

TSP

Wygenerowano przez Doxygen 1.9.6



<b>1 Indeks klas</b>	<b>1</b>
1.1 Lista klas	1
<b>2 Indeks plików</b>	<b>3</b>
2.1 Lista plików	3
<b>3 Dokumentacja klas</b>	<b>5</b>
3.1 Dokumentacja struktury Graph	5
<b>4 Dokumentacja plików</b>	<b>7</b>
4.1 Dokumentacja pliku File.cpp	7
4.1.1 Opis szczegółowy	7
4.1.2 Dokumentacja funkcji	8
4.1.2.1 loadGraph()	8
4.1.2.2 saveGraph()	8
4.1.2.3 split()	8
4.2 Dokumentacja pliku File.h	9
4.2.1 Opis szczegółowy	9
4.2.2 Dokumentacja funkcji	9
4.2.2.1 loadGraph()	9
4.2.2.2 saveGraph()	10
4.2.2.3 split()	10
4.3 File.h	10
4.4 Dokumentacja pliku Graph.cpp	11
4.4.1 Opis szczegółowy	11
4.4.2 Dokumentacja funkcji	11
4.4.2.1 addEdge()	11
4.4.2.2 initGraph()	12
4.5 Dokumentacja pliku Graph.h	12
4.5.1 Opis szczegółowy	13
4.5.2 Dokumentacja funkcji	13
4.5.2.1 addEdge()	13
4.5.2.2 initGraph()	13
4.6 Graph.h	15
4.7 Dokumentacja pliku main.cpp	15
4.7.1 Opis szczegółowy	15
4.7.2 Dokumentacja funkcji	16
4.7.2.1 main()	16
4.8 Dokumentacja pliku TSP.cpp	16
4.8.1 Opis szczegółowy	16
4.8.2 Dokumentacja funkcji	17
4.8.2.1 calcFitness()	17
4.8.2.2 crossover()	17

4.8.2.3 findBestRoute()	17
4.8.2.4 generateRoute()	18
4.8.2.5 mutate()	18
4.8.2.6 TSP()	18
4.9 Dokumentacja pliku TSP.h	19
4.9.1 Opis szczegółowy	19
4.9.2 Dokumentacja funkcji	20
4.9.2.1 calcFitness()	20
4.9.2.2 crossover()	20
4.9.2.3 findBestRoute()	20
4.9.2.4 generateRoute()	21
4.9.2.5 mutate()	21
4.9.2.6 TSP()	21
4.10 TSP.h	22
<b>Skorowidz</b>	<b>23</b>

# Rozdział 1

## Indeks klas

### 1.1 Lista klas

Tutaj znajdują się klasy, struktury, unie i interfejsy wraz z ich krótkimi opisami:

<a href="#">Graph</a> . . . . .	5
---------------------------------	---



## Rozdział 2

# Indeks plików

### 2.1 Lista plików

Tutaj znajduje się lista wszystkich udokumentowanych plików z ich krótkimi opisami:

<a href="#">File.cpp</a>	Funkcje operacji na plikach . . . . .	7
<a href="#">File.h</a>	Plik nagłówkowy operacji na plikach . . . . .	9
<a href="#">Graph.cpp</a>	Funkcje odpowiedzialne za graf . . . . .	11
<a href="#">Graph.h</a>	Plik nagłówkowy grafu . . . . .	12
<a href="#">main.cpp</a>	Główny plik programu . . . . .	15
<a href="#">TSP.cpp</a>	Funkcje odpowiedzialne za rozwiązanie TSP . . . . .	16
<a href="#">TSP.h</a>	Plik nagłówkowy TSP . . . . .	19





## Rozdział 3

# Dokumentacja klas

### 3.1 Dokumentacja struktury Graph

#### Atrybuty publiczne

- `std::vector< char > cities`
- `std::map< std::pair< char, char >, int > map`

Dokumentacja dla tej struktury została wygenerowana z pliku:

- [Graph.h](#)



## Rozdział 4

# Dokumentacja plików

### 4.1 Dokumentacja pliku File.cpp

Funkcje operacji na plikach.

```
#include "File.h"
```

#### Funkcje

- `std::vector< std::string > split (std::string str)`  
*Dzielenie tekstu na fragmenty względem znaku.*
- `Graph loadGraph (std::string graphfile)`  
*Wczytywanie grafu z pliku.*
- `void saveGraph (std::vector< std::pair< std::vector< char >, int > > tsp, std::string filename)`  
*Zapis wyniku do pliku.*

#### 4.1.1 Opis szczegółowy

Funkcje operacji na plikach.

##### Autor

Piotr Nowak ( [nowakpiotr510@gmail.com](mailto:nowakpiotr510@gmail.com) )

##### Wersja

0.1

##### Data

2023-02-06

##### Copyright

Copyright (c) 2023

## 4.1.2 Dokumentacja funkcji

### 4.1.2.1 loadGraph()

```
Graph loadGraph (
    std::string graphfile )
```

Wczytywanie grafu z pliku.

#### Parametry

<i>route</i> <i>file</i>	
<i>graph</i> <i>file</i>	

#### Zwraca

Graph

### 4.1.2.2 saveGraph()

```
void saveGraph (
    std::vector< std::pair< std::vector< char >, int > > tsp,
    std::string filename )
```

Zapis wyniku do pliku.

#### Parametry

<i>tsp</i>	
<i>filename</i>	

### 4.1.2.3 split()

```
std::vector< std::string > split (
    std::string str )
```

Dzielenie tekstu na fragmenty względem znaku.

#### Parametry

<i>str</i>	
------------	--

Zwraca

```
std::vector<std::string>
```

## 4.2 Dokumentacja pliku File.h

Plik nagłówkowy operacji na plikach.

```
#include "TSP.h"
#include <fstream>
#include <string>
#include <sstream>
#include <iostream>
```

### Funkcje

- [Graph loadGraph](#) (std::string graphfile)  
*Wczytywanie grafu z pliku.*
- std::vector< std::string > [split](#) (std::string str)  
*Dzielenie tekstu na fragmenty względem znaku.*
- void [saveGraph](#) (std::vector< std::pair< std::vector< char >, int > > tsp, std::string filename)  
*Zapis wyniku do pliku.*

### 4.2.1 Opis szczegółowy

Plik nagłówkowy operacji na plikach.

Autor

Piotr Nowak ( [nowakpiotr510@gmail.com](mailto:nowakpiotr510@gmail.com) )

Wersja

0.1

Data

2023-02-06

Copyright

Copyright (c) 2023

### 4.2.2 Dokumentacja funkcji

#### 4.2.2.1 loadGraph()

```
Graph loadGraph (
    std::string graphfile )
```

Wczytywanie grafu z pliku.

## Parametry

<i>route</i> <i>file</i>	
<i>graph</i> <i>file</i>	

## Zwraca

[Graph](#)

#### 4.2.2.2 saveGraph()

```
void saveGraph (
    std::vector< std::pair< std::vector< char >, int > > tsp,
    std::string filename )
```

Zapis wyniku do pliku.

## Parametry

<i>tsp</i>	
<i>filename</i>	

#### 4.2.2.3 split()

```
std::vector< std::string > split (
    std::string str )
```

Dzielenie tekstu na fragmenty względem znaku.

## Parametry

<i>str</i>	
------------	--

## Zwraca

`std::vector<std::string>`

## 4.3 File.h

[Idź do dokumentacji tego pliku.](#)

```
00001
00012 #include "TSP.h"
00013 #include <fstream>
```

```
00014 #include <string>
00015 #include <sstream>
00016 #include <iostream>
00017
00018 Graph loadGraph(std::string graphfile);
00019
00020 std::vector<std::string> split(std::string str);
00021
00022 void saveGraph(std::vector<std::pair<std::vector<char>, int> tsp, std::string filename);
```

## 4.4 Dokumentacja pliku Graph.cpp

Funkcje odpowiedzialne za graf.

```
#include "Graph.h"
```

### Funkcje

- `Graph initGraph` (std::vector< char > cities)  
*Inicjalizacja grafu.*
- void `addEdge` (Graph &graph, char from, char to, int distance)  
*Dodawanie krawędzi grafu (pojedyncze połączenia)*

### 4.4.1 Opis szczegółowy

Funkcje odpowiedzialne za graf.

Autor

Piotr Nowak ( [nowakpiotr510@gmail.com](mailto:nowakpiotr510@gmail.com))

Wersja

0.1

Data

2023-02-06

Copyright

Copyright (c) 2023

### 4.4.2 Dokumentacja funkcji

#### 4.4.2.1 addEdge()

```
void addEdge (
    Graph & graph,
    char from,
    char to,
    int distance )
```

Dodawanie krawędzi grafu (pojedyncze połączenia)

**Parametry**

<i>graph</i>	
<i>from</i>	
<i>to</i>	
<i>distance</i>	

**4.4.2.2 initGraph()**

```
Graph initGraph (
    std::vector< char > cities )
```

Inicjalizacja grafu.

**Parametry**

<i>index</i>	
<i>cities</i>	

**Zwraca**

Graph

**4.5 Dokumentacja pliku Graph.h**

Plik nagłówkowy grafu.

```
#include <vector>
#include <map>
#include <iostream>
#include <algorithm>
```

**Komponenty**

- struct [Graph](#)

**Funkcje**

- [Graph initGraph](#) (std::vector< char > cities)  
*Inicjalizacja grafu.*
- void [addEdge](#) ([Graph](#) &graph, char from, char to, int distance)  
*Dodawanie krawędzi grafu (pojedyncze połączenia)*



### 4.5.1 Opis szczegółowy

Plik nagłówkowy grafu.

#### Autor

Piotr Nowak ( [nowakpiotr510@gmail.com](mailto:nowakpiotr510@gmail.com) )

#### Wersja

0.1

#### Data

2023-02-06

#### Copyright

Copyright (c) 2023

### 4.5.2 Dokumentacja funkcji

#### 4.5.2.1 addEdge()

```
void addEdge (
    Graph & graph,
    char from,
    char to,
    int distance )
```

Dodawanie krawędzi grafu (pojedyncze połączenia)

#### Parametry

<i>graph</i>	
<i>from</i>	
<i>to</i>	
<i>distance</i>	

#### 4.5.2.2 initGraph()

```
Graph initGraph (
    std::vector< char > cities )
```

Inicjalizacja grafu.

## Parametry

<i>index</i>	
<i>cities</i>	

## Zwraca

[Graph](#)

## 4.6 Graph.h

[Idź do dokumentacji tego pliku.](#)

```
00001
00011 #include <vector>
00012 #include <map>
00013 #include <iostream>
00014 #include <algorithm>
00015
00016 struct Graph{
00017     std::vector<char> cities;
00018     std::map<std::pair<char, char>, int> map;
00019 };
00020
00021 Graph initGraph(std::vector<char> cities);
00022
00023 void addEdge(Graph &graph, char from, char to, int distance);
```

## 4.7 Dokumentacja pliku main.cpp

Główny plik programu.

#include "File.h"

### Funkcje

- int [main](#) (int argc, char \*\*argv)

*Główna funkcja programu.*

#### 4.7.1 Opis szczegółowy

Główny plik programu.

## Autor

Piotr Nowak ( [nowakpiotr510@gmail.com](mailto:nowakpiotr510@gmail.com))

## Wersja

0.1

## Data

2023-02-06

## Copyright

Copyright (c) 2023

## 4.7.2 Dokumentacja funkcji

### 4.7.2.1 main()

```
int main (
    int argc,
    char ** argv )
```

Główna funkcja programu.

Zwraca

int

## 4.8 Dokumentacja pliku TSP.cpp

Funkcje odpowiedzialne za rozwiązanie TSP.

```
#include "TSP.h"
```

### Funkcje

- int [calcFitness](#) (std::vector< char > route, [Graph](#) graph)  
*Funkcja obliczająca fitness danej trasy.*
- std::vector< char > [generateRoute](#) ([Graph](#) graph)  
*Funkcja generuje losowe ścieżki do populacji.*
- std::vector< char > [crossover](#) (std::vector< char > &parent1, std::vector< char > &parent2)  
*Funkcja odpowiadająca za krzyżowanie.*
- std::pair< std::vector< char >, int > [findBestRoute](#) (std::vector< std::pair< std::vector< char >, int > > population)  
*Funkcja znajdująca najlepszą trasę spośród tras w populacji.*
- void [mutate](#) (std::vector< char > &route)  
*Funkcja odpowiadająca za zachodzenie mutacji.*
- std::vector< std::pair< std::vector< char >, int > > [TSP](#) ([Graph](#) graph, int population\_size, int generations)  
*Główna funkcja rozwiązująca TSP.*

### 4.8.1 Opis szczegółowy

Funkcje odpowiedzialne za rozwiązanie TSP.

Autor

Piotr Nowak ( [nowakpiotr510@gmail.com](mailto:nowakpiotr510@gmail.com) )

Wersja

0.1

Data

2023-02-06

Copyright

Copyright (c) 2023

## 4.8.2 Dokumentacja funkcji

### 4.8.2.1 calcFitness()

```
int calcFitness (
    std::vector< char > route,
    Graph graph )
```

Funkcja obliczająca fitness danej trasy.

#### Parametry

<i>route</i>	
<i>graph</i>	

#### Zwraca

int

### 4.8.2.2 crossover()

```
std::vector< char > crossover (
    std::vector< char > & parent1,
    std::vector< char > & parent2 )
```

Funkcja odpowiadająca za krzyżowanie.

#### Parametry

<i>parent1</i>	
<i>child1</i>	
<i>parent2</i>	

#### Zwraca

std::vector<char>

### 4.8.2.3 findBestRoute()

```
std::pair< std::vector< char >, int > findBestRoute (
    std::vector< std::pair< std::vector< char >, int > > population )
```

Funkcja znajdująca najlepszą trasę spośród tras w populacji.

**Parametry**

<i>population</i>	
-------------------	--

**Zwraca**

`std::pair<std::vector<char>, int>`

**4.8.2.4 generateRoute()**

```
std::vector< char > generateRoute (
    Graph graph )
```

Funkcja generuje losowe ścieżki do populacji.

**Parametry**

<i>graph</i>	
--------------	--

**Zwraca**

`std::vector<char>`

**4.8.2.5 mutate()**

```
void mutate (
    std::vector< char > & route )
```

Funkcja odpowiadająca za zachodzenie mutacji.

**Parametry**

<i>route</i>	
--------------	--

**4.8.2.6 TSP()**

```
std::vector< std::pair< std::vector< char >, int > > TSP (
    Graph graph,
    int population_size,
    int generations )
```

Główna funkcja rozwiązująca TSP.

## Parametry

<i>graph</i>	
<i>population_size</i>	
<i>generations</i>	

## Zwraca

`std::vector<std::pair<std::vector<char>, int>>>`

## 4.9 Dokumentacja pliku TSP.h

Plik nagłówkowy TSP.

```
#include <vector>
#include <set>
#include "Graph.h"
```

## Funkcje

- `std::vector< std::pair< std::vector< char >, int > >` [TSP](#) ([Graph](#) graph, int population\_size, int generations)  
*Główna funkcja rozwiązująca TSP.*
- `std::vector< char >` [crossover](#) (std::vector< char > &parent1, std::vector< char > &parent2)  
*Funkcja odpowiadająca za krzyżowanie.*
- `std::vector< char >` [generateRoute](#) ([Graph](#) graph)  
*Funkcja generuje losowe ścieżki do populacji.*
- int [calcFitness](#) (std::vector< char > route, [Graph](#) graph)  
*Funkcja obliczająca fitness danej trasy.*
- void [mutate](#) (std::vector< char > &route)  
*Funkcja odpowiadająca za zachodzenie mutacji.*
- `std::pair< std::vector< char >, int >` [findBestRoute](#) (std::vector< std::pair< std::vector< char >, int > > population)  
*Funkcja znajdująca najlepszą trasę spośród tras w populacji.*

### 4.9.1 Opis szczegółowy

Plik nagłówkowy TSP.

## Autor

Piotr Nowak ( [nowakpiotr510@gmail.com](mailto:nowakpiotr510@gmail.com) )

## Wersja

0.1

## Data

2023-02-06

## Copyright

Copyright (c) 2023

## 4.9.2 Dokumentacja funkcji

### 4.9.2.1 calcFitness()

```
int calcFitness (
    std::vector< char > route,
    Graph graph )
```

Funkcja obliczająca fitness danej trasy.

#### Parametry

<i>route</i>	
<i>graph</i>	

#### Zwraca

int

### 4.9.2.2 crossover()

```
std::vector< char > crossover (
    std::vector< char > & parent1,
    std::vector< char > & parent2 )
```

Funkcja odpowiadająca za krzyżowanie.

#### Parametry

<i>parent1</i>	
<i>child1</i>	
<i>parent2</i>	

#### Zwraca

std::vector<char>

### 4.9.2.3 findBestRoute()

```
std::pair< std::vector< char >, int > findBestRoute (
    std::vector< std::pair< std::vector< char >, int > > population )
```

Funkcja znajdująca najlepszą trasę spośród tras w populacji.



## Parametry

<i>population</i>	
-------------------	--

## Zwraca

`std::pair<std::vector<char>, int>`**4.9.2.4 generateRoute()**

```
std::vector< char > generateRoute (
    Graph graph )
```

Funkcja generuje losowe ścieżki do populacji.

## Parametry

<i>graph</i>	
--------------	--

## Zwraca

`std::vector<char>`**4.9.2.5 mutate()**

```
void mutate (
    std::vector< char > & route )
```

Funkcja odpowiadająca za zachodzenie mutacji.

## Parametry

<i>route</i>	
--------------	--

**4.9.2.6 TSP()**

```
std::vector< std::pair< std::vector< char >, int > > TSP (
    Graph graph,
    int population_size,
    int generations )
```

Główna funkcja rozwiązująca TSP.

**Parametry**

<i>graph</i>	
<i>population_size</i>	
<i>generations</i>	

**Zwraca**

`std::vector<std::pair<std::vector<char>, int>>>`

## 4.10 TSP.h

[Idź do dokumentacji tego pliku.](#)

```
00001
00012 #include <vector>
00013 #include <set>
00014 #include "Graph.h"
00015
00016 std::vector<std::pair<std::vector<char>, int>> TSP(Graph graph, int population_size, int generations);
00017
00018 std::vector<char> crossover(std::vector<char> &parent1, std::vector<char> &parent2);
00019
00020 std::vector<char> generateRoute(Graph graph);
00021
00022 int calcFitness(std::vector<char> route, Graph graph);
00023
00024 void mutate(std::vector<char> &route);
00025
00026 std::pair<std::vector<char>, int> findBestRoute(std::vector<std::pair<std::vector<char>, int>>
    population);
```

# Skorowidz

- addEdge
  - Graph.cpp, [11](#)
  - Graph.h, [13](#)
- calcFitness
  - TSP.cpp, [17](#)
  - TSP.h, [20](#)
- crossover
  - TSP.cpp, [17](#)
  - TSP.h, [20](#)
- File.cpp, [7](#)
  - loadGraph, [8](#)
  - saveGraph, [8](#)
  - split, [8](#)
- File.h, [9](#)
  - loadGraph, [9](#)
  - saveGraph, [10](#)
  - split, [10](#)
- findBestRoute
  - TSP.cpp, [17](#)
  - TSP.h, [20](#)
- generateRoute
  - TSP.cpp, [18](#)
  - TSP.h, [21](#)
- Graph, [5](#)
- Graph.cpp, [11](#)
  - addEdge, [11](#)
  - initGraph, [12](#)
- Graph.h, [12](#)
  - addEdge, [13](#)
  - initGraph, [13](#)
- initGraph
  - Graph.cpp, [12](#)
  - Graph.h, [13](#)
- loadGraph
  - File.cpp, [8](#)
  - File.h, [9](#)
- main
  - main.cpp, [16](#)
- main.cpp, [15](#)
  - main, [16](#)
- mutate
  - TSP.cpp, [18](#)
  - TSP.h, [21](#)
- saveGraph
  - File.cpp, [8](#)
  - File.h, [10](#)
- split
  - File.cpp, [8](#)
  - File.h, [10](#)
- TSP
  - TSP.cpp, [18](#)
  - TSP.h, [21](#)
- TSP.cpp, [16](#)
  - calcFitness, [17](#)
  - crossover, [17](#)
  - findBestRoute, [17](#)
  - generateRoute, [18](#)
  - mutate, [18](#)
  - TSP, [18](#)
- TSP.h, [19](#)
  - calcFitness, [20](#)
  - crossover, [20](#)
  - findBestRoute, [20](#)
  - generateRoute, [21](#)
  - mutate, [21](#)
  - TSP, [21](#)