Dokumentacja projektu Dijxter

Sebastian Cielemęcki

23 czerwca 2014

Spis treści

1 Wstęp

Celem projektu Dijxter było przygotowanie aplikacji edukacyjnej, mającej być pomocą w nauce algorytmów. Algorytm opracowany przez Edsgera Dijkstrę, służący do znajdywania najkrótszych ścieżek w grafie, jest jednym z najbardziej znanych, podstawowych algorytmów grafowych. Program Dijxter, nazwą nawiązujący do algorytmu Dijkstry, poprzez wizualizację jego działania ma ułatwić zapoznanie się z algorytmem, jego ideą, a także używaną przezeń strukturą kolejki priorytetowej opartej na kopcu.

2 Opis

Część użytkowa aplikacji składa się z 3 głównych widoków - kolumn

- Widok kolejki priorytetowej
- Część kontrolna z listą wykonanych kroków algorytmu
- Widok stanu grafu

W części kontrolnej za pomocą przycisku można wygenerować nowy graf, a następnie prześledzić kolejne kroki wykonywanego na nim algorytmu Dijkstry. W trakcie jego działania aktualizowana jest lista kroków wykonywanych przez algorytm, tj. operacji przejścia do nowego wierzchołka, zmniejszenie odległości zmieniających stan grafu.

Kliknięcie na jeden z zapisanych kroków na liście powoduje odtworzenia stanu struktur na sąsiednich widokach (grafu i kopca) po wykonaniu tego kroku.

3 Technologie

Aplikacja oparta jest o technologie webowe, stąd uruchamiana jest pod przeglądarką internetową. Aplikacja wykorzystuje następujące technologie i narzędzia:

3.1 CoffeeScript 1.6.x (pliki *.coffee)

Język skryptowy, kompilowalny do JavaScripta, który jest głównym językiem sterującym działaniem aplikacji webowych po stronie klienta. Dzięki uproszczonej składni, przypominającej bardziej języki obiektowe jak np. Ruby, istotnie zwiększa produktywność, a kod jest bardziej zwięzły i czytelny.

3.2 HTML5 (pliki *.html)

HTML jest językiem opisowym strony internetowych, stanowi szkielet wizualnej części aplikacji.
Aplikacja Dijkster ponadto korzysta intensywnie z dostępnego w HTML5 obiektu canvas, który pozwala za pomocą różnych kształtów rysować grafy i kopce.

3.3 CSS (pliki *.css)

CSS jest wykorzystywany do opisu wyglądu, właściwości i ułożenia elementów wizualnej części aplikacji.

3.4 JQuery v1.11

JQuery ułatwia odwoływanie się do elementów drzewa DOM i manipulowanie nimi, podpinanie zdarzeń, w porównaniu z czystym JavaScriptem.

3.5 underscore.js

Podobnie jak JQuery jest to biblioteka pozwalająca poradzić sobie z ograniczeniami JavaScriptu, zwłaszcza w oparacjach na kolekcjach. Charakterystyczne jest użycie znaku '_' podczas korzystania z funkcji tej biblioteki.

3.6 Coffee Toaster

Minimalistyczny system do budowania aplikacji opartych o język Coffee Script. Pozwala skompilować wiele plików *.coffeescript do jednego pliku *.js (w wersji debug oraz release), który następnie załączany jest do głównego pliku html aplikacji.

4 Struktura aplikacji

4.1 Pliki i katalogi

W katalogu głównym znajdują się 2 pliki oraz 4 katalogi:

- /app.html
  Główny plik aplikacji, jego uruchomienie otwira aplikację w przeglądarce internetowej.
- /toaster.coffee
  Plik konfiguracyjny zawierający informacje o ścieżkach plików *.coffee do kompilowania, a także o ścieżkach skompilowanych rezultatów.
- /css
  Zawiera plik *.css, czyli arkusz stylów definiujący wygląd aplikacji
- /js
  Zawiera dwa pliki:
  - app.js - skompilowane źródło plików *.coffee w wersji 'release'
  - app-debug.js - plik załączający do aplikacji skompilowane osobno pliki *.coffee w folderze /js/toaster. Przeznaczony do debugowania aplikacji.
- /vendors
  Folder zawierający dodatkowe, zewnętrzne biblioteki (JQuery, underscore.js).
- /src
  Zawiera wszystkie klasy, o które oparta jest cała logika programu.

4.2 Klasy

4.2.1 App

Główna klasa, której utworzenie uruchamia działanie programu, steruje wewnętrzną logiką i komunikacją między obiektami innych klas.

Główne metody:
- **generate_graph**
- **dijkstra** - uruchamia algorytm dijkstry
- **process_nodes** - steruje działaniem algorytmu, dodając odpowiednie opóźnienia między krokami

4.2.2 GNode

Reprezentuje wierzchołek grafu; posiada nazwę, wartość odległości od ustalonego wierzchołka oraz listę sąsiedztwa.

4.2.3 Graph

Reprezentuje graf, czyli listę wierzchołków wraz z metodami pozwalającymi wykonywać podstawowe operacje.

Metody:
- **add_node** - dodaje do listy wierzchołków nowy obiekt GNode o określonej etykiecie
- **add_edge** - dodaje nową krawędź o podanej wartości między wierzchołkami o danych numerach (dopisuje do listy sąsiedztwa)

4.2.4 Queue

Reprezentuje kolejkę opartą na kopcu binarnym, przetrzymuje na nim wierzchołki grafu.

Metody:
- **delete_min**
- **heapify** - przywraca porządek (w dół)
- **heapify_up** - przywraca porządek (w górę)
- **swap** - zamienia wierzchołki w kopcu
- **index_transform** - uaktualnia tablicę rzeczywistych pozycji wierzchołków w tablicy podczas wykonania operacji zamiany

4.2.5 GraphView

Klasa odpowiadająca za widok grafu w obiekcie canvas. Przechowuje wszystkie poprzednie stany, w jakich znajdował się graf.

Główne metody:
- **generate** - zapamiętuje podaną listę wierzchołków i generuje pozycje wierzchołków w widoku
- **draw_graph**
- **draw_node**
- **update** - dodaje nowy stan grafu i aktualizuje widok

4.2.6 QueueView

Klasa odpowiadająca za widok kolejki priorytetowej, przechowuje wszystkie poprzednie stany kolejki.

Główne metody:
- **generate** - zapamiętuje podaną listę wierzchołków i generuje pozycje elementów kopca w widoku
- **draw_queue**
- **draw_node**
- **update** - dodaje nowy stan kolejki i aktualizuje widok

4.2.7 StatesList

Klasa zarządzająca listą kroków, podpina zdarzenia pod dodane elementy na liście kroków i komunikuje się z obiektami klas GraphView i QueueView w celu odtworzenia wybranego stanu. Metoda **update** dodaje nowy stan do listy kroków i podłącza odpowiednio zdarzenie kliknięcia.

4.2.8 Rand

Klasa wspomagająca generowanie losowych wartości w okrzęlonym przedziale lub ciągu unikalnych wartości w przedziale.

Metody:
- **random** - metoda klasowa, generuje losową liczbę w określonym przedziale
- **unique_random** - metoda klasowa, generuje n unikalnych wartości w przedziale