ердош. За све касније чворове које будемо обилазили гледаћемо само *coll* чворове који су Пол Ердош, јер су нам сви остали непотребни. На крају повећавамо бројач j и проверавамо услове петље. Уколико смо прошли све чворове у текућем нивоу *queue* додајемо нови празан ниво и прелазимо на следећи. Уколико је следећи ниво кроз који треба да итерирамо празан или смо пронашли пут до Пола Ердоша (*path_exists == True*) излазимо из петље.

9 "level" у path_map променљивој је користан јер да њега нема морали бисмо да пролазимо кроз цео последњи ниво, док је проналажење елемента у речнику

```
if not path_exists:
    return []


paths = dfs_paths(erdos_name, author, path_map)
path_map = {}
return paths
```

Уколико нисмо нашли пут, вратићемо празну листу, а уколико јесмо покрећемо претраживање у дубину (*dfs_paths*) који пролази кроз *path_map* и реконструише све путање од нашег траженог аутора до Пола Ердоша.

```
def dfs_paths(curr: str, author: str, map: dict):
    if curr == author:
        return [[author]]

    parents = map[curr]["parents"]

    paths = []
    for v in parents:
        paths.extend([a + [curr] for a in dfs_paths(v, author, map)])

    return paths
```

## Iterative Deepening Depth First Search (IDDFS) – Претрага у дубину итеративним продубљивањем

Претраживање у дубину итеративним продубљивањем (*IDDFS*) је комбинација претраживања у ширину (*BFS*) и претраживања у ширину (*DFS*). Оно функционише као *DFS*, али са ограниченом дубином при чему се дубина повећава кроз сваку итерацију. На тај начин оно има временску сложеност као *BFS* а просторну сложеност као *DFS*.

```
def find_paths_ids(author: str):
    i = 0
    while True:
        result = limited_dfs(author, i, [author])
        if result == None:
            return [[]]
        if result != [[]]:
            return result
        i += 1
```

Метод *find_paths_ids* покреће методу *limited_dfs* и повећава дубину све док не пронађе Пола Ердоша, или док не буде имао куда више да тражи.

```python
def limited_dfs(author: str, i: int, visited):
    if i == 0:
        if author == erdos_name:
            return [[erdos_name]]
        else:
            return [[]]

    paths = []
    colleagues = db.find_colleagues(author)
    to_visit = [c for c in colleagues if c not in visited]
    if to_visit == []:
        return None
    else:
        empty_list = False
        full_list = False
        for coll in to_visit:
            subpaths = limited_dfs(coll, i-1, visited + [coll])
            if subpaths == None:
                continue
            if subpaths == [[]]:
                empty_list = True
                continue

            paths.extend([[author] + sub for sub in subpaths])
            full_list = True

    if full_list:
        return paths

    if empty_list:
        return [[]]

    return None
```

Метод *limited_dfs* најпре проверава тривијални случај, када је *i == 0*. Затим, за све суседе текућег чвора (*author*) који нису посећени (нису у *visited*), ако их има, покреће *limited_dfs* и иде један степен ниже у рекурзију. Након повратка рекурзије проверавамо резултат (*subpaths*). Уколико је резултат *None* знамо да та подграна рекурзије наишла на “слепу улицу”, тј. прошли смо све слободне чворове до којих се могло доћи тим путем пре него што смо достигли жељену дубину претраге. Уколико смо добили празну листу, онда знамо да идаље смо постигли жељену дубину претраге, али у тој подграни стабла претраживања нисмо наишли на Пола Ердоша. Даље, уколико *subpaths* није празна листа, то значи да смо нашли пут (или више њих) до Ердоша и то додајемо у *paths*. На послетку враћамо одговарајући резултат претраге.

## Bidirectional Breadth-first Search (BBFS) – Двосмерна претрага у ширину

Двосмерна претрага у ширину (*BBFS*) је побољшана *BFS*-а. У овом случају покрећемо две претраге у ширину: једну из почетног чвора, а другу из траженог чвора. Идеја је да се,

уколико постоји пут између ова два чвора, ове две претраге сусретну на пола пута. На тај начин добијамо алгоритам чија је просторна и временска сложеност $O(b^{d/2})$.

```
def find_paths_bbfs(author: str):
    if author == erdos_name:
        return [[erdos_name]]

    # path_map_bbfs sadrzi dve mape: 0 - polazi od authora, 1 - polazi od erdosa
    # Mapa je strukture {<element>: {"level": <broj>, "parents": [<parent1>, ...]}}
    path_map_bbfs = ({},{})
    path_map_bbfs[0][author] = {"level": 0, "parents": []}
    path_map_bbfs[1][erdos_name] = {"level": 0, "parents": []}

    # Isto vazi i za queue
    queue = ([[author],[]],[[erdos_name],[]])

    path_exists = False

    k = 0
    i = 0
```

Метод *find_paths_bbfs* имплементира *BBFS* алгоритам. Он функционише слично као *find_paths_bfs* али су неки делови адаптирани за претрагу у оба смера. *path_maps_bbfs* је пар мапа идентичних са *path_map* из метода *find_paths_bfs*, где прва служи за смер претраге из траженог аутора, а друга из Пола Ердоша. Аналогно је дефинисан и queue.

Даља претрага такође функционише слично. Наизменично итерирамо кроз претпоследње ниве *queue*-а и за сваки чвор (*queue[k][i][j]*) итерирамо кроз његове суседе (*coll*). За разлику од класичног *BFS*-а где смо проверавали да ли је *coll* Пол Ердош, сада морамо да проверавамо да ли се *coll* налази у мапи за претрагу из супротног смера јер ако се налази, онда то значи да смо пронашли пут, и да ћемо у даљем итерирању кроз *queue* само гледати оне који се налазе у *path_map_bbfs* за супротни смер. Такође, пошто овде не постоји један кључан чвор за престанак претраге, већ може бити низ различитих чворова, ми не можемо одмах прекинути итерацију кроз *colleagues* и прећи на следећи чвор из *queue*-а, као што је то случај код обичног *BFS*-а, него морамо да наставимо даље јер постоји могућност да нађемо још неке путеве. Претрага се врши све док се не пронађе пут, или док не понестане чворова за даљу претрагу.

```
while True:
    for j in range(len(queue[k][i])):
        colleagues = db.find_colleagues(queue[k][i][j])
        if not path_exists:
            for coll in colleagues:
                if not path_exists:
                    if coll in path_map_bbfs[(k+1)%2]:
                        path_map_bbfs[k][coll] = {"level": i+1, "parents": [queue[k][i][j]]}
                        path_exists = True
                        continue

                    #Check if coll is in that level in queue or above (smaller index),
                    # making sure we don't go backward
                    if coll not in path_map_bbfs[k]:
                        queue[k][i+1].append(coll)
                        path_map_bbfs[k][coll] = {"level": i+1, "parents": [queue[k][i][j]]}
                    elif path_map_bbfs[k][coll]["level"] == i+1:
                        path_map_bbfs[k][coll]["parents"].append(queue[k][i][j])
                else:
                    if coll in path_map_bbfs[(k+1)%2]:
                        if coll not in path_map_bbfs[k]:
                            path_map_bbfs[k][coll] = {"level": i+1, "parents": [queue[k][i][j]]}
                        else:
                            path_map_bbfs[k][coll]["parents"].append(queue[k][i][j])
        else:
            for coll in colleagues:
                if coll in path_map_bbfs[(k+1)%2]:
                    if coll not in path_map_bbfs[k]:
                        path_map_bbfs[k][coll] = {"level": i+1, "parents": [queue[k][i][j]]}
                    else:
                        path_map_bbfs[k][coll]["parents"].append(queue[k][i][j])

    if path_exists or len(queue[k][i+1]) == 0:
        break

    queue[k].append([])
    i += (k+1) // 2
    k = (k+1) % 2
```

На крају, уколико пут постоји, тражимо пресек ове две мапе претраге. За сваки елемент из пресека пуштамо *dfs_paths* у оба смера и комбинујемо сваки подпут из једног смера (*path_author*) са сваким подпутем из другог смера (*path_erdos*), додајемо их у листу свих путева и враћамо резултат.

```
if not path_exists:
    return [[]]

paths = []
intersection = [key for key in path_map_bbfs[0].keys() & path_map_bbfs[1].keys()]

for k in intersection:
    paths_author = [path[:-1] for path in dfs_paths(k,author,path_map_bbfs[0])]
    paths_erdos = [path[-2::-1] for path in dfs_paths(k, erdos_name, path_map_bbfs[1])]

    for pa in paths_author:
        for pe in paths_erdos:
            paths.append(pa + [k] + pe)

return paths
```

## Кориснички интерфејс

Програм је осмишљен тако да има текстуални кориснички интерфејс. Приликом покретања програма исписује се порука која описује рад програма.

```
WELCOME TO "PATH TO ERDÖS" FINDER!!!

If u want to search authors insert text like this:
>> -a Milos Radovanovic
and as a response you will get all authors that have both "Milos" and "Radovanovic" in their name.

If you want to see all authors that have worked with some author, insert text like this:
>> -c Milos Radovanovic
and as a response you will get all authors that "Milos Radovanovic" have worked with.

If you want to see "PATH TO ERDÖS" of some author, you have three choises:
1. Using Breadth First Search (BFS):
>> -bfs Milos Radovanovic
2. Using Iterative Deepening Search(IDS):
>> -ids Milos Radovanovic
3. Using Bidirectional Breadth First Search (BBFS)
>> -bbfs Milos Radovanovic
and as a response you will get shortest path (or multiple, if there is more than one) to the legendary mathematician "Paul Erdös"!

For exit type:
>> exit

To print again this welcome message type:
>> welcome
```

Прва функционалност нам омогућава проналажење аутора. Уносом кључне речи за функционалност (“-a”) и низ речи, програм позива метод *find_author* из *DataAccess* класе и враћа све оне ауторе који у свом имену садрже све наведене речи.

```
if tokens[0] == "-a":
    authors = db.find_author(tokens[1])
    if len(authors) == 0:
        print("There is no author with that or similar name...")
        continue

    print("I found those authors: ")
    for a in authors:
        print(a)

    continue

if tokens[0] == "-c":
    if not db.check_author(tokens[1]):
        print("There is no author with that name...")
        continue

    colleagues = db.find_colleagues(tokens[1])
    if len(colleagues) == 0:
        print(f"There are no colleagues for {tokens[1]}...")
        continue

    print("I found those colleagues: ")
    for c in colleagues:
        print(c)

    continue
```

Друга функционалност нам омогућава проналажење свих аутора са којима је наш тражени аутор сарађивао. Уносом кључне речи “-c” и назив аутора, програм позива методу *find_colleagues* из *DataAccess* класе и враћа се све оне ауторе који су сарађивали са аутором који има име идентично наведеном.

```
if tokens[0] == "-bfs":
    search(tokens[1], find_paths_bfs)
    continue

if tokens[0] == "-ids":
    search(tokens[1], find_paths_ids)
    continue

if tokens[0] == "-bbfs":
    search(tokens[1], find_paths_bbfs)
    continue
```

Трећа функционалност јесте сама претрага, односно приказивање свих путева од задатог аутора до Пола Ердоша. Овде имамо три опције:

1. “-bfs” – Breadth-first search (BFS) – Претрага у ширину
2. “-ids” – Iterative Deepening Depth First Search (IDDFS) – Претрага у дубину итеративним продубљивањем
3. “-bbfs” – Bidirectional Breadth-first Search (BBFS) – Двосмерна претрага у ширину

```
def search(author: str, fun: FunctionType):
    if not db.check_author(author):
        print("There is no author with that name...")
        return

    start_time = time()
    paths = fun(author)
    print(f"Execution time: {time() - start_time}")
    if len(paths) == 0:
        print(f"There are no paths from {author} to {erdos_name}")
        return

    print(f"Erdös number of {author} is {len(paths[0])-1}")
    print("Paths:")
    for i in range(len(paths)):
        print(f"{i+1}."," - ".join(paths[i]))
```

Свака од опција позива метод *search* и прослеђује одговарајући метод за претрагу као параметар. У методу најпре проверавамо да ли дати аутор постоји, а затим покрећемо прослеђени метод. Након завршетка исписујемо време извршавања, Ердошев број и све путеве од датог аутора до Пола Ердоша (уколико их има).

# Примери извршавања програма

Показаћемо рад програма на примерима три аутора: Душице Кнежевић, Мирјане Микалачки и Милоша Рацковића. У случају Милоша Рацковића, приказан је само део путева (укупно има 99).

### *1. Претраживање аутора*

```
>> -a Dusica k
I found those authors:
Dusica Vujaklija
Dusica Novakovic
Dusica Semencenko
Dusica Knezevic
Dusica Pleterski Rigler
```

```
>> -a Mirjana K L
I found those authors:
Mirjana Mikalacki
Mirjana Kljajic Borstnar
Mirjana Vukelja
Mirjana Kocaleva
Mirjana Babic Leko
Mirjana D. Radosavljevic-Nikolic
Mirjana Maljkovic
```

```
>> -a Rackovic
I found those authors:
Milos Rackovic
Stevo Rackovic
Kresimir Perackovic
```

### *2. Листање коаутора*

```
>> -c Dusica Knezevic
I found those colleagues:
Milos Radovanovic
Jela Babic
Milos Savic 0001
```

```
>> -c Mirjana Mikalacki
I found those colleagues:
Zoltán Lóránt Nagy
Fiona Skerman
Alon Naor
Andrzej Grzesik
Balázs Patkós
Mihyun Kang
Chris Dowden
Pranshu Gupta
Yannick Mogge
Dan Hefetz
Alexander Haupt
Fabian Hamann
Jovana Forcan
Dennis Clemens
Milos Stojakovic
```

```
>> -c Milos Rackovic
I found those colleagues:
Jovana Vidakovic
Srdjan Skrbic
Sasa Kresoja
Bojana Dimic Surla
Zdravko Ivankovic
Miodrag Ivkovic
Aleksandar Takaci
Dragan Ivanovic
Predrag Pecev
Dusan Surla
Miomir Vukobratovic
Srdan Skrbic
Goran Panic
Nemanja Milosevic
```

### *3. Пут до Ердоша – BFS*

```
>> -bfs Dusica Knezevic
Execution time: 27.105910539627075
Erdös number of Dusica Knezevic is 4
Paths:
1. Dusica Knezevic - Milos Radovanovic - Michael E. Houle - David Avis - Paul Erdös
2. Dusica Knezevic - Milos Radovanovic - Ken-ichi Kawarabayashi - Guantao Chen - Paul Erdös
3. Dusica Knezevic - Milos Radovanovic - Ken-ichi Kawarabayashi - Ronald J. Gould - Paul Erdös
4. Dusica Knezevic - Milos Radovanovic - Ken-ichi Kawarabayashi - Carsten Thomassen - Paul Erdös
5. Dusica Knezevic - Milos Radovanovic - Ken-ichi Kawarabayashi - Alexandr V. Kostochka - Paul Erdös
```

```
>> -bfs Mirjana Mikalacki
Execution time: 2.2887330055236816
Erdös number of Mirjana Mikalacki is 3
Paths:
1. Mirjana Mikalacki - Alon Naor - Ron Aharoni - Paul Erdös
2. Mirjana Mikalacki - Dan Hefetz - Ron Aharoni - Paul Erdös
3. Mirjana Mikalacki - Alon Naor - Noga Alon - Paul Erdös
4. Mirjana Mikalacki - Mihyun Kang - Noga Alon - Paul Erdös
5. Mirjana Mikalacki - Dan Hefetz - Noga Alon - Paul Erdös
6. Mirjana Mikalacki - Alon Naor - Nicholas C. Wormald - Paul Erdös
7. Mirjana Mikalacki - Alon Naor - Michael Krivelevich - Paul Erdös
8. Mirjana Mikalacki - Balázs Patkós - Michael Krivelevich - Paul Erdös
9. Mirjana Mikalacki - Mihyun Kang - Michael Krivelevich - Paul Erdös
10. Mirjana Mikalacki - Chris Dowden - Michael Krivelevich - Paul Erdös
11. Mirjana Mikalacki - Dan Hefetz - Michael Krivelevich - Paul Erdös
12. Mirjana Mikalacki - Dennis Clemens - Michael Krivelevich - Paul Erdös
13. Mirjana Mikalacki - Milos Stojakovic - Michael Krivelevich - Paul Erdös
14. Mirjana Mikalacki - Balázs Patkós - Zoltán Füredi - Paul Erdös
15. Mirjana Mikalacki - Balázs Patkós - Peter Frankl - Paul Erdös
16. Mirjana Mikalacki - Balázs Patkós - András Gyárfás - Paul Erdös
17. Mirjana Mikalacki - Balázs Patkós - Zsolt Tuza - Paul Erdös
18. Mirjana Mikalacki - Mihyun Kang - Béla Bollobás - Paul Erdös
19. Mirjana Mikalacki - Mihyun Kang - Vojtech Rödl - Paul Erdös
20. Mirjana Mikalacki - Mihyun Kang - Tomasz Luczak 0001 - Paul Erdös
21. Mirjana Mikalacki - Dan Hefetz - Tomasz Luczak 0001 - Paul Erdös
22. Mirjana Mikalacki - Mihyun Kang - Joel Spencer - Paul Erdös
23. Mirjana Mikalacki - Dennis Clemens - Yoshiharu Kohayakawa - Paul Erdös


>> -bfs Milos Rackovic
Execution time: 2393.7166125774384
Erdös number of Milos Rackovic is 5
Paths:
1. Milos Rackovic - Srdjan Skrbic - Nicolas Kourtellis - David García-Soriano - Noga Alon - Paul Erdös
2. Milos Rackovic - Nemanja Milosevic - Nicolas Kourtellis - David García-Soriano - Noga Alon - Paul Erdös
3. Milos Rackovic - Srdjan Skrbic - Ioannis Arapakis - David García-Soriano - Noga Alon - Paul Erdös
4. Milos Rackovic - Nemanja Milosevic - Ioannis Arapakis - David García-Soriano - Noga Alon - Paul Erdös
5. Milos Rackovic - Srdjan Skrbic - Nicolas Kourtellis - Rubén Cuevas Rumín - Shmuel Zaks - Paul Erdös
6. Milos Rackovic - Nemanja Milosevic - Nicolas Kourtellis - Rubén Cuevas Rumín - Shmuel Zaks - Paul Erdös
7. Milos Rackovic - Srdjan Skrbic - Julien-Etienne Mascolo - Vincenzo Sciancalepore - Shmuel Zaks - Paul Erdös
8. Milos Rackovic - Srdan Skrbic - Julien-Etienne Mascolo - Vincenzo Sciancalepore - Shmuel Zaks - Paul Erdös
9. Milos Rackovic - Nemanja Milosevic - Julien-Etienne Mascolo - Vincenzo Sciancalepore - Shmuel Zaks - Paul Erdös
10. Milos Rackovic - Srdjan Skrbic - Milan Lukic - Ivan Stojmenovic - Pavel Valtr 0001 - Paul Erdös
11. Milos Rackovic - Srdan Skrbic - Milan Lukic - Ivan Stojmenovic - Pavel Valtr 0001 - Paul Erdös
12. Milos Rackovic - Nemanja Milosevic - Milan Lukic - Ivan Stojmenovic - Pavel Valtr 0001 - Paul Erdös
13. Milos Rackovic - Srdjan Skrbic - Ivan Mezei - Ivan Stojmenovic - Pavel Valtr 0001 - Paul Erdös
14. Milos Rackovic - Srdan Skrbic - Ivan Mezei - Ivan Stojmenovic - Pavel Valtr 0001 - Paul Erdös
15. Milos Rackovic - Nemanja Milosevic - Ivan Mezei - Ivan Stojmenovic - Pavel Valtr 0001 - Paul Erdös
16. Milos Rackovic - Srdjan Skrbic - Dejan Vukobratovic - Milos Stojakovic - Michael Krivelevich - Paul Erdös
17. Milos Rackovic - Srdan Skrbic - Dejan Vukobratovic - Milos Stojakovic - Michael Krivelevich - Paul Erdös
18. Milos Rackovic - Nemanja Milosevic - Dejan Vukobratovic - Milos Stojakovic - Michael Krivelevich - Paul Erdös
19. Milos Rackovic - Srdjan Skrbic - Lovro Subelj - Tilen Marc - Zsolt Tuza - Paul Erdös
20. Milos Rackovic - Srdjan Skrbic - Taha Yasseri - András Kornai - Zsolt Tuza - Paul Erdös
21. Milos Rackovic - Dragan Ivanovic - Zsolt Németh - Gábor Bacsó - Zsolt Tuza - Paul Erdös
22. Milos Rackovic - Nemanja Milosevic - Qi Zhang - Zhiyi Tan 0001 - Zsolt Tuza - Paul Erdös
23. Milos Rackovic - Srdjan Skrbic - Sotiris Ioannidis - Ian F. Akyildiz - D. Frank Hsu - Paul Erdös
24. Milos Rackovic - Nemanja Milosevic - Sotiris Ioannidis - Ian F. Akyildiz - D. Frank Hsu - Paul Erdös
25. Milos Rackovic - Srdjan Skrbic - Ilias Spais - Ian F. Akyildiz - D. Frank Hsu - Paul Erdös
26. Milos Rackovic - Nemanja Milosevic - Ilias Spais - Ian F. Akyildiz - D. Frank Hsu - Paul Erdös
27. Milos Rackovic - Miomir Vukobratovic - Toshio Fukuda - Lotfi A. Zadeh - D. Frank Hsu - Paul Erdös
```

## 4. Пут до Ердоша – IDS

```
>> -ids Dusica Knezevic
Execution time: 43.06368279457092
Erdös number of Dusica Knezevic is 4
Paths:
1. Dusica Knezevic - Milos Radovanovic - Michael E. Houle - David Avis - Paul Erdös
2. Dusica Knezevic - Milos Radovanovic - Ken-ichi Kawarabayashi - Guantao Chen - Paul Erdös
3. Dusica Knezevic - Milos Radovanovic - Ken-ichi Kawarabayashi - Ronald J. Gould - Paul Erdös
4. Dusica Knezevic - Milos Radovanovic - Ken-ichi Kawarabayashi - Carsten Thomassen - Paul Erdös
5. Dusica Knezevic - Milos Radovanovic - Ken-ichi Kawarabayashi - Alexandr V. Kostochka - Paul Erdös


>> -ids Mirjana Mikalacki
Execution time: 3.2595837116241455
Erdös number of Mirjana Mikalacki is 3
Paths:
1. Mirjana Mikalacki - Alon Naor - Ron Aharoni - Paul Erdös
2. Mirjana Mikalacki - Alon Naor - Noga Alon - Paul Erdös
3. Mirjana Mikalacki - Alon Naor - Nicholas C. Wormald - Paul Erdös
4. Mirjana Mikalacki - Alon Naor - Michael Krivelevich - Paul Erdös
5. Mirjana Mikalacki - Balázs Patkós - Michael Krivelevich - Paul Erdös
6. Mirjana Mikalacki - Balázs Patkós - Zoltán Füredi - Paul Erdös
7. Mirjana Mikalacki - Balázs Patkós - Peter Frankl - Paul Erdös
8. Mirjana Mikalacki - Balázs Patkós - András Gyárfás - Paul Erdös
9. Mirjana Mikalacki - Balázs Patkós - Zsolt Tuza - Paul Erdös
10. Mirjana Mikalacki - Mihyun Kang - Noga Alon - Paul Erdös
11. Mirjana Mikalacki - Mihyun Kang - Béla Bollobás - Paul Erdös
12. Mirjana Mikalacki - Mihyun Kang - Vojtech Rödl - Paul Erdös
13. Mirjana Mikalacki - Mihyun Kang - Tomasz Luczak 0001 - Paul Erdös
14. Mirjana Mikalacki - Mihyun Kang - Joel Spencer - Paul Erdös
15. Mirjana Mikalacki - Mihyun Kang - Michael Krivelevich - Paul Erdös
16. Mirjana Mikalacki - Chris Dowden - Michael Krivelevich - Paul Erdös
17. Mirjana Mikalacki - Dan Hefetz - Ron Aharoni - Paul Erdös
18. Mirjana Mikalacki - Dan Hefetz - Michael Krivelevich - Paul Erdös
19. Mirjana Mikalacki - Dan Hefetz - Noga Alon - Paul Erdös
20. Mirjana Mikalacki - Dan Hefetz - Tomasz Luczak 0001 - Paul Erdös
21. Mirjana Mikalacki - Dennis Clemens - Yoshiharu Kohayakawa - Paul Erdös
22. Mirjana Mikalacki - Dennis Clemens - Michael Krivelevich - Paul Erdös
23. Mirjana Mikalacki - Milos Stojakovic - Michael Krivelevich - Paul Erdös
```

```
>> -ids Milos Rackovic
Execution time: 20148.162115335464
Erdös number of Milos Rackovic is 5
Paths:
1. Milos Rackovic - Srdjan Skrbic - Nicolas Kourtellis - David García-Soriano - Noga Alon - Paul Erdös
2. Milos Rackovic - Srdjan Skrbic - Nicolas Kourtellis - Rubén Cuevas Rumín - Shmuel Zaks - Paul Erdös
3. Milos Rackovic - Srdjan Skrbic - Ioannis Arapakis - David García-Soriano - Noga Alon - Paul Erdös
4. Milos Rackovic - Srdjan Skrbic - Julien-Etienne Mascolo - Vincenzo Sciancalepore - Shmuel Zaks - Paul Erdös
5. Milos Rackovic - Srdjan Skrbic - Milan Lukic - Ivan Stojmenovic - Pavel Valtr 0001 - Paul Erdös
6. Milos Rackovic - Srdjan Skrbic - Ivan Mezei - Ivan Stojmenovic - Pavel Valtr 0001 - Paul Erdös
7. Milos Rackovic - Srdjan Skrbic - Dejan Vukobratovic - Milos Stojakovic - Michael Krivelevich - Paul Erdös
8. Milos Rackovic - Srdjan Skrbic - Lovro Subelj - Tilen Marc - Zsolt Tuza - Paul Erdös
9. Milos Rackovic - Srdjan Skrbic - Taha Yasseri - András Kornai - Zsolt Tuza - Paul Erdös
10. Milos Rackovic - Srdjan Skrbic - Sotiris Ioannidis - Ian F. Akyildiz - D. Frank Hsu - Paul Erdös
11. Milos Rackovic - Srdjan Skrbic - Rizos Sakellariou - Ping Hu - Miklós Simonovits - Paul Erdös
12. Milos Rackovic - Srdjan Skrbic - Rizos Sakellariou - Sushil K. Prasad - Guantao Chen - Paul Erdös
13. Milos Rackovic - Srdjan Skrbic - Rizos Sakellariou - Daniel S. Katz - Israel Koren - Paul Erdös
14. Milos Rackovic - Srdjan Skrbic - Ilias Spais - Ian F. Akyildiz - D. Frank Hsu - Paul Erdös
15. Milos Rackovic - Bojana Dimic Surla - Milos Radovanovic - Michael E. Houle - David Avis - Paul Erdös
16. Milos Rackovic - Bojana Dimic Surla - Milos Radovanovic - Ken-ichi Kawarabayashi - Guantao Chen - Paul Erdös
17. Milos Rackovic - Bojana Dimic Surla - Milos Radovanovic - Ken-ichi Kawarabayashi - Ronald J. Gould - Paul Erdös
18. Milos Rackovic - Bojana Dimic Surla - Milos Radovanovic - Ken-ichi Kawarabayashi - Carsten Thomassen - Paul Erdös
19. Milos Rackovic - Bojana Dimic Surla - Milos Radovanovic - Ken-ichi Kawarabayashi - Alexandr V. Kostochka - Paul Erdös
20. Milos Rackovic - Aleksandar Takaci - Endre Pap - Radko Mesiar - Jozef Sirán - Paul Erdös
21. Milos Rackovic - Aleksandar Takaci - Miroslav Maric - Panos M. Pardalos - Ronald L. Graham - Paul Erdös
22. Milos Rackovic - Dragan Ivanovic - Philipp Leitner 0001 - Yaron Wolfsthal - Shlomo Moran - Paul Erdös
23. Milos Rackovic - Dragan Ivanovic - Philipp Leitner 0001 - Artur Andrzejak 0001 - Boris Aronov - Paul Erdös
24. Milos Rackovic - Dragan Ivanovic - Andreas Metzger - Yaron Wolfsthal - Shlomo Moran - Paul Erdös
25. Milos Rackovic - Dragan Ivanovic - Georgia M. Kapitsaki - Nikolay Nikolov - László Babai - Paul Erdös
26. Milos Rackovic - Dragan Ivanovic - Georgia M. Kapitsaki - Nikolay Nikolov - László Pyber - Paul Erdös
27. Milos Rackovic - Dragan Ivanovic - Zsolt Németh - Gábor Bacsó - András Gyárfás - Paul Erdös
```

## 5. Пут до Ердоша – BBFS

```
>> -bbfs Dusica Knezevic
Execution time: 0.7046539783477783
Erdös number of Dusica Knezevic is 4
Paths:
1. Dusica Knezevic - Milos Radovanovic - Ken-ichi Kawarabayashi - Carsten Thomassen - Paul Erdös
2. Dusica Knezevic - Milos Radovanovic - Ken-ichi Kawarabayashi - Alexandr V. Kostochka - Paul Erdös
3. Dusica Knezevic - Milos Radovanovic - Ken-ichi Kawarabayashi - Ronald J. Gould - Paul Erdös
4. Dusica Knezevic - Milos Radovanovic - Ken-ichi Kawarabayashi - Guantao Chen - Paul Erdös
5. Dusica Knezevic - Milos Radovanovic - Michael E. Houle - David Avis - Paul Erdös
```

```
>> -bbfs Mirjana Mikalacki
Execution time: 0.12143993377685547
Erdös number of Mirjana Mikalacki is 3
Paths:
1. Mirjana Mikalacki - Balázs Patkós - András Gyárfás - Paul Erdös
2. Mirjana Mikalacki - Alon Naor - Noga Alon - Paul Erdös
3. Mirjana Mikalacki - Mihyun Kang - Noga Alon - Paul Erdös
4. Mirjana Mikalacki - Dan Hefetz - Noga Alon - Paul Erdös
5. Mirjana Mikalacki - Balázs Patkós - Zoltán Füredi - Paul Erdös
6. Mirjana Mikalacki - Balázs Patkós - Peter Frankl - Paul Erdös
7. Mirjana Mikalacki - Balázs Patkós - Zsolt Tuza - Paul Erdös
8. Mirjana Mikalacki - Alon Naor - Michael Krivelevich - Paul Erdös
9. Mirjana Mikalacki - Balázs Patkós - Michael Krivelevich - Paul Erdös
10. Mirjana Mikalacki - Mihyun Kang - Michael Krivelevich - Paul Erdös
11. Mirjana Mikalacki - Chris Dowden - Michael Krivelevich - Paul Erdös
12. Mirjana Mikalacki - Dan Hefetz - Michael Krivelevich - Paul Erdös
13. Mirjana Mikalacki - Dennis Clemens - Michael Krivelevich - Paul Erdös
14. Mirjana Mikalacki - Milos Stojakovic - Michael Krivelevich - Paul Erdös
15. Mirjana Mikalacki - Mihyun Kang - Vojtech Rödl - Paul Erdös
16. Mirjana Mikalacki - Alon Naor - Ron Aharoni - Paul Erdös
17. Mirjana Mikalacki - Dan Hefetz - Ron Aharoni - Paul Erdös
18. Mirjana Mikalacki - Dennis Clemens - Yoshiharu Kohayakawa - Paul Erdös
19. Mirjana Mikalacki - Alon Naor - Nicholas C. Wormald - Paul Erdös
20. Mirjana Mikalacki - Mihyun Kang - Béla Bollobás - Paul Erdös
21. Mirjana Mikalacki - Mihyun Kang - Joel Spencer - Paul Erdös
22. Mirjana Mikalacki - Mihyun Kang - Tomasz Luczak 0001 - Paul Erdös
23. Mirjana Mikalacki - Dan Hefetz - Tomasz Luczak 0001 - Paul Erdös
```

```
>> -bbfs Milos Rackovic
Execution time: 2.5892109870910645
Erdös number of Milos Rackovic is 5
Paths:
1. Milos Rackovic - Srdjan Skrbic - Rizos Sakellariou - Ping Hu - Miklós Simonovits - Paul Erdös
2. Milos Rackovic - Nemanja Milosevic - Rizos Sakellariou - Ping Hu - Miklós Simonovits - Paul Erdös
3. Milos Rackovic - Srdjan Skrbic - Dejan Vukobratovic - Milos Stojakovic - Michael Krivelevich - Paul Erdös
4. Milos Rackovic - Srdan Skrbic - Dejan Vukobratovic - Milos Stojakovic - Michael Krivelevich - Paul Erdös
5. Milos Rackovic - Nemanja Milosevic - Dejan Vukobratovic - Milos Stojakovic - Michael Krivelevich - Paul Erdös
6. Milos Rackovic - Nemanja Milosevic - Qi Zhang - Zhi Li - D. Frank Hsu - Paul Erdös
7. Milos Rackovic - Nemanja Milosevic - Qi Zhang - Toyoaki Nishida - D. Frank Hsu - Paul Erdös
8. Milos Rackovic - Bojana Dimic Surla - Milos Radovanovic - Michael E. Houle - David Avis - Paul Erdös
9. Milos Rackovic - Srdan Skrbic - Milos Radovanovic - Michael E. Houle - David Avis - Paul Erdös
10. Milos Rackovic - Dragan Ivanovic - Zsolt Németh - Gábor Bacsó - András Gyárfás - Paul Erdös
11. Milos Rackovic - Dragan Ivanovic - Zsolt Németh - Gábor Bacsó - Zsolt Tuza - Paul Erdös
12. Milos Rackovic - Nemanja Milosevic - Soummya Kar - Kavita Ramanan - Prasad Tetali - Paul Erdös
13. Milos Rackovic - Srdjan Skrbic - Julien-Etienne Mascolo - Vincenzo Sciancalepore - Shmuel Zaks - Paul Erdös
14. Milos Rackovic - Srdan Skrbic - Julien-Etienne Mascolo - Vincenzo Sciancalepore - Shmuel Zaks - Paul Erdös
15. Milos Rackovic - Nemanja Milosevic - Julien-Etienne Mascolo - Vincenzo Sciancalepore - Shmuel Zaks - Paul Erdös
16. Milos Rackovic - Srdjan Skrbic - Sotiris Ioannidis - Ian F. Akyildiz - D. Frank Hsu - Paul Erdös
17. Milos Rackovic - Nemanja Milosevic - Sotiris Ioannidis - Ian F. Akyildiz - D. Frank Hsu - Paul Erdös
18. Milos Rackovic - Srdjan Skrbic - Ilias Spais - Ian F. Akyildiz - D. Frank Hsu - Paul Erdös
19. Milos Rackovic - Nemanja Milosevic - Ilias Spais - Ian F. Akyildiz - D. Frank Hsu - Paul Erdös
20. Milos Rackovic - Nemanja Milosevic - Qi Zhang - Weihua Gui 0001 - Lou Caccetta - Paul Erdös
21. Milos Rackovic - Nemanja Milosevic - Qi Zhang - Limin Xiao - Peter Horák - Paul Erdös
22. Milos Rackovic - Srdjan Skrbic - Lovro Subelj - Tilen Marc - Zsolt Tuza - Paul Erdös
23. Milos Rackovic - Miomir Vukobratovic - Hugh M. Herr - Jing Wang - D. Frank Hsu - Paul Erdös
24. Milos Rackovic - Nemanja Milosevic - Qi Zhang - Jing Wang - D. Frank Hsu - Paul Erdös
25. Milos Rackovic - Srdjan Skrbic - Nicolas Kourtellis - David García-Soriano - Noga Alon - Paul Erdös
26. Milos Rackovic - Nemanja Milosevic - Nicolas Kourtellis - David García-Soriano - Noga Alon - Paul Erdös
27. Milos Rackovic - Srdjan Skrbic - Ioannis Arapakis - David García-Soriano - Noga Alon - Paul Erdös
```

# Закључак

У овом пројекту смо на примеру мреже сарадње аутора имплементирали три алгоритма за проналажење најкраћег пута између два чвора у графу.

| | Ердошев број | BFS | IDS | BBFS |
|---|---|---|---|---|
| Мирјана Микалачки | 3 | 2,29 s | 3,26 s | 0,12 s |
| Душица Кнежевић | 4 | 27,11 s | 43,06 s | 0,70 s |
| Милош Рацковић | 5 | ~ 40 min | ~ 5h 35 min | 2,59 s |

Прва два алгоритма (*BFS* и *IDS*) су једноставнији и лакши за имплементацију. Када су у питању мање раздаљине (4 и мање) време њиховог извршавања је прихватљиво, при чему је *BFS* бржи од *IDS* алгоритма услед понављања претраге кроз сваку итерацију код *IDS*. Како време њиховог извршавања расте експоненцијално са дистанцом (временска сложеност им је $O(b^d)$), за веће бројеве (5 и више) ови алгоритми постају неупотребљиви.

Трећи алгоритам (*BBFS*) је сложенији за имплементацију. Међутим, због мање временске сложености (= $O(b^{d/2})$) он је далеко моћнији алгоритам за претрагу. Та разлика се јасно види код већих дистанци, где коришћење овог алгоритма уместо претходна два није ствар избора него нужде.

Сва три алгоритма за претрагу су примери неинформисаног претраживања где је свака грана у графу исте тежине. Уколико бисмо имали тежину грана или неке особине по којима би се могла направити згодна функција за хеуристку били бисмо у могућности да имплементирамо још неке алгоритме за претрагу, као што су *Uniform Cost Search (UCS)* и *A* Search*.