Uniwersytet Jagielloński w Krakowie
Wydział Fizyki, Astronomii i Informatyki Stosowanej

Łukasz Kostrzewa
Nr albumu: 1080514

Wizualizacja, edycja i przetwarzanie grafów on-line

Praca magisterska
na kierunku Informatyka stosowana

Praca wykonana pod kierunkiem
dr hab. Barbary Strug
Zakład Projektowania i Grafiki Komputerowej

Kraków 2017

Oświadczenie autora pracy

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

Kraków, dnia Podpis autora pracy

Oświadczenie kierującego pracą

Potwierdzam, że niniejsza praca została przygotowana pod moim kierunkiem i kwalifikuje się do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

Kraków, dnia Podpis kierującego pracą

Spis treści

Wstęp

„This question is so banal, but seemed to me worthy of attention *in that geometry, nor algebra, nor even the art of counting was sufficient to solve it*[1] *”*. Tak w 1736 roku pisał Leonhard Euler w liście do Giovanniego Marinoniego, włoskiego matematyka i inżyniera, o jednym z pierwszych problemów w teorii grafów – problemie mostów królewskich. Banalny, ale warty uwagi.

W dzisiejszych czasach teoria grafów rozwiązuje wiele nietrywialnych problemów, a część z nich nadal pozostaje otwarta. Grafy znalazły praktyczne zastosowanie w wielu różnorodnych dziedzinach nauki, takich jak informatyka, ekonomia, socjologia, jak również chemia, lingwistyka, geografia czy nawet architektura. Bez wątpienia teoria grafów jest dziedziną matematyki i informatyki, która zasługuje na uwagę, co postaram się w niniejszej pracy przedstawić.

Głównym celem mojej pracy jest stworzenie aplikacji służącej do wizualizacji i edycji grafów w przeglądarce. W przeciągu kilku ostatnich lat mogliśmy zaobserwować gwałtowny wzrost znaczenia aplikacji internetowych. Co dziwne, na dzień dzisiejszy w sieci praktycznie nie ma rozwiązania, które pozwalałoby wczytać graf, wyświetlić, w łatwy sposób przetworzyć, a następnie wyeksportować do znanego formatu. Praca ta jest odpowiedzią na ów deficyt.

W pracy dokonam również przeglądu i analizy bibliotek JavaScript oraz technologii służących do wizualizacji grafów w przeglądarce.

[1] Cytat zaczerpnięty z [HW04], wyróżnienie własne.

Rozdział 1

Wprowadzenie

W tym rozdziale omówię czym są grafy – na początku przedstawię intuicyjne wyjaśnienie, po czym podam formalną definicję. Pojawią się także definicje pojęć związanych z grafami, które będą występować w kolejnych rozdziałach pracy. Następnie przytoczę przykłady znanych grafów, takich jak graf pełny czy graf cykliczny. W ostatniej sekcji przedstawię zastosowania grafów.

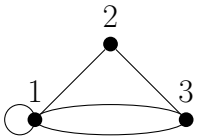
1.1 Czym są grafy?

1.2 Definicje

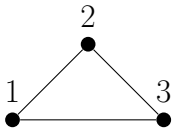
Graf ogólny, graf prosty

Graf (**graf ogólny**, **multigraf**) G jest parą $(V(G), E(G))$, gdzie $V(G)$ jest skończonym, niepustym zbiorem elementów zwanych **wierzchołkami**, a $E(G)$ jest skończoną rodziną nieuporządkowanych par elementów zbioru $V(G)$ zwanych **krawędziami**[1] (tj. $E \subseteq \{\{u, v\} : u, v \in V(G)\}$). Zbiór $V(G)$ nazywamy zbiorem wierzchołków, a rodzinę $E(G)$ – **rodziną krawędzi** grafu G; gdy nie ma możliwości pomyłki często są skracane do odpowiednio V oraz E. (Niektóre definicje nie wymagają, aby zbiory V oraz E były skończone[2], ale ponieważ w naszych zastosowaniach będziemy mieli do czynienia ze zbiorami skończonymi, przyjmiemy, że zbiory te są skończone). Wierzchołki $u, v \in V$ są **połączone** krawędzią $\{u, v\}$ (lub krócej uv), gdy $\{u, v\} \in E$.

[1] Wil07, s. 20.
[2] RW08, s. 143.



Rysunek 1.1: Przykład grafu ogólnego



Rysunek 1.2: Przykład grafu prostego

Zauważmy, że taka definicja dopuszcza sytuację, w której dwa wierzchołki są połączone więcej niż jedną krawędzią (tzw. **krawędź wielokrotna**) oraz że wierzchołek jest połączony z samym sobą (tzw. **pętla**). Graf, który nie posiada krawędzi wielokrotnych oraz pętli nazywamy **grafem prostym**[3].

Sąsiedztwo

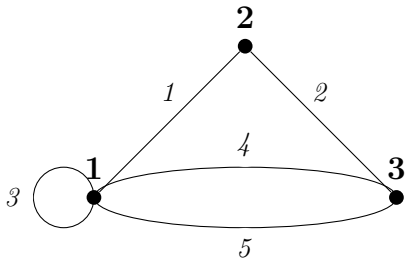
Wierzchołki $u, v \in V$ są **sąsiednie** jeśli istnieje krawędź uv (wierzchołki u i v są **incydentne** z tą krawędzią). Dwie krawędzie są **sąsiednie**, jeśli są incydentne z tym samym wierzchołkiem.

Stopień wierzchołka $v \in V$ jest liczbą krawędzi incydentnych z v. **Wierzchołek izolowany** to wierzchołek stopnia 0, a **wierzchołek końcowy** – stopnia 1.

Istnieją dwie standardowe reprezentacje grafów w pamięci komputera: jako **listy sąsiedztwa** lub jako **macierze sąsiedztwa**[4]. Pierwsza z nich polega na zapamiętaniu dla każdego wierzchołka listy wierzchołków z nim sąsiadujących. Druga zakłada, że wierzchołki są ponumerowane liczbami ze zbioru $\{1, 2, \ldots, n\}$ (gdzie n oznacza moc zbioru V) i opiera się na stworzeniu macierzy wymiaru $n \times n$, której wyraz o indeksach i, j jest równy liczbie krawędzi łączących wierzchołek o numerze i z wierzchołkiem o numerze j.

Innym sposobem reprezentacji grafu za pomocą macierzy jest **macierz incydencji**. Jeśli krawędzie oznakujemy liczbami ze zbioru $\{1, 2, \ldots, m\}$ (gdzie m moc zbioru E), to jest to macierz o rozmiarze $n \times m$, której wyraz o indeksach i, j jest równy 1, jeśli wierzchołek z numerem i jest incydentny z krawędzią j, i równy 0 w przeciwnym przypadku.

[3] Wil07, s. 19.
[4] BDR99, s. 29.



Rysunek 1.3

Macierz sąsiedztwa A i macierz incydencji M dla grafu z rysunku 1.3:

$$A = \begin{pmatrix} 1 & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{pmatrix}, \qquad M = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

Spójność

1.3 Przykłady grafów

1.4 Zastosowania grafów

Rozdział 2

Wymagania

Rozdział ten zawiera wszystkie wymagania funkcjonalne, które powinna spełniać aplikacja, aby praca z grafami była możliwie przystępna i intuicyjna.

2.1 Tworzenie grafów

Podstawowym i oczywistym wymaganiem jest, aby użytkownik mógł stworzyć nowy, pusty graf skierowany oraz nieskierowany. Ponadto użytkownik powinien mieć możliwość zaimportowania istniejącego grafu oraz wygenerowania znanego grafu, np. cyklu lub grafu pełnego o zadanej ilości wierzchołków.

2.1.1 Importowanie grafów

Użytkownik powinien móc wczytać graf z komputera lub z chmury (np. Google Drive lub Dropbox) w trzech znanych formatach:

- `GraphML`,
- `GEXF`,
- `JGF`.

Opisy formatów znajdują się w sekcji 4.1.

2.1.2 Generowanie grafów

Użytkownik powinien mieć możliwość wygenerowania znanych grafów, dla zadanych parametrów wejściowych:

- grafu pustego,

- grafu liniowego,
- grafu cyklicznego,
- koła,
- grafu pełnego (lub turnieju dla grafów skierowanych),
- grafu pełnego dwudzielnego,
- grafu Petersena,
- drzewa (o zadanej wysokości i ilości dzieci)

Definicje i przykłady powyższych grafów znajdują się w sekcji 1.3.

Ponadto przydatnym dodatkiem w aplikacji będzie możliwość wygenerowania grafu losowego – o danej ilości wierzchołków oraz parametrem prawdopodobieństwa określającym, czy pomiędzy dwoma wierzchołkami istnieje krawędź.

2.2 Wizualizacja

Użytkownik powinien móc przesuwać widok, przybliżać i oddalać graf oraz rozmieszczać wierzchołki grafu w dowolny sposób. W aplikacji powinna istnieć możliwość zmiany układu grafu: układ oparty na oddziaływaniach (ang. *force-based layout*), układ siatki, układ okręgu, układ koncentryczny, układ hierarchiczny.

Użytkownik powinien być w stanie zmienić kategorię wierzchołka oraz typ krawędzi. Inne typy i kategorie powinny być oznaczone innym kolorem oraz powinna istnieć możliwość zmiany koloru.

Aplikacja powinna również dostarczać opcję wyszukiwania i filtrowania danych (np. tylko dany typ wierzchołków, wierzchołki o stopniu większym niż zadany parametr). Przydatną funkcjonalnością będzie wyświetlanie sąsiadów danego wierzchołka po najechaniu na niego kursorem myszy.

2.3 Edycja

W aplikacji powinien istnieć osobny tryb edycji. Gdy użytkownik jest w tym trybie, powinien móc dodawać oraz usuwać wierzchołki i krawędzie. Powinien być w stanie także dodawać oraz modyfikować etykiety wierzchołków i krawędzi.

Użytkownik powinien mieć możliwość zaznaczania wielu wierzchołków i krawędzi na raz. Użyteczną funkcjonalnością będzie również grupowanie (lub rozgrupowanie) zaznaczonych wierzchołków.

Aplikacja powinna wyświetlać ostatnio wykonaną akcję oraz udostępniać możliwość jej cofnięcia.

2.4 Przetwarzanie

Aplikacja powinna dawać możliwość wykonania podstawowych algorytmów na danym grafie:

- wyszukiwanie najkrótszej ścieżki pomiędzy dwoma wybranymi wierzchołkami,
- znajdowanie minimalnego drzewa rozpinającego,
- obliczanie algorytmu PageRank,
- znajdowanie (silnie) spójnych składowych oraz dwuspójnych składowych,
- znajdowanie cyklu Eulera,
- znajdowanie cyklu Hamiltona.

2.5 Eksportowanie

Użytkownik powinien mieć możliwość wyeksportowania do formatów, które zostały przedstawione w podsekcji 2.1.1.

Ponadto przydatną funkcjonalnością będzie możliwość wyeksportowania obecnego widoku do pliku graficznego, np. `PNG` lub `JPG`.

2.6 Udostępnianie grafu

W aplikacji powinna istnieć możliwość udostępniania grafu innym użytkownikom. Po wybraniu tej opcji, powinien zostać wygenerowany unikalny odnośnik do grafu. Po przejściu na ten adres (w podstawowej wersji) inni użytkownicy mogą wyświetlić i edytować graf.

Rozdział 3

Istniejące rozwiązania

W tym rozdziale przedstawię istniejące aplikacje internetowe [Sta] i desktopowe służące do tworzenia i wizualizacji grafów. Tabela 3.1 zawiera porównanie funkcjonalności opisywanych aplikacji internetowych.

3.1 Aplikacje internetowe

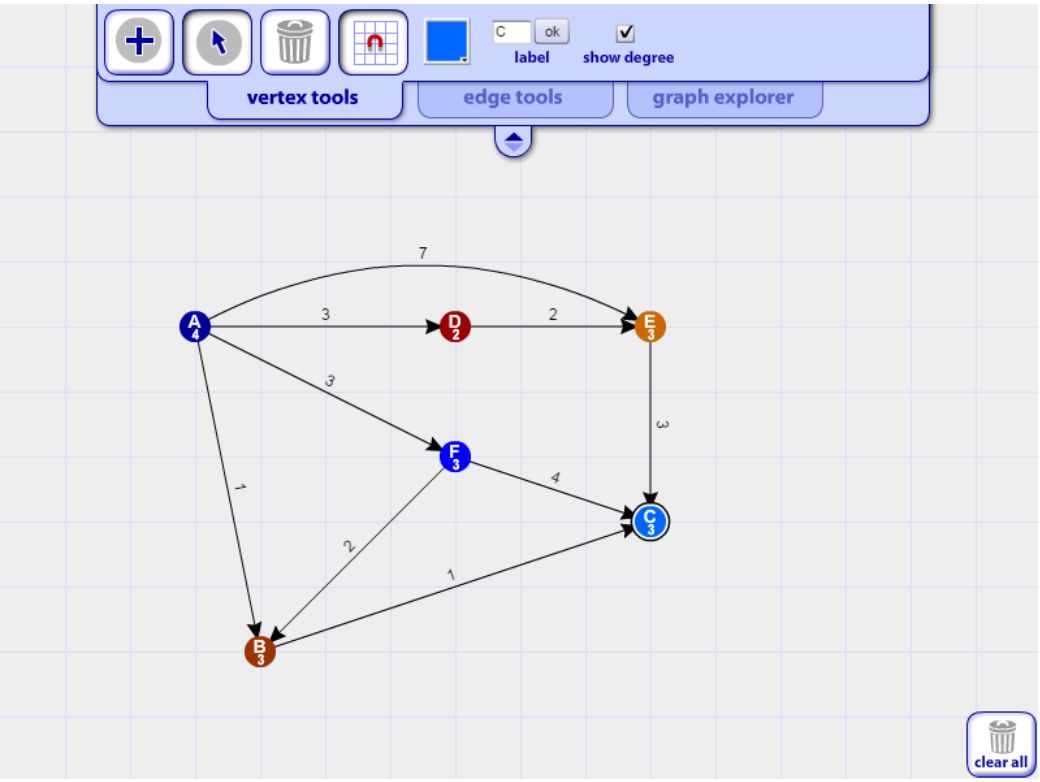
Graph Creator

Adres URL	http://illuminations.nctm.org/Activity.aspx?id=3550
Autor	National Council of Teachers of Mathematics

Aplikacja pozwala tworzyć grafy skierowane i nieskierowane. Posiada możliwość kolorowania wierzchołków, wyrównania ich do siatki oraz ustawienia wag na krawędziach i etykiet w wierzchołkach. Ponadto użytkownik może wyświetlić stopnie wierzchołków oraz wyginać krawędzie. Dodatkową funkcjonalnością jest możliwość zaznaczenia kilku wierzchołków na raz.

Graph Creator nie daje możliwości eksportowania i importowania grafów. Nie można również przesuwać widoku ani oddalać oraz przybliżać grafu. Aplikacja posiada ograniczenie liczby wierzchołków – maksymalna dozwolona ilość to 52 wierzchołki.

Rysunek 3.1: Zrzut ekranu z aplikacji Graph Creator



Graph Online

Adres URL	http://graphonline.ru/en/
Autor	Unick-soft

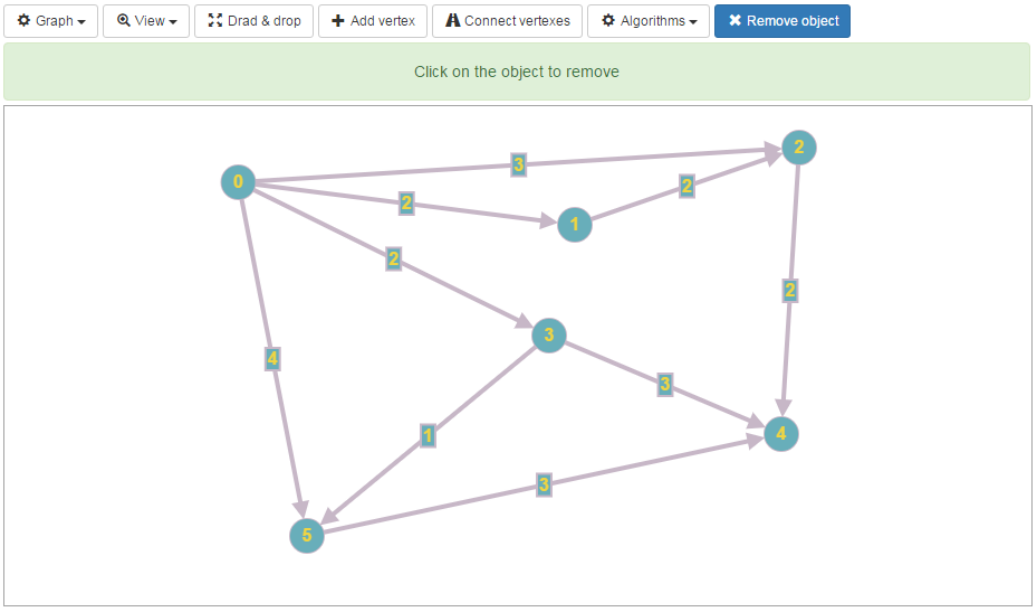
Aplikacja również daje możliwość stworzenia grafów zarówno skierowanych jak i nieskierowanych. Podobnie jak poprzednia aplikacja pozwala na zmianę etykiet wierzchołków, nadanie wag krawędziom oraz na wyświetlenie stopnia wierzchołków. Ponadto użytkownik ma możliwość przesuwania widoku oraz jego przybliżania i oddalania. Dodatkowo *Graph Online* pozwala zapisać graf jako macierz sąsiedztwa lub incydencji oraz wczytać graf zapisany w takiej postaci. Użytkownik może również zapisać graf na serwerze – po zapisaniu wyświetlany jest ogólnodostępny adres URL do grafu. Ciekawą funkcjonalnością jest eksport grafu do obrazka (plik `PNG`).

Graph Online posiada możliwość wykonania podstawowych algorytmów na grafie, takich jak: znajdowanie najkrótszej ścieżki pomiędzy dwoma wierz-

chołkami, znajdowanie cyklu Eulera, znajdowanie spójnych składowych, znajdowanie minimalnego drzewa rozpinającego.

W przeciwieństwie do poprzedniej aplikacji nie mamy możliwości kolorowania wierzchołków, zaznaczania kilku wierzchołków na raz oraz wyginania krawędzi. Maksymalna dozwolona ilość wierzchołków to 299.

Rysunek 3.2: Zrzut ekranu z aplikacji Graph Online



GraphJS

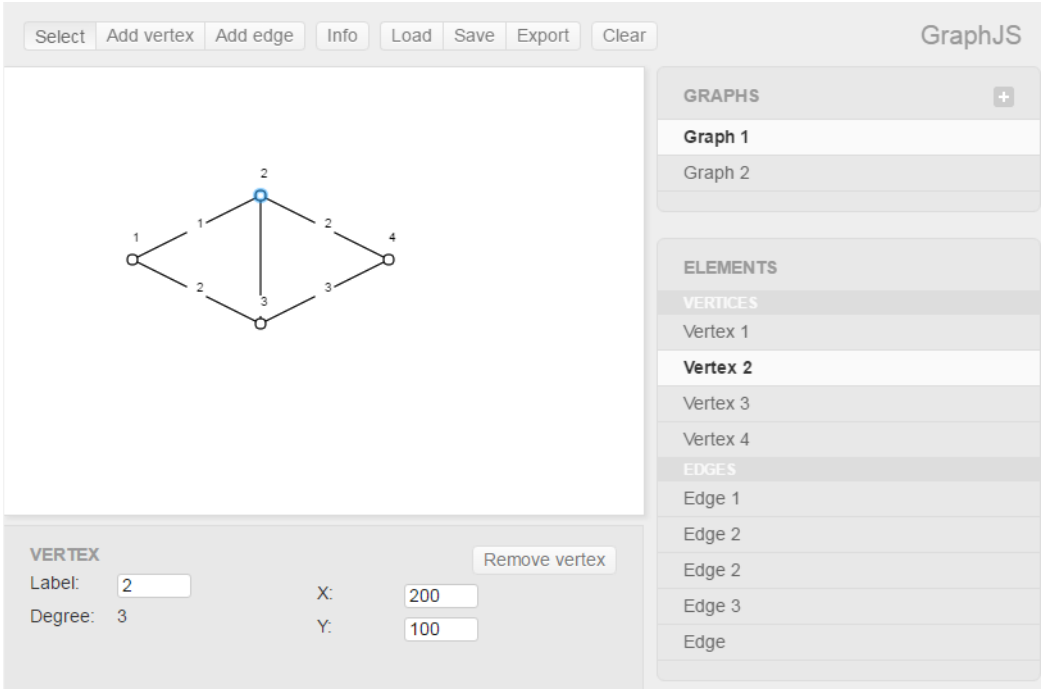
Adres URL	https://dl.dropboxusercontent.com/u/4189520/GraphJS/graphjs.html
Autor	David Kofoed Wind

Aplikacja pozwala na tworzenie grafów nieskierowanych. Podobnie jak w poprzednich aplikacjach możemy nadawać etykiety wierzchołkom i krawędziom. Niespotykaną funkcjonalnością jest możliwość stworzenia kilku grafów i przełączania się pomiędzy nimi oraz możliwość eksportu grafu do formatu LaTeX(pakiet Ti*k*Z). Ponadto użytkownik ma możliwość eksportu do własnego formatu `JSON` oraz importu grafu z tego formatu. Aplikacja posiada funkcjonalność zaznaczania wielu wierzchołków na raz.

W *GraphJS* nie ma możliwości przesuwania widoku oraz przybliżania i oddalania grafu. Nie ma również możliwości kolorowania wierzchołków oraz

wyginania krawędzi. Aplikacja zdaje się nie mieć limitu na liczbę wierzchołków – udało się wczytać graf C_{1000} jednakże dodanie kolejnego wierzchołka zajmuje około 10 sekund.

Rysunek 3.3: Zrzut ekranu z aplikacji GraphJS



Graphrel

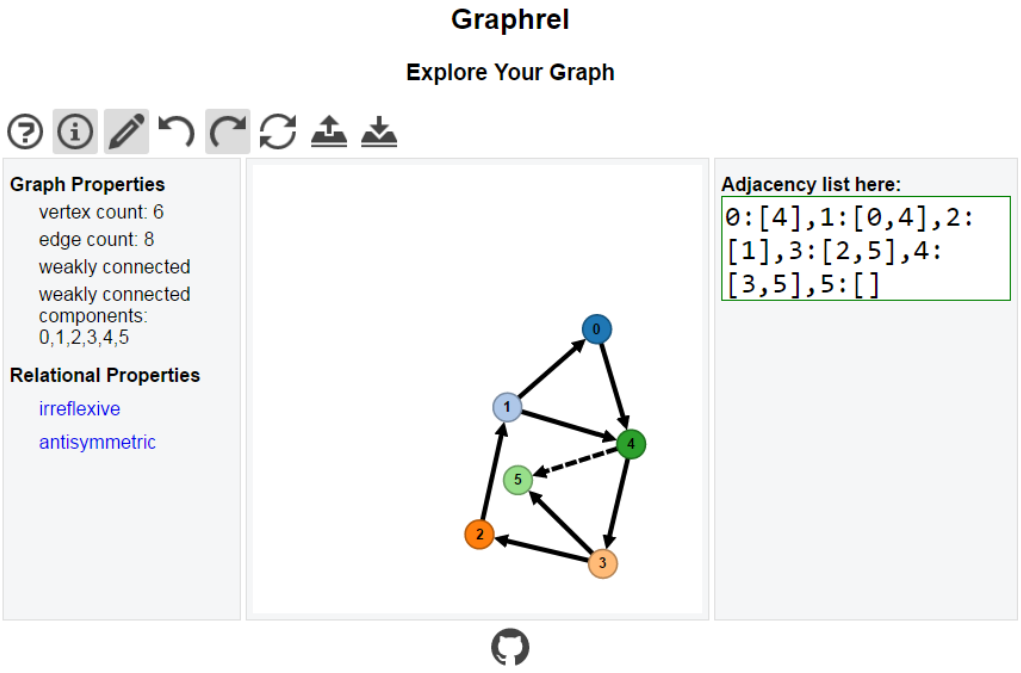
Adres URL	https://yiboyang.github.io/graphrel/
Autor	Yibo Yang

Aplikacja daje możliwość tworzenia grafów skierowanych. W przeciwieństwie do poprzednio opisywanych aplikacji posiada układ kierowany siłą (ang. *force-directed layout*), choć istnieje również opcja samodzielnego rozstawienia wierzchołków – poprzez przytrzymanie klawisza `Ctrl`. Użytkownik może zaimportować graf z formatu stworzonego przez aplikację (tablice list sąsiedztwa dla każdego wierzchołka). Bardzo przydatną i niespotykaną funkcjonalnością jest możliwość cofania oraz ponawiania ostatnich akcji.

W *Graphrel* nie możemy nadawać własnych etykiet na krawędziach ani w wierzchołkach, nie możemy przesuwać widoku ani zmieniać przybliżenia grafu. Nie ma również możliwości wyginania krawędzi, zaznaczania kliku

wierzchołków na raz oraz kolorowania wierzchołków. Do aplikacji udało się wczytać graf C_{100}, przy próbie wczytania C_{101} pojawia się informacja o niepoprawnym formacie.

Rysunek 3.4: Zrzut ekranu z aplikacji Graphrel

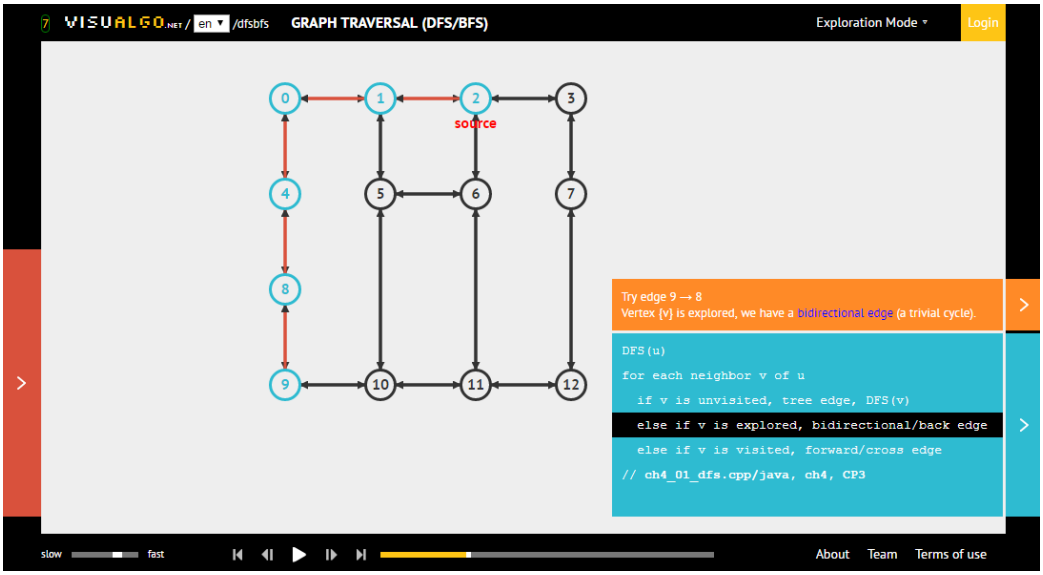


VisuAlgo

Adres URL	https://visualgo.net/en/
Autor	Dr Steven Halim

Aplikacja stworzona przez Dr Stevena Halima z National University of Singapore. Posiada możliwość tworzenia prostych grafów, jednak jej głównym celem jest wizualizacja algorytmów przez animację (nie tylko na grafach, ale również na strukturach danych). Użytkownik wraz z przebiegiem algorytmu może obserwować przebieg kodu, może zatrzymać się w dowolnym jego kroku, cofnąć się do kroku poprzedniego albo przejść do następnego.

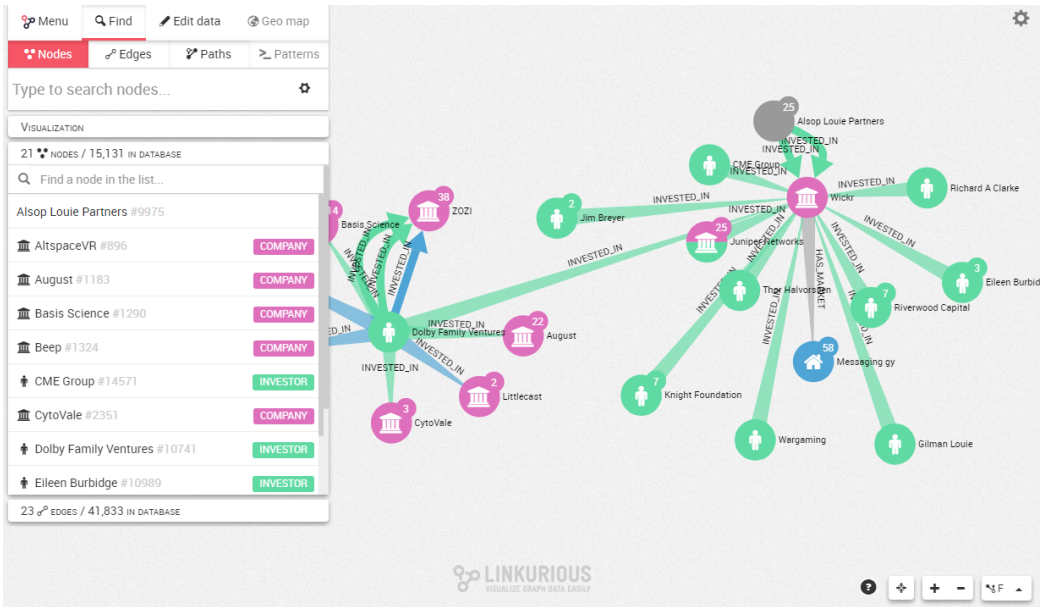
Rysunek 3.5: Zrzut ekranu z aplikacji VisuAlgo



Linkurious

Adres URL	http://linkurio.us
Autor	Linkurious

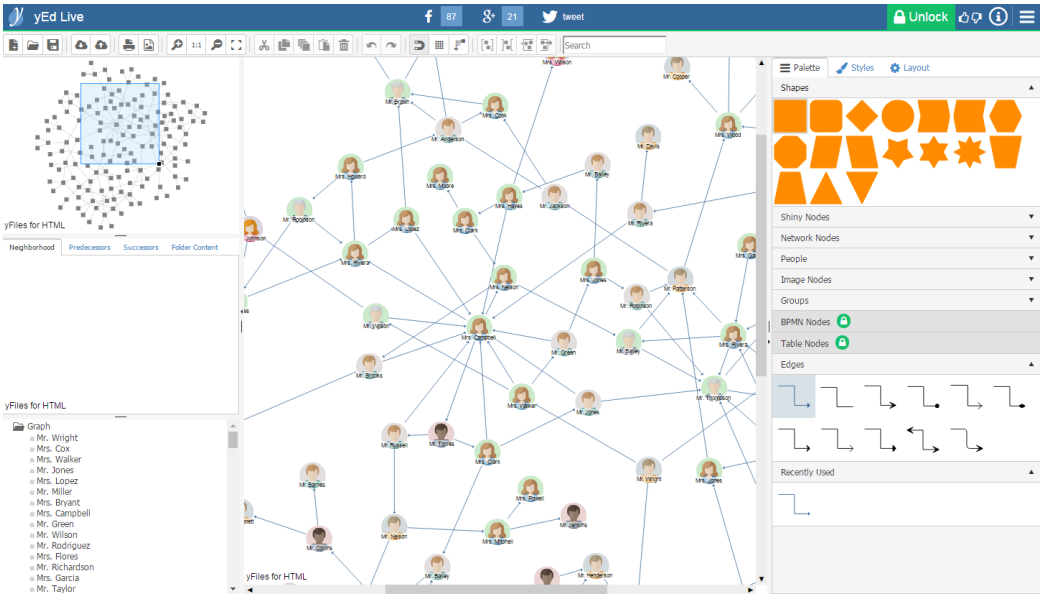
Rysunek 3.6: Zrzut ekranu z aplikacji Linkurious



yEd Live

Adres URL	https://www.yworks.com/yed-live/
Autor	yWorks

Rysunek 3.7: Zrzut ekranu z aplikacji yEd Live



Tablica 3.1: Porównanie aplikacji *Graph Creator*, *Graph Online*, *GraphJS* i *Graphrel*

	Graph Creator	Graph Online	GraphJS	Graphrel
graf nieskierowany	✓	✓	✓	–
graf skierowany	✓	✓	–	✓
etykiety na krawędziach	✓	✓	✓	–
etykiety w wierzchołkach	✓	✓	✓	–
kolorowanie wierzchołków	✓	–	–	–
wyginanie krawędzi	✓	–	–	–
zaznaczanie kilku wierzchołków	✓	–	✓	–
przesuwanie widoku	–	✓	–	–
przybliżanie/oddalanie	–	✓	–	–
zapisywanie/wczytywanie	–	✓[1]	✓[2]	✓[3]

[1] jako macierz sąsiedztwa lub jako obrazek
[2] własny format `JSON` lub jako LaTeX
[3] własny format (listy sąsiedztwa)

3.2 Aplikacje desktopowe

Gephi

https://gephi.org/

GraphTea

http://www.graphtheorysoftware.com/

Cytoscape

http://www.cytoscape.org/

yEd Graph Editor

https://www.yworks.com

Rozdział 4

Analiza

4.1 Formaty zapisu grafów

Istnieje wiele formatów służących do opisu grafów. Do najpopularniejszych należą [MB04; Gep]

- `GraphML` – *Graph Markup Language*
- `GEXF` – *Graph Exchange XML Format*
- `JGF` – *JSON Graph Format*
- `DOT` – format programu Graphviz
- `GML` – *Graph Modeling Language*
- `DGML` – *Directed Graph Markup Language*
- `XGMML` – *eXtensible Graph Markup and Modeling Language*

Graph Markup Language (GraphML)

Listing 4.1: Przykład grafu w formacie GraphML

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
     http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
  <graph id="G" edgedefault="undirected">
    <node id="1"/>
    <node id="2"/>
    <edge source="1" target="2"/>
  </graph>
</graphml>
```

Graph Exchange XML Format (GEXF)

Listing 4.2: Przykład grafu w formacie GEXF

```
<?xml version="1.0" encoding="UTF-8"?>
<gexf xmlns="http://www.gexf.net/1.2draft" version="1.2">
    <graph mode="static" defaultedgetype="directed">
        <nodes>
            <node id="0" label="Hello" />
            <node id="1" label="Word" />
        </nodes>
        <edges>
            <edge id="0" source="0" target="1" />
        </edges>
    </graph>
</gexf>
```

JSON Graph Format (JGF)

Listing 4.3: Przykład grafu w formacie JGF

```
{
        "graph": {
                "nodes": [{
                        "id": "A",

                },
                {
                        "id": "B",

                }],
                "edges": [{
                        "source": "A",
                        "target": "B"
                }]
        }
}
```

DOT Graphviz

Listing 4.4: Przykład grafu w formacie DOT

```
graph graphname {
     a -- b -- c;
     b -- d;
}
```

4.2 Biblioteki do wizualizacji grafów w JavaScript

	Cytoscape.js	Sigma	VivaGraphJS
Licencja	MIT	MIT	BSD 3
Rozmiar	294	112,9	60,4
Renderowanie			
SVG	•	tak	•
HTML5 Canvas	•	tak	•
WebGL Canvas	•	tak	•
Obsługiwane formaty	•	•	•
Rozszerzalność	•	•	•
•	•	•	•

4.2.1 Cytoscape.js

4.2.2 sigma.js

4.2.3 VivaGraph.js

4.2.4 Linkurious.js

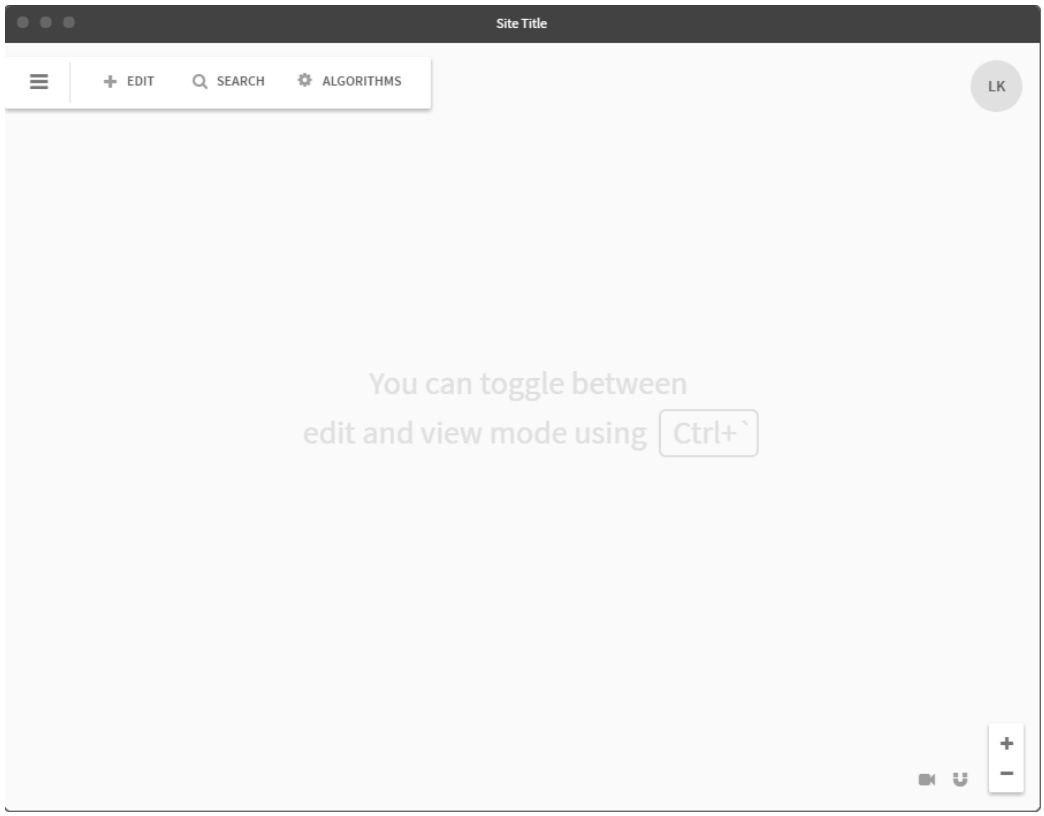
4.3 Grafowe bazy danych

Rozdział 5

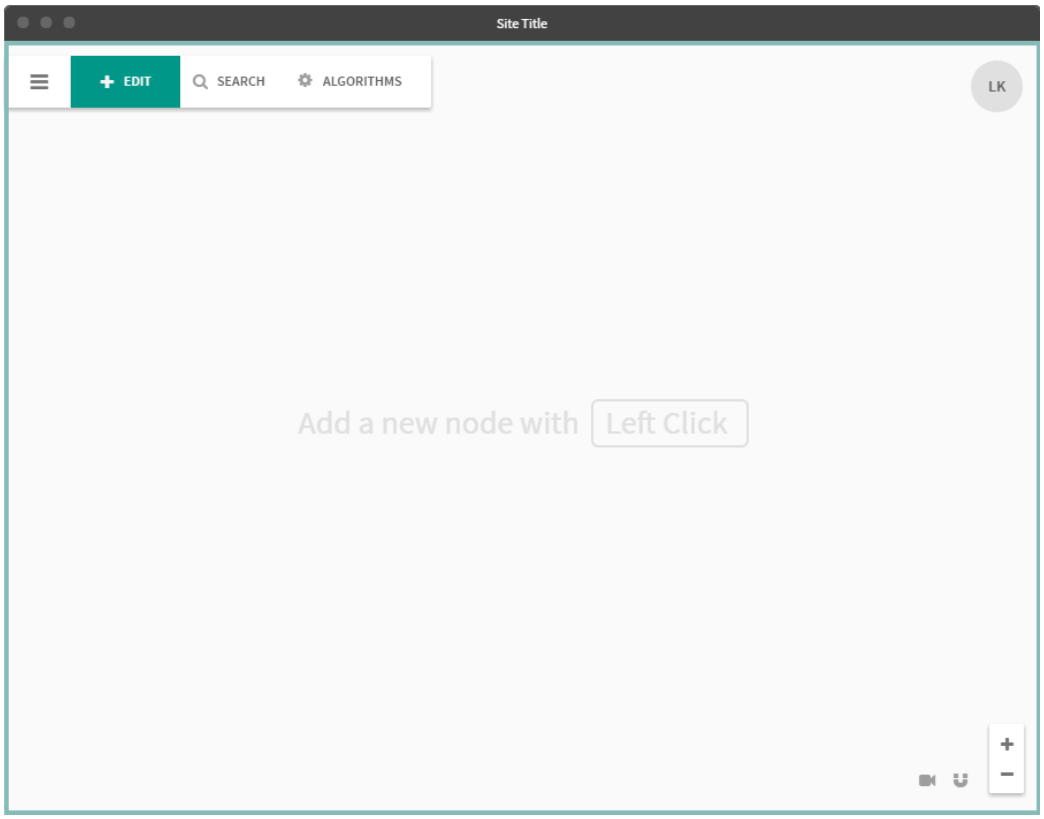
Projekt

5.1 Interfejs użytkownika

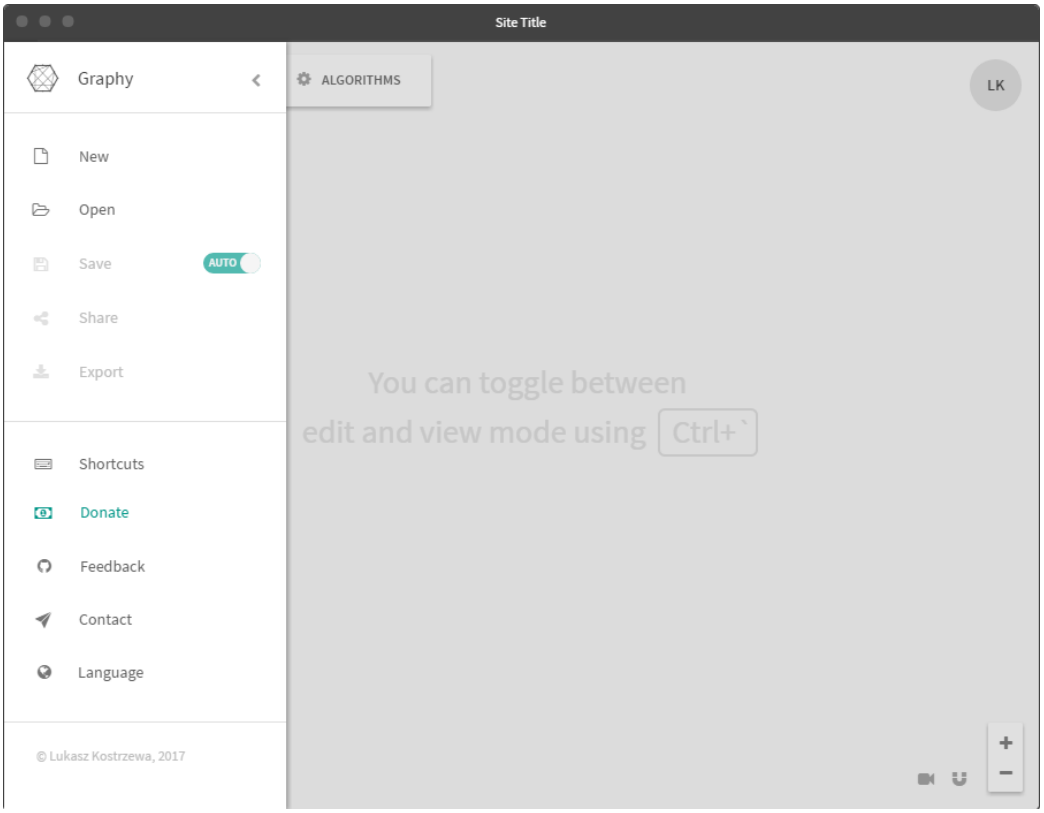
Rysunek 5.1: Tryb widoku grafu



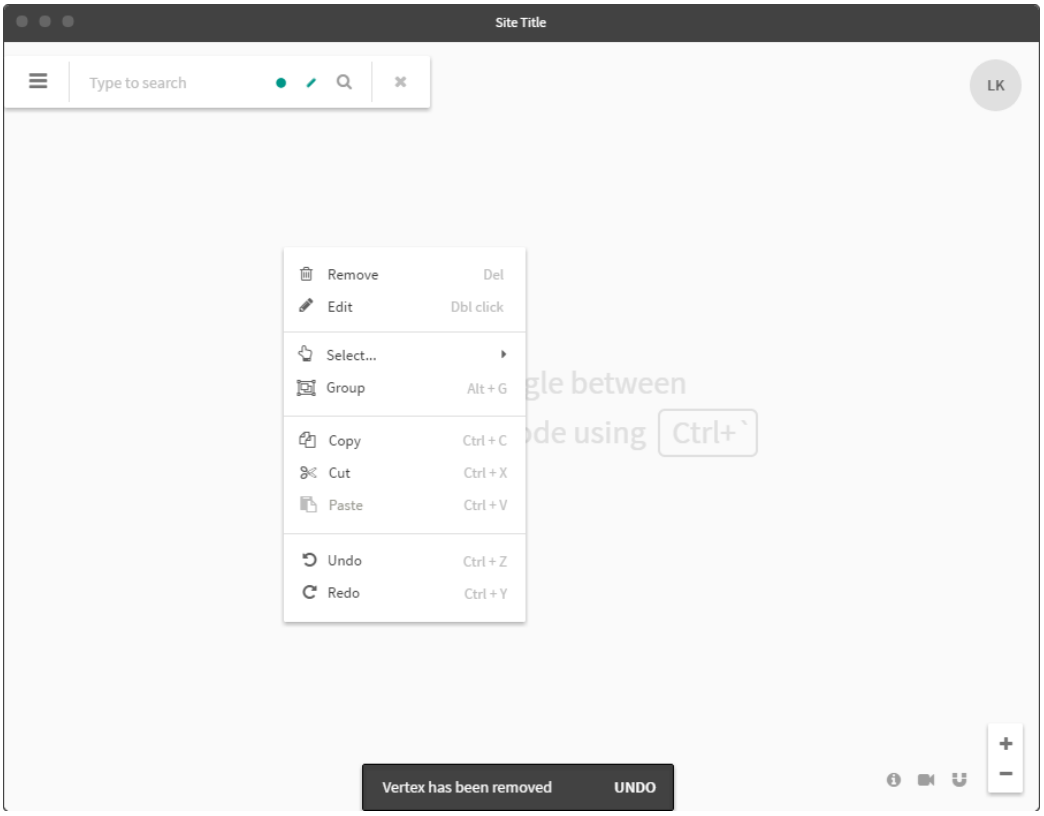
Rysunek 5.2: Tryb edycji grafu



Rysunek 5.3: Widok menu



Rysunek 5.4: Menu kontekstowe i informacja o ostatniej akcji



Rozdział 6

Implementacja

Rozdział 7

Testy

Rozdział 8

Wnioski

Dodatek A

Instrukcje dla użytkowników

Dodatek B

Instrukcje dla programistów

Dodatek C

Użyte narzędzia

Bibliografia

[BDR99] Lech Banachowski, Krzysztof Diks i Wojciech Rytter. *Algorytmy i struktury danych*. 2 wyd. Wydawnictwa Naukowo-Technicznе, 1999. ISBN: 83-204-2403-8.

[HW04] Brian Hopkins i Robin Wilson. *„The Truth about Königsberg"*. W: *College Mathematics Journal* 35 (maj 2004), s. 198–207. URL: `https://www.maa.org/sites/default/files/pdf/upload_library/22/Polya/hopkins.pdf` (term. wiz. 29. 04. 2017).

[MB04] S. Mohammed i M. Bernard. *Graph File Formats*. Spraw. tech. Mona, Kingston, Jamajka: Department of Mathematics and Computer Science, The University of the West Indies, 2004. URL: `http://www2.sta.uwi.edu/~mbernard/research_files/fileformats.pdf` (term. wiz. 29. 04. 2017).

[Wil07] Robin J. Wilson. *Wprowadzenie do teorii grafów*. 2 wyd. Wydawnictwo Naukowe PWN, 2007. ISBN: 978-83-01-15066-2.

[RW08] Kenneth A. Ross i Charles R.B. Wright. *Matematyka dyskretna*. 5 wyd. Wydawnictwo Naukowe PWN, 2008. ISBN: 978-83-01-14380-0.

[Gep] Gephi. *Supported Graph Formats*. The Gephi Consortium. URL: `https://gephi.org/users/supported-graph-formats/` (term. wiz. 29. 04. 2017).

[Sta] Mathematics StackExchange. *Online tool for making graphs (vertices and edges)*. Stack Overflow. URL: `https://math.stackexchange.com/questions/13841/online-tool-for-making-graphs-vertices-and-edges` (term. wiz. 02. 05. 2017).