מגישים :
איתמר בבאי 206847584
שריאל משה 322772880

פרויקט גמר מערכות הפעלה

שאלה 1 :

התבקשנו ליצור מבנה נתונים של גרף, בחרתי להשתמש במטריצת שכנויות.

כאשר יש צלע יופיע 1 וכאשר אין צלע inf .

```
class Graph {
    private:
    int vertices;
    std::vector<std::vector<int>> adjMat; // adjacency matrix representation
    int EdgesNum;

public:
    Graph(int vertices, int edges);
    Graph();
    ~Graph();
    void addEdge(int u, int v);
    void printGraph();
```

שאלה 2 : בשאלה זו התבקשנו לממש אלגוריתם למציאת מעגל אויילר או הוכחה שאינו קיים.

נבצע זאת במספר שלבים :

שלב ראשון : נבחן האם כל דרגות הקודקודים הגרף זוגיות, במידה לא קיים מעגל אויילר (נלמד בדיסקרטיים). במקרה זה נדפיס את הקודקוד שדרגתו אי זוגית.

```
bool Graph::EvenDeg(){
  int counter;
  for(int i=0 ; i<vertices ;i++){
      counter=0;
      for (int j=0; j<vertices ;j++){
          if(adjMat[i][j]==1){
              counter++;
          }
      }
      if(counter%2==1){
          std::cerr << "Graph is not Eulerian: vertex " << i << " has odd degree\n";
          return false;
      }
  }
  return true;
}
```

מגישים :
איתמר בבאי 206847584
שריאל משה 322772880

<u>שלב שני</u> : נבחן האם הגרף קשיר, נעשה זאת באמצעות הפעלת DFS על הגרף. אם הגרף אינו קשיר (בתנאי שיש צלעות בשני רכיבי הקשירות) הוא בפרט לא מכיל מעגל אויילר נדפיס את הקודקוד שהתגלה כלא קשיר.

```
void dfs(int u, const std::vector<std::vector<int>>& adj, std::vector<bool>& visited) {
    visited[u] = true;
    for (size_t v = 0; v < adj.size(); ++v) {
        if (adj[u][v] == 1 && !visited[v])
            dfs(v, adj, visited);
    }
}
```

```
bool isConnected(const std::vector<std::vector<int>>& adj) {
    int n = adj.size();
    std::vector<bool> visited(n, false);
    int start = -1;
    for (int i = 0; i < n; ++i) {
        int degree = 0;
        for (int j = 0; j < n; ++j)
            if (adj[i][j] == 1) degree++;
        if (degree > 0) {
            start = i;
            break;
        }
    }
    if (start == -1)
        return true;

    dfs(start, adj, visited);
    for (int i = 0; i < n; ++i) {
        int degree = 0;
        for (int j = 0; j < n; ++j)
            if (adj[i][j] == 1) degree++;
        if (degree > 0 && !visited[i]){
            std::cerr << "Graph is not connected: vertex " << i << " is not reachable from vertex " << start << "\n";
            return false;
        }
    }
    return true;
}
```

מגישים:
איתמר בבאי 206847584
שריאל משה 322772880

<u>שלב שלישי</u>: אחרי שאנחנו יודעים שקיים נמצא אותו באופן הבא:

- ניצור מטריצת עזר שתהיה העתק של מטריצת השכנויות המקורית.
- יצירת מחסנית שתחזיק את הנוכחי.
- נתחיל מקודקוד מקור כלשהו, נכניס קודקוד שכן אחד שלו למחסנית ונסיר את הצלע בניהם.
- נעבור לקודקוד הבא במחסנית ונכניס קודקוד שכן אחד שלו ונסיר את הצלע בניהם.
- באופן זה נמשיך עבור כל הצלעות עד שנגיע לקודקוד שלא קיימות עבורו צלעות וזה אומר שסיימנו את הקטע ממסלול זה ונכניס אותו לווקטור של אויילר, נמשיך רקורסיבית על התוכן של המחסנית.
- נמשיך כך כל עוד המחסנית מכילה קודקודים.

```cpp
bool Graph::EulerianCycle(std::vector<int>& euler){
    if(!EvenDeg()|| !isConnected(adjMat)){
        return false;
    }
    std::stack<int> path;
    std::vector<std::vector<int>> copyMat=this->adjMat;
    int source=0;
    path.push(source);//start from 0 - positiv and even deg

    while(!path.empty()){
        int u=path.top();
        bool found=false;
        for(int i =0; i<vertices; i++ ){
            if(copyMat[u][i]==1){
                path.push(i);
                found=true;
                copyMat[u][i]=inf;
                copyMat[i][u]=inf;
                break;
            }
        }
       if (!found) {
            euler.push_back(u);
            path.pop();
        }
    }
    std::reverse(euler.begin(), euler.end());
    return true;
```

מגישים :
איתמר בבאי 206847584
שריאל משה 322772880

שאלה 3 :

בשאלה זו התבקשנו ליצור גרף רנדומלי בעזרת דגלים שנקבל מהטרמינל.

נעזרנו בפונקציה getopt_long על מנת לפרסר בצורה נוחה את הקלט המתקבל.

נבצע בדיקות תקינות על הקלט (בדיקה שבאמת הגיע מספר, בדיקה שמספר הצלעות לא גבוה מידי , בדיקה שכל הארגומנטים אותחלו)

נאתחל את השדות של הגרף, נקצה מקום למטריצת השכנויות ונפעיל את הפונקציה היוצרת גרף רנדומלי.

```cpp
void  Graph::parseFlags(int argc, char* argv[]) {
    int e_num=-1,v_num=-1,seed=-1;
    int opt;

    while ((opt = getopt_long(argc, argv, "v:e:s:", long_options, nullptr)) != -1) {
        try{
            switch (opt) {
                    case 'v':
                        v_num = std::stoi(optarg);
                        if (v_num <= 0) throw std::invalid_argument("vertices must be positive");
                        break;
                    case 'e':
                        e_num = std::stoi(optarg);
                        if (e_num <= 0) throw std::invalid_argument("edges must be positive");
                        break;
                    case 's':
                        seed = std::stoi(optarg);
                        if (seed <= 0) throw std::invalid_argument("seed must be positive");
                        break;
                    default:
                        std::cerr << "Usage: " << argv[0]
                                << " -v <Vertex Number> -e <Edge Number> -s <Seed>\n";
                        exit(EXIT_FAILURE);
                }
        } catch (const std::invalid_argument& e) {
            std::cerr << "Invalid argument for -" << (char)opt << ": not a number\n";
            exit(EXIT_FAILURE);
        } catch (const std::out_of_range& e) {
            std::cerr << "Invalid argument for -" << (char)opt << ": number out of range\n";
            exit(EXIT_FAILURE);
        }
    }
    if (v_num<0 || e_num<0 || seed<0) {
        std::cerr << "Error: missing or invalid arguments\n";
        std::cerr << "Usage: " << argv[0]
                << " -v <Vertect Number> -e <Edges Number> -s <Seed>\n";
        exit(EXIT_FAILURE);
    }
    int max_edges= (v_num*(v_num-1))/2 ;

    if(e_num>max_edges){
        std::cerr << "Error: Too many edges\n";
        exit(EXIT_FAILURE);
    }
    this->vertices = v_num;
    this->EdgesNum = e_num;
    this->adjMat.resize(v_num, std::vector<int>(v_num, inf));

    CreateRandomGraph(v_num,e_num,seed);
```

מגישים:
איתמר בבאי 206847584
שריאל משה 322772880

את יצירת הגרף הרנדומלי נבצע כך:

- יצירת ווקטור שיכיל את כל זוגות הצלעות האפשריות.
- ערבוב הווקטור בעזרת הגרעי הרנדומלי שקיבלנו בקלט.
- הוספה לגרף את e הצלעות הראשונות בווקטור הרנדומלי (כאשר e זה מספר הצלעות הנדרשות)

```
void Graph::CreateRandomGraph(size_t v_num,size_t e_num, size_t seed){
    std::vector<std::pair<int, int>> allEdges;
    for (size_t u = 0; u < v_num; ++u) {
        for (size_t v = u + 1; v < v_num; ++v) {
        allEdges.emplace_back(u, v);
        }
    }
    std::mt19937 gen(seed);
    std::shuffle(allEdges.begin(), allEdges.end(), gen);

    for (size_t i = 0; i < e_num && i<allEdges.size(); ++i) {
        int u = allEdges[i].first;
        int v = allEdges[i].second;
        std::cout<<"( "<<u<<" , "<<v<<" ) "<<std::endl;
        adjMat[u][v] = 1;
        adjMat[v][u] = 1;
    }
     printGraph();
}
```

לבסוף נפעיל את מציאת מעגל אויילר על הגרף שנוצר ונקבל אחת מהאופציות הבאות:

1. אין מעגל אויילר כי קיימת דרגה אי זוגית בגרף של הקודקוד i .
2. אין מעגל אויילר כי הגרף לא קשיר, הקודקוד s לא מגיע אל הקודקוד i.
3. יש מעגל אויילר והוא : (תצוגה של מעגל אויילר בקודקודים וחצים)

```
int main(int argc, char* argv[]) {
    Graph g;
    g.parseFlags(argc,argv);
    std::vector<int> euler;
    if (g.EulerianCycle(euler)) {
        std::cout << "Eulerian Cycle exists:\n";
        for (int v : euler) {
            std::cout << v << " -> ";
        }
        std::cout << "\n";
    } else {
        std::cout << "No Eulerian Cycle exists.\n";
    }
    return 0;
}
```

נציג 2 קלטים (seed=42):

מגישים :
איתמר בבאי 206847584
שריאל משה 322772880

א. 5 קודקודים , 10 צלעות – הגרף המלא כאשר כל הדרגות הם 4 לכן קיים מעגל אויילר.

ב. 6 קודקודים, 15 צלעות – הגרף המלא כאשר כל הדרגות הם 5 – אין מעגל אויילר.

```
itamar@DESKTOP-2A3BOQL:~/projects_CPP/OS_Final_Assignment$ ./main -v 5 -e 10 -s 42
( 0 , 2 )
( 1 , 4 )
( 2 , 3 )
( 0 , 1 )
( 1 , 3 )
( 3 , 4 )
( 2 , 4 )
( 0 , 3 )
( 0 , 4 )
( 1 , 2 )
      0   1   2   3   4
    --------------------
 0 | inf   1   1   1   1
 1 |   1 inf   1   1   1
 2 |   1   1 inf   1   1
 3 |   1   1   1 inf   1
 4 |   1   1   1   1 inf
Eulerian Cycle exists:
0 -> 1 -> 2 -> 0 -> 3 -> 1 -> 4 -> 2 -> 3 -> 4 -> 0 ->
```

```
itamar@DESKTOP-2A3BOQL:~/projects_CPP/OS_Final_Assignment$ ./main -v 6 -e 15 -s 42
( 0 , 5 )
( 1 , 4 )
( 0 , 3 )
( 2 , 4 )
( 3 , 4 )
( 4 , 5 )
( 0 , 2 )
( 2 , 3 )
( 3 , 5 )
( 2 , 5 )
( 0 , 4 )
( 0 , 1 )
( 1 , 5 )
( 1 , 3 )
( 1 , 2 )
      0   1   2   3   4   5
    ------------------------
 0 | inf   1   1   1   1   1
 1 |   1 inf   1   1   1   1
 2 |   1   1 inf   1   1   1
 3 |   1   1   1 inf   1   1
 4 |   1   1   1   1 inf   1
 5 |   1   1   1   1   1 inf
Graph is not Eulerian: vertex 0 has odd degree
No Eulerian Cycle exists.
```

מגישים :
איתמר בבאי 206847584
שריאל משה 322772880

שאלה 4:

בתיקיית gcov_reports נמצאים הדוחות של cpp.gcov. שמציגים את השורות שהגענו אליהם.
כאן נציג את הדוח שיוצא הטרמינל.
עבור main.cpp : עם ההרצה של 2 גרף עם מעגל אויילר וגרף ללא מעגל אויילר.

```
itamar@DESKTOP-2A3BOQL:~/projects_CPP/OS_Final_Assignment$ gcov main.cpp
File '/usr/include/c++/13/bits/stl_construct.h'
Lines executed:100.00% of 4
Creating 'stl_construct.h.gcov'

File '/usr/include/c++/13/bits/new_allocator.h'
Lines executed:100.00% of 4
Creating 'new_allocator.h.gcov'

File '/usr/include/c++/13/bits/stl_iterator.h'
Lines executed:100.00% of 11
Creating 'stl_iterator.h.gcov'

File '/usr/include/c++/13/bits/stl_vector.h'
Lines executed:100.00% of 26
Creating 'stl_vector.h.gcov'

File '/usr/include/c++/13/bits/alloc_traits.h'
Lines executed:100.00% of 3
Creating 'alloc_traits.h.gcov'

File '/usr/include/c++/13/bits/allocator.h'
Lines executed:100.00% of 1
Creating 'allocator.h.gcov'

File 'main.cpp'
Lines executed:100.00% of 12
Creating 'main.cpp.gcov'

Lines executed:100.00% of 61
```

עבור Graph.cpp אתאר אילו הרצות ביצעתי בשביל להגיע לכיסוי :

- חוסר בדגלים , דגלים לא קיימים.
- קלט בטווח לא מתאים או לא מספר.
- יותר מידי צלעות.
- קלט תקין ללא מעגל אויילר.
- קלט תקין עם מעגל אויילר.

```
File 'Graph.cpp'
Lines executed:97.06% of 136
Creating 'Graph.cpp.gcov'

File '/usr/include/c++/13/iomanip'
Lines executed:100.00% of 2
Creating 'iomanip.gcov'

File '/usr/include/c++/13/new'
Lines executed:66.67% of 3
Creating 'new.gcov'

File '/usr/include/x86_64-linux-gnu/c++/13/bits/c++config.h'
Lines executed:100.00% of 2
Creating 'c++config.h.gcov'

Lines executed:71.50% of 1709
```

כעת נבצע בדיקת זליגת זיכרון באמצעות valgrind :

מגישים:
איתמר בבאי 206847584
שריאל משה 322772880

ניתן לראות שאין שום הקצאה של משתנים בheap (כי לא ביצענו new) ושאין זליגת זיכרון.

```
itamar@DESKTOP-2A3BOQL:~/projects_CPP/OS_Final_Assignment$ make valgrind-memcheck
g++ -c -g -Wall --coverage Graph.cpp -o Graph.o
g++ -o main main.o Graph.o -g --coverage
valgrind --tool=memcheck --leak-check=full --show-leak-kinds=all ./main -v 5 -e 10 -s 42
==125686== Memcheck, a memory error detector
==125686== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==125686== Using Valgrind-3.22.0 and LibVEX; rerun with -h for copyright info
==125686== Command: ./main -v 5 -e 10 -s 42
==125686==
       0   1   2   3   4
    --------------------
 0 | inf   1   1   1   1
 1 |   1 inf   1   1   1
 2 |   1   1 inf   1   1
 3 |   1   1   1 inf   1
 4 |   1   1   1   1 inf
Eulerian Cycle exists:
0 -> 1 -> 2 -> 0 -> 3 -> 1 -> 4 -> 2 -> 3 -> 4 -> 0 ->
libgcov profiling error:/home/itamar/projects_CPP/OS_Final_Assignment/Graph.gcda:overwriting an existing prof
fferent checksum
==125686==
==125686== HEAP SUMMARY:
==125686==     in use at exit: 0 bytes in 0 blocks
==125686==   total heap usage: 34 allocs, 34 frees, 85,419 bytes allocated
==125686==
==125686== All heap blocks were freed -- no leaks are possible
==125686==
==125686== For lists of detected and suppressed errors, rerun with: -s
==125686== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

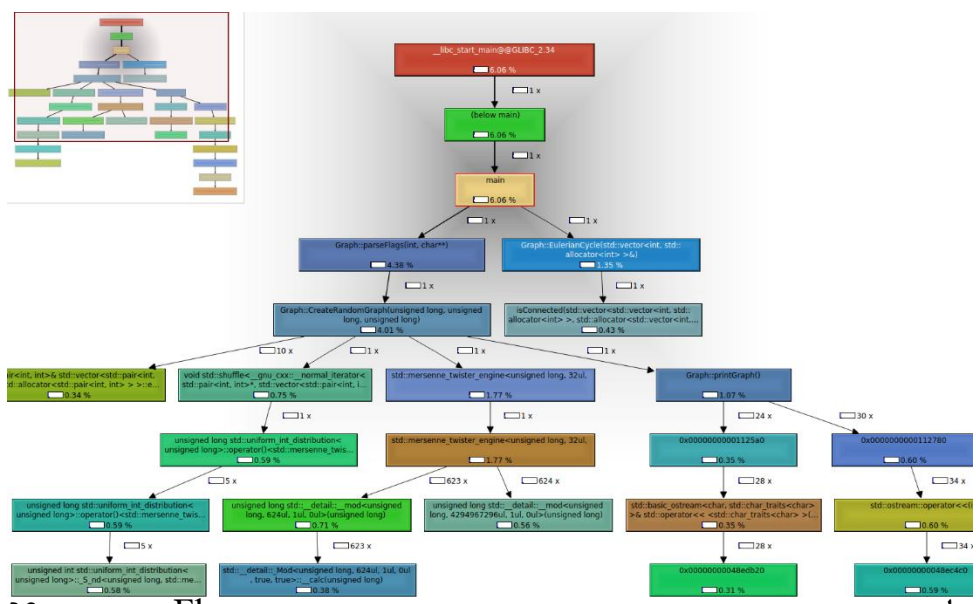
נציג גם את גרף הקריאות שמתאר את חלוקת הקריאות בין הפונקציות השונות, מי קוראת למי וכמה פעמים כל פונקציה נקראת.
באופן זה נזהה צווארי בקבוק.
כדי לראות תצוגה גרפית תקינה נתקין כלי בשם KCachegrind עם הפקודה :
sudo apt install kcachegrind

ואחרי שהרצנו:
Make valgrind-callgrind
נריץ את השורה הבאה:
*.kcachegrind callgrind.out

מגישים :
איתמר בבאי 206847584
שריאל משה 322772880

```
itamar@DESKTOP-2A3BOQL:~/projects_CPP/OS_Final_Assignment$ make valgrind-callgrind
g++ -o main main.o Graph.o -g --coverage
valgrind --tool=callgrind ./main -v 5 -e 10 -s 42
==126116== Callgrind, a call-graph generating cache profiler
==126116== Copyright (C) 2002-2017, and GNU GPL'd, by Josef Weidendorfer et al.
==126116== Using Valgrind-3.22.0 and LibVEX; rerun with -h for copyright info
==126116== Command: ./main -v 5 -e 10 -s 42
==126116==
==126116== For interactive control, run 'callgrind_control -h'.
       0   1   2   3   4
    --------------------
 0 | inf   1   1   1   1
 1 |   1 inf   1   1   1
 2 |   1   1 inf   1   1
 3 |   1   1   1 inf   1
 4 |   1   1   1   1 inf
Eulerian Cycle exists:
0 -> 1 -> 2 -> 0 -> 3 -> 1 -> 4 -> 2 -> 3 -> 4 -> 0 ->
==126116==
==126116== Events    : Ir
==126116== Collected : 3768940
==126116==
==126116== I   refs:      3,768,940
```

Ir	Ir per call	Count	Callee
4.38	164 949	1	Graph::parseFlags(int, char**) (main: Graph.cpp, ...)
1.35	50 910	1	Graph::EulerianCycle(std::vector<int, std::allocator<int> >&) (main: Graph.cpp)
0.12	407	11	0x0000000000112780
0.11	313	13	0x00000000001125a0
0.06	2 108	1	Graph::~Graph() (main: Graph.cpp)
0.02	50	12	bool __gnu_cxx::operator!=<int*, std::vector<int, std::allocator<int> > >(__gnu_cxx::__normal_iterator<int*, std:
0.01	294	1	std::vector<int, std::allocator<int> >::~vector() (main: stl_vector.h, ...)
0.00	15	11	__gnu_cxx::__normal_iterator<int*, std::vector<int, std::allocator<int> > >::operator++() (main: stl_iterator.h)
0.00	11	11	__gnu_cxx::__normal_iterator<int*, std::vector<int, std::allocator<int> > >::operator*() const (main: stl_iterator.h
0.00	95	1	Graph::Graph() (main: Graph.cpp)
0.00	73	1	std::vector<int, std::allocator<int> >::vector() (main: stl_vector.h)
0.00	41	1	std::vector<int, std::allocator<int> >::end() (main: stl_vector.h)
0.00	40	1	std::vector<int, std::allocator<int> >::begin() (main: stl_vector.h)



ניתן לראות כי הפונקציה שצרכה הכי הרבה משאבים היא parseFlags עם 4.38 מיליון הוראות למעבד.

<u>שאלה 5</u> : נכתבה בטעות לא סופק קוד.

מגישים :
איתמר בבאי 206847584
שריאל משה 322772880

<u>שאלה 6</u> : בשאלה זו התבקשנו לממש תקשורת שרת לקוח, הלקוח ישלח גרף לשרת ויקבל מעגל אויילר אם יש או יקבל שאין.

צד שרת : (TCP)

- נשתמש באופציות על מנת לפרסר את הפורט שעליו השרת יאזין.

```
    while ((opt = getopt(argc, argv, "T:")) != -1) {
        switch (opt) {

            case 'T':
                if(std::atoi(optarg) <= 0) {
                    std::cerr << "Error: any flag must be a positive integer." << std::endl;
                    exit(EXIT_FAILURE);
                }
                port_tcp = std::atoi(optarg);
                if (port_tcp <= 0 || port_tcp > 65535) { // Check if port is valid
                    std::cerr << "Error: Port must be a positive integer between 1 and 65535." << std::endl;
                    exit(EXIT_FAILURE);
                }
                has_tcp = true;
                break;

            default:
                std::cerr << "Usage: " << argv[0]
                          << " -T <tcp_port>\n";
                exit(EXIT_FAILURE);
        }
    }
    if (!has_tcp ) {
        std::cerr << "Error: Missing TCP port." << std::endl;
        exit(EXIT_FAILURE);
    }

    run_server(port_tcp, g); // Start the server with the provided TCP port

    return 0;
```

מגישים :
איתמר בבאי 206847584
שריאל משה 322772880

- בפונקציית run_server נבצע מספר פעולות על מנת לאפשר קבלת תקשורת מלקוח.
- הגדרת משתנים ומבנים הדרושים ליצירת socket ו bind
- יצירת ה socket המאזין של השרת.
- שינוי הגדרת ה socket בפקודה setsocketopt המאפשרת שימוש חוזר בפורט.
- הגדרת הכתובת בתור ipv4 והמרת הפורט למבנה מתאים.
- יצירת חיבור בין הכתובת לפורט באמצעות bind.
- הגדרת התור להמתנה לaccept ל10 לקוחות באמצעות listen.

```
void run_server(int port_tcp, Graph& g) {

    int server_fd, new_socket, max_sd, activity, sd;
    int client_socket[FD_SETSIZE] = {0};
    struct sockaddr_in address;
    fd_set readfds;

    // Create a TCP socket
    if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0) {
        throw std::runtime_error("socket failed");
    }
    int opt = 1;
    // Set socket options to allow reuse of the address
    if (setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR, &opt, sizeof(opt)) < 0) {
        throw std::runtime_error("setsockopt failed");
    }

    // Set up the address structure for TCP
    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons(port_tcp);
    if (bind(server_fd, (struct sockaddr *)&address, sizeof(address)) < 0) {// Bi
        throw std::runtime_error("bind failed");
    }
    if (listen(server_fd, 10) < 0) {// Listen for incoming connections
        throw std::runtime_error("listen failed");
    }
```

מגישים :
איתמר בבאי 206847584
שריאל משה 322772880

בלולאה ראשית :

- נכניס אל קבוצת fd את stdin ואת כל הלקוחות.
- נמצא את המספר המקסימלי של fd כדי שנדע את הטווח שצריך להעביר ל select.
-

```
while (true) {// Main loop for handling connections and commands
    FD_ZERO(&readfds);
    FD_SET(server_fd, &readfds); // Add TCP server socket to the set
    FD_SET(0, &readfds); // Add stdin to the set


    max_sd = server_fd;
    if (max_sd < 0) max_sd = 0;

    for (int i = 0; i < FD_SETSIZE; i++) { // Add all client sockets to the set
        sd = client_socket[i];
        if (sd > 0)
            FD_SET(sd, &readfds);
        if (sd > max_sd)
            max_sd = sd;
    }
```

```
activity = select(max_sd + 1, &readfds, nullptr, nullptr, nullptr);// Wait
if (activity < 0 && errno != EINTR) {
    std::cout << "select error" << std::endl;
    break;
}
```

מגישים:
איתמר בבאי 206847584
שריאל משה 322772880

בעת קבלת מידע מלקוח ניכנס ללולאה הבאה לפירוש המידע:

- קבלת גודל המטריצה ובדיקת תקינות.
- יצירת המטריצה מהגודל המתאים שהתקבל.
- בלולאה קבלת שורה אחר שורה של קודקודים במטריצה והזנתם ישירות במטריצה החדשה.
- במידה וכל המידע התקבל באופן תקין נעדכן את הפרמטרים המתאימים של המטריצה בפונקציית parseFromMatrix ונפעיל את האלגוריתם שבנינו על הגרף.

```
        for (int i = 0; i < FD_SETSIZE; i++) {
            sd = client_socket[i];
            if (sd > 0 && FD_ISSET(sd, &readfds)) {
                // First read the matrix size (number of vertices)
                int n = 0;
                size_t received = recv(sd, &n, sizeof(n), MSG_WAITALL);// Read the size of the mat
                if (received != sizeof(n)) {
                    std::cerr << "Failed to receive matrix size\n";
                    close(sd);
                    client_socket[i] = 0;
                    continue;
                }
                // Read adjacency matrix
                std::vector<std::vector<int>> newMatrix(n, std::vector<int>(n));
                bool success = true;
                for (int row = 0; row < n; ++row) {
                    received = recv(sd, newMatrix[row].data(), n * sizeof(int), MSG_WAITALL);
                    if (received != n * sizeof(int)) {
                        std::cerr << "Failed to receive row " << row << "\n";
                        success = false;
                        break;
                    }
                }
                if (success) {
                    std::cout << "Received adjacency matrix of size " << n << "x" << n << "\n";
                    g.parseFromMatrix(newMatrix);
                    std::cout << "Graph updated.\n";
                    g.printGraph();
                    CheckForEulerianCycle(g, sd);
                }
                else{
```

- שליחת התשובה תתבצע בפונקציה הבאה:

```
void CheckForEulerianCycle(Graph& g, int socket_fd){
    std::vector<int> euler;
    std::string result;
    if (g.EulerianCycle(euler)) {
        std::cout << "Eulerian Cycle exists:\n";
        result = "Eulerian Cycle exists:\n";
        for (int v : euler) {
            std::cout << v << " -> ";
            result += std::to_string(v) + " -> ";
        }
        std::cout << "\n";
        result += "\n";

    } else {
        result = "No Eulerian Cycle exists.\n";
        std::cout << "No Eulerian Cycle exists.\n";
    }

    send(socket_fd, result.c_str(), result.size(), 0);
}
```

מגישים:
איתמר בבאי 206847584
שריאל משה 322772880

צד לקוח:

- בדומה לשרת נפרסר את המידע שהתקבל בדגלים בעזרת getopt.

```
int main(int argc, char* argv[]) {
    if (argc < 3) { // Check if there are enough arguments
    }
    const char* hostname;
    int port;
    int opt;
    bool has_host = false, has_port = false;

     while ((opt = getopt(argc, argv, "h:p:")) != -1) {
        switch (opt) {

            case 'h':
                hostname = optarg;
                has_host = true;
                break;
            case 'p':

                port = std::atoi(optarg);
                has_port = true;
                if (port <= 0 || port > 65535) { // Check if port is valid
                    std::cerr << "Error: Port must be a positive integer between 1 and 6553
                    exit(EXIT_FAILURE);
                }
                break;
```

- נחלץ את כתובת הip של hostname באמצעות gethostname.
- ניצור את הsocket שאיתו נתחבר לשרת.
- ניצור את ה struct של הaddress ונתאים אותם כנדרש.
- נבצע connect לאותו ip באותו port שהתקבלו באופציות.

```
    struct hostent* server = gethostbyname(hostname);
    if (!server) {
        std::cerr << "No such host: " << hostname << std::endl;
        return 1;
    }

    // Create socket
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0) {
        std::cerr << "Error opening socket" << std::endl;
        return 1;
    }
    // Set up server address structure
    struct sockaddr_in serv_addr;
    std::memset(&serv_addr, 0, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    std::memcpy(&serv_addr.sin_addr.s_addr, server->h_addr, server->h_length);
    serv_addr.sin_port = htons(port);

    // Connect to server
    if (connect(sockfd, (struct sockaddr*)&serv_addr, sizeof(serv_addr)) < 0) {
        std::cerr << "Connection failed" << std::endl;
        close(sockfd);
        return 1;
    }
```

- בשלב זה נקבל מהמשתמש את המטריצה בקלט של stdin ונמלא את המטריצה של הלקוח.
- לאחר מכן נשלח אותה שורה אחר שורה בפונקציה הבאה

```
bool sendMatrix(int sockfd, const std::vector<std::vector<int>>& adjMat) {
    int n = adjMat.size();

    if (send(sockfd, &n, sizeof(n), 0) < 0) {
        std::cerr << "Failed to send matrix size\n";
        return false;
    }

    for (int i = 0; i < n; ++i) {
        if (send(sockfd, adjMat[i].data(), n * sizeof(int), 0) < 0) {
            std::cerr << "Failed to send matrix row " << i << "\n";
            return false;
        }
    }

    return true;
}
```

מגישים :
איתמר בבאי 206847584
שריאל משה 322772880

שאלה 7 :

התבקשנו לבחור 4 אלגוריתמים והפעלתם על הגרף של הלקוח לפי בחירה, תוך שימוש בfactory ן strategy .

נבחרו האלגוריתמים הבאים :

- Mst
- Max flow
- Path cover
- SCC

-יצרנו אובייקט אבסטרקטי בשם algorithms שכל האלגוריתמים ירשו ממנו ויממשו את הפונקציה הווירטואלית הטהורה בשם activate .
-כמו כן מימשנו factory ליצירת כל אובייקט תוך שימוש בפולימורפיזם בעזרת האובייקט האבסטרקטי.
-האובייקט strategy קיבל מצביע לאלגוריתם וקריאה לפונקציית execute שלו הפעיל את הactivate המתאימה של האלגוריתם הנכון.

בהמשך אציג רק את strategy , factory , algorithms ואת אופן השילוב בין שניהם ע״י השרת.

```
class Algorithms{
    public:
    ~Algorithms()=default;
    virtual std::string activate(Graph& g)=0;
};

#endif
```

```
class AlgorithmFactory {
public:
    static Algorithms* createAlgorithm(const std::string& name) {
        if (name == "MST")
            return new MST();
        else if (name == "SCC")
            return new SCC();
        else if (name == "MaxFlow")
            return new MaxFlow();
        else if (name == "PathCover")
            return new PathCover();
        else
            return nullptr;
    }
};

#endif
```

מגישים:
איתמר בבאי 206847584
שריאל משה 322772880

```cpp
class Strategy {
    Algorithms* stra = nullptr;

public:
    void setStrategy(Algorithms* alg) {
        stra = alg;
    }

    std::string execute(Graph& graph) {
        if (!stra) return "No strategy set.";
        return stra->activate(graph);
    }
};
#endif
```

```cpp
if (success) {
    std::cout << "Received adjacency matrix of size " << n << "x" << n << "\n";
    g.parseFromMatrix(newMatrix);
    std::cout << "Graph updated.\n";
    g.printGraph();

    Strategy stra; //create a Strategy object
    // Create the algorithm object using the factory
    Algorithms* algObj = AlgorithmFactory::createAlgorithm(alg);

    stra.setStrategy(algObj); // Set the strategy with the created algorithm
  std::string result = stra.execute(g);

int result_len = result.size();
if (send(sd, &result_len, sizeof(result_len), 0) < 0) {
    std::cerr << "Failed to send result length\n";
    close(sd);
    client_socket[i] = 0;
    continue;
}

if (send(sd, result.c_str(), result_len, 0) < 0) {
    std::cerr << "Failed to send result to client\n";
    close(sd);
    client_socket[i] = 0;
    continue;
}
std::cout << "Result sent to client." << std::endl;
```

מגישים :
איתמר בבאי 206847584
שריאל משה 322772880

שאלה 8 :

בשאלה זו התבקשנו לממש leader follower עם threads מרובים.

- נגדיר תור של משימות taskQueue שיכיל גרף ו fd של socket .
- נגדיר mutex להגנה על הגרף.
- נגדיר condition variable לטובת הודעה לthread על הגעה של משימה חדשה.
- נגדיר משתנה גלובלי אטומי שיצביע על פעילות השרת.
- נגדיר פונקציה בשם workerFunction שתהיה הפונקציה שהthread מפעיל , בפונקציה זו נריץ את כלל האלגוריתמים על הגרף הנתון.

```
// Task structure for Leader-Follower pattern
struct Task {
    int client_socket;
    std::vector<std::vector<int>> matrix;
};

// Global variables for Leader-Follower
std::queue<Task> taskQueue; // Queue to hold tasks
std::mutex queueMutex; // Mutex for thread-safe access to the queue
std::condition_variable queueCV; // Condition variable to notify workers
std::atomic<bool> serverRunning{true}; // Flag to control server running state
```

```
// Worker function that runs 4 algorithms and sends results back
void workerFunction() {
    while (serverRunning) {
        std::unique_lock<std::mutex> lock(queueMutex);
        queueCV.wait(lock, []{ return !taskQueue.empty() || !serverRunning; });

        if (!serverRunning && taskQueue.empty()) break;

        Task task = taskQueue.front();
        taskQueue.pop();
        lock.unlock();

        // Process the task
        Graph g;
        g.parseFromMatrix(task.matrix);

        // Run all 4 algorithms
        std::string results = "";

        // MST
        Strategy mst_stra;
        Algorithms* mst_alg = AlgorithmFactory::createAlgorithm("MST");
        mst_stra.setStrategy(mst_alg);
        results += "MST: " + mst_stra.execute(g) + "\n";
        delete mst_alg;
```

מגישים:
איתמר בבאי 206847584
שריאל משה 322772880

תיאור כרונולוגי של התהליך:

- האזנה של השרת יצירת הthreads והמתנה ב accept ל connect מלקוח.
- בצד הלקוח שליחת ה connect , בחירת גרף רנדומלי או נתון והעברת הנתונים בהתאמה לשרת.
- בצד השרת המידה ונבחר גרף רנדומלי מייצר גרף חדש לפי הנתונים שהתקבלו (קודקודים וצלעות), במידה והגיע גרף נתון מקבל אותו.
- יצרת הstruct של המשימה החדשה שהתקבלה והכנסתה לתור המשימות, הפעלת notify של condition variable כדי שהthread הפנוי הבא ייקח את המשימה ויבצע אותה.
- חישוב האלגוריתמים ושליחת תשובה ללקוח.

גרף רנדומלי:

```
void CreateRandomGraph(size_t v_num, size_t e_num, Graph& g, int max_weight = 20) {
    std::cout << "Creating random graph with " << v_num << " vertices and " << e_num << " edges." << std::endl;
    g.vertices = v_num;
    g.EdgesNum = 0;
    g.adjMat.assign(v_num, std::vector<int>(v_num, 0));

    std::vector<std::pair<int, int>> allEdges;
    for (size_t u = 0; u < v_num; ++u) {
        for (size_t v = u + 1; v < v_num; ++v) {
            allEdges.emplace_back(u, v);
        }
    }
    std::random_device rd;
    std::mt19937 gen(rd());
    std::shuffle(allEdges.begin(), allEdges.end(), gen);
    std::uniform_int_distribution<int> weight_dist(1, max_weight);
    for (size_t i = 0; i < e_num && i < allEdges.size(); ++i) {
        int u = allEdges[i].first;
        int v = allEdges[i].second;
        int w = weight_dist(gen);
        g.adjMat[u][v] = w;
        g.adjMat[v][u] = w;
        g.EdgesNum++;

    }
      g.printGraph();
}
```

הפרדה בין מטריצה נתונה לבין בחירה ברנדומלי:

```
            if (type == "matrix") {
                newMatrix.resize(n, std::vector<int>(n));
                for (int row = 0; row < n; ++row) {
                    received = recv(sd, newMatrix[row].data(), n * sizeof(int), MSG_WAITALL);
                    if (received != n * sizeof(int)) {
                        std::cerr << "Failed to receive row " << row << "\n";
                        success = false;
                        break;
                    }
                }
            } else if (type == "random") {
                size_t e_num = 0;
                received = recv(sd, &e_num, sizeof(e_num), MSG_WAITALL);
                if (received != sizeof(e_num)) {
                    std::cerr << "Failed to receive number of edges\n";
                    close(sd);
                    client_socket[i] = 0;
                    continue;
                }

                std::cout << "Received random graph request with vertices: " << n << " edges: " << e_num << "\n";
                Graph tempGraph;
                CreateRandomGraph(n, e_num, tempGraph);
```

הכנסת המשימה לתור והודעה בעזרת condition variable:

```
if (success) {
    std::cout << "Received adjacency matrix of size " << n << "x" << n << "\n";
    {
        std::lock_guard<std::mutex> lock(queueMutex);
        taskQueue.push({sd, newMatrix});
    }
    queueCV.notify_one();

    std::cout << "Task added to queue for processing" << std::endl;
} else {
    close(sd);
    client_socket[i] = 0;
}
```

מגישים :
איתמר בבאי 206847584
שריאל משה 322772880

שאלה 9:

ארבעת האובייקטים הפעילים יהיו אחד לכל אלגוריתם.

התהליך המרכזי

מקבל את הקלט מהמשתמש ומכין אותו לקראת הפעלת התהליך הבא, מכניס משימה לתור של ה thread של mst ומיידע אותו בעזרת CV .

mst thread

מקבל את הקלט מחשב את המשקל המינימלי , כותב אותו בתוצאה של המשימה.
מוסיף את המשימה ל thread של maxflow ומפעיל את CV שלו.

maxFlow

מקבל את הקלט מחשב את הזרימה המקסימלית, כותב אותו בתוצאה שלהבמשימה.
מוסיף את המשימה ל thread של pathcover ומפעיל את CV שלו.

pathCover

מקבל את הקלט מחשב את המסלולים הזרים, כותב אותם בתוצאה של המשימה.
מוסיף את המשימה ל thread של SCC ומפעיל את CV שלו.

SCC

מקבל את הקלט מחשב את רכיבי הקשירות החזקה. כותב אותם בתוצאה של המשימה.
מחזיר את התשובה המשורשרת של כל התהליכים בחזרה ללקוח.

התהליך המרכזי

ממשיך לקבל קריאות חדשות מלקוחות אחרים (או אותו לקוח) וחוזר להפעיל את התהליך הראשון.

מגישים :
איתמר בבאי 206847584
שריאל משה 322772880

נציג את הנקודות המרכזיות בקוד :

הגדרות ראשוניות לכל thread :

- הגדרנו תור משימות
- Condition variable
- Mutex

הוספנו למבנה של ה task את התוצאה שנבנית בתהליך.

```
struct Task {
    int client_socket;
    std::vector<std::vector<int>> matrix;
    std::string results = "";
};

// Global variables for Leader-Follower
std::queue<Task> mst_task_Q,maxFlow_task_Q,pathCover_task_Q,scc_task_Q; // Queue to hold tasks
std::mutex mstQueueMutex,maxFlowQueueMutex,pathCoverQueueMutex,sccQueueMutex; // Mutex for thread-safe
std::condition_variable mstCV,maxFlowCV,pathCoverCV,sccCV; // Condition variable to notify workers

std::atomic<bool> serverRunning{true}; // Flag to control server running state
```

פיצלנו את הפונקציה הראשית (מתרגיל 8) שהייתה לכל thread ל4 פונקציות כך שכל סוג של thread קיבל אחת אחרת.

בכל פונקציה נבצע מספר פעולות :

- הגנה בעזרתmutex בזמן עבודה על התור המשותף.
- הוצאת המשימה מהתור.
- ביצוע המשימה בעזרת strategy ו factory (יצירת האובייקט שיבצע).
- הכנסת התוצאה לresult של המשימה.
- נעילה נוספת של התור של הthread הבא בpipe.
- הכנסת המשימה לתור שלו.
- הודעה ל CV שלו שיש משימה חדשה.

למשל הthread שאחראי לחישוב mst קיבל את הפונקציה הבאה

```
void MST_worker() {
    while (serverRunning) {
        std::unique_lock<std::mutex> lock(mstQueueMutex);
        mstCV.wait(lock, []{ return !mst_task_Q.empty() || !serverRunning; });

        if (!serverRunning && mst_task_Q.empty()) break;
        Task task = mst_task_Q.front();
        mst_task_Q.pop();
        lock.unlock();

        // Process the task
        g.parseFromMatrix(task.matrix);

        // MST
        Strategy mst_stra;
        Algorithms* mst_alg = AlgorithmFactory::createAlgorithm("MST");
        mst_stra.setStrategy(mst_alg);
        task.results += "MST: " + mst_stra.execute(g) + "\n";
        delete mst_alg;
        {
            std::lock_guard<std::mutex> lock(maxFlowQueueMutex);
            maxFlow_task_Q.push({task.client_socket, task.matrix, task.results});
        }
        maxFlowCV.notify_one();

    }
}
```

מגישים :
איתמר בבאי 206847584
שריאל משה 322772880

התאמה הThreads לפונקציות :

```
std::vector<std::thread> workers; // Create worker threads

workers.emplace_back(MST_worker);
workers.emplace_back(MaxFlow_worker);
workers.emplace_back(PathCover_worker);
workers.emplace_back(SCC_worker);
```

התחלת הpipline

```
if (success) {
    std::cout << "Received adjacency matrix of size " << n << "x" << n << "\n";
    {
        std::lock_guard<std::mutex> lock(mstQueueMutex);
        mst_task_Q.push({sd, newMatrix});
    }
    mstCV.notify_one();

    std::cout << "Task added to queue for processing" << std::endl;
} else {
```

שאלה 10:

Valgrind-memcheck

```
itamar@DESKTOP-2A3BOQL:~/projects_CPP/OS_Final_Assignment/tar9$ make valgrind-memcheck
valgrind valgrind --tool=memcheck --leak-check=full --show-leak-kinds=all ./server -T 9999
==43081== Memcheck, a memory error detector
==43081== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==43081== Using Valgrind-3.22.0 and LibVEX; rerun with -h for copyright info
==43081== Command: /usr/bin/valgrind --tool=memcheck --leak-check=full --show-leak-kinds=all ./server -T 9999
==43081==
==43081== Memcheck, a memory error detector
==43081== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==43081== Using Valgrind-3.22.0 and LibVEX; rerun with -h for copyright info
==43081== Command: ./server -T 9999
==43081==
Server listening over tcp on port 9999
New connection, socket fd: 4
```

```
Server listening over tcp on port 9999
New connection, socket fd: 4
Received matrix size: 10
Received adjacency matrix of size 10x10
Task added to queue for processing
MST weight: 54
Max Flow: 2
The path cover is:
Path Cover (1 paths):
0 -> 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 -> 8 -> 9

The strongly connected components are:
SCCs (1 components):
0 1 2 3 4 5 6 7 8 9

Task completed and sent to client
Failed to receive type
exit
Exiting server...
Server closed
==43081==
==43081== HEAP SUMMARY:
==43081==     in use at exit: 0 bytes in 0 blocks
==43081==   total heap usage: 256 allocs, 256 frees, 116,752 bytes allocated
==43081==
==43081== All heap blocks were freed -- no leaks are possible
==43081==
==43081== For lists of detected and suppressed errors, rerun with: -s
==43081== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
itamar@DESKTOP-2A3BOQL:~/projects_CPP/OS_Final_Assignment/tar9$
```

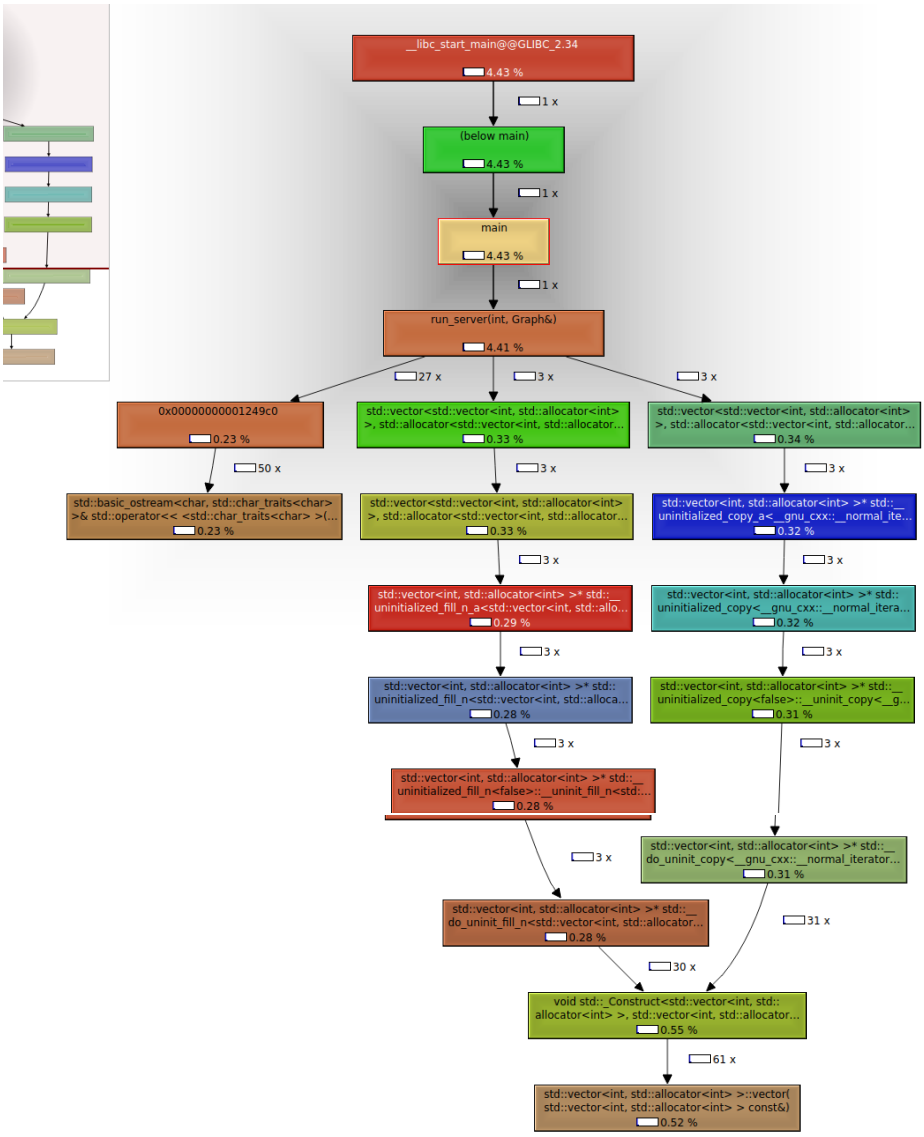
מגישים :
איתמר בבאי 206847584
שריאל משה 322772880

עבור callgrind נריץ 3 לקוחות וכל אחד מהם ישלח גרף, נקבל את התוצאות הבאות :

ניתן לראות שכל הקריאות עוברות דרך runserver.

ניתן גם לראות שישנם הקצאות זיכרון מרובות של וקטור _ניתן לייעל באמצעות שימוש באלגוריתמים כמו move.

Ir	Ir per call	Count	Callee
4.41	387 192	1	run_server(int, Graph&) (server: Graph.cpp, ...)
0.01	349	2	0x00000000001246f0
0.00	176	2	0x0000000000124bc0
0.00	150	1	Graph::~Graph() (server: Graph.hpp)
0.00	129	1	std::vector<int, std::allocator<int> >::~vector() (server: stl_vector.h, ...)
0.00	77	1	Graph::Graph() (server: Graph.cpp)
0.00	59	1	std::vector<int, std::allocator<int> >::vector() (server: stl_vector.h)



מגישים :
איתמר בבאי 206847584
שריאל משה 322772880

– Helgrind

```
itamar@DESKTOP-2A3BOQL:~/projects_CPP/OS_Final_Assignment/tar9$ make valgrind-helgrind
g++ -c -g -Wall --coverage -pthread Graph.cpp -o Graph.o
Graph.cpp: In function ‘void run_server(int, Graph&)’:
Graph.cpp:321:38: warning: comparison of integer expressions of different signedness: ‘ssize_t’ {aka ‘long int’}
 and ‘long unsigned int’ [-Wsign-compare]
  321 |                         if (received != n * sizeof(int)) {
      |                                      ~~~~~~~~~~^~~~~~~~~~~~~~~~~~
g++ -o server Graph.o MST.o MaxFlow.o PathCover.o SCC.o -g --coverage -pthread
valgrind valgrind --tool=helgrind ./server -T 9999 ./server -T 9999
==64376== Memcheck, a memory error detector
==64376== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==64376== Using Valgrind-3.22.0 and LibVEX; rerun with -h for copyright info
==64376== Command: /usr/bin/valgrind --tool=helgrind ./server -T 9999 ./server -T 9999
==64376==
==64376== Helgrind, a thread error detector
==64376== Copyright (C) 2007-2017, and GNU GPL'd, by OpenWorks LLP et al.
==64376== Using Valgrind-3.22.0 and LibVEX; rerun with -h for copyright info
==64376== Command: ./server -T 9999 ./server -T 9999
==64376==
Server listening over tcp on port 9999
exit
libgcov profiling error:/home/itamar/projects_CPP/OS_Final_Assignment/tar9/Graph.gcda:overwriting an existing pr
ofile data with a different checksum
==64376==
==64376== Use --history-level=approx or =none to gain increased speed, at
==64376== the cost of reduced accuracy of conflicting-access information
==64376== For lists of detected and suppressed errors, rerun with: -s
==64376== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 119 from 32)
itamar@DESKTOP-2A3BOQL:~/projects_CPP/OS_Final_Assignment/tar9$
```

שאלה 11:

נחלץ דוח עבור שאלה 9 בלבד מכיוון שמכילה את כל הפונקציות של כל התרגילים.

קבצי ה cpp.gcov נשמרו בתיקיית code_coverage בתוך תיקייה 9.

1. נריץ את makefile של תרגיל 9.
2. נריץ את ה server ואת הclient .
3. נריץ את כל האפשרויות על מנת להגיע לכל השורות בקוד של הgraph.
4. נריץ gcov על הקובץ cpp של הגרף כדי לצפות באחוזי הביצוע ונפתח גם את הקובץ graph.cpp.gcov שנוצר על מנת להציג את הדוח.
5. כמו כן נבצע זאת על כל שאר קבצי הcpp

```
itamar@DESKTOP-2A3BOQL:~/projects_CPP/OS_Final_Assignment/tar9$ gcov Graph.cpp
File 'Graph.cpp'
Lines executed:92.04% of 314
Creating 'Graph.cpp.gcov'
```

```
File '../tar8/MST.cpp'
Lines executed:100.00% of 35
Creating 'MST.cpp.gcov'
```

```
File '../tar8/MaxFlow.cpp'
Lines executed:100.00% of 36
Creating 'MaxFlow.cpp.gcov'
```

מגישים :
איתמר בבאי 206847584
שריאל משה 322772880

```
File '../tar8/PathCover.cpp'
Lines executed:100.00% of 23
Creating 'PathCover.cpp.gcov'
```

```
File '../tar8/SCC.cpp'
Lines executed:100.00% of 48
Creating 'SCC.cpp.gcov'
```

```
File '../tar8/client.cpp'
Lines executed:78.20% of 133
Creating 'client.cpp.gcov'
```