Sprawozdanie z testowania algorytmu FORDA-FULKERSONA

Testowanie i weryfikacja oprogramowania 2012/2013 — Projekt

Data	2013-01-08
Wersja	1.0
Autorzy	TC, MM, MO, RW

1 Plan testowania

1.1 Wprowadzenie

Testowaniu podlega system realizujący algorytm FORDA-FULKERSONA służący do wyznaczania maksymalnego przepływu w sieciach. Testowane elementy pochodzą z kodu załączonego wraz z książką *Algorytmy. Almanach*. Celem testowania jest zweryfikowanie poprawności już zaimplementowanej funkcjonalności.

1.2 Testowane elementy

Testowaniu podlega projekt `ford-fulkerson-test-suite` zawierający implementację algorytmu FORDA-FULKERSONA. Testowana wersja projektu znajduje się w repozytorium, w rewizji `e5ebef7` o nazwie `testing-final`.

1.3 Testowana funkcjonalność

Uwaga procesu testowania będzie skupiona na konkretnych implementacjach systemu:

- dla klasy `FlowNetwork` — implementacja `FlowNetworkArray`,
- dla klasy `Search` — implementacja `DFS_SearchArray`.

W chwili obecnej dostępne są też alternatywne implementacje operujące na listach, odpowiednio `FlowNetworkAdjacencyList` oraz `DFS_SearchList`. Wyszczególnienie implementacji jest istotne dla planowania testów biało-skrzynkowych. Testy czarno-skrzynkowe nie mają wiedzy o wybranej implementacji, powinny działać niezależnie od wyboru.

1.4 Testowana funkcjonalność — wyłączenia

Do zakresu testów nie wchodzą moduły oferujące funkcjonalność trywialną lub wyłącznie pomocniczną:

- pakiet `algs.heap`,
- pakiet `algs.list`.

Z zakresu testowania wyłączone zostaną również klasy, które

- udostępniają funkcjonalność nie wykorzystywaną przez testowany algorytm:
  - klasa `algs.network.ShortestPathArray`,
  - klasa `algs.network.generator.FlowNetworkGenerator`;
- oraz zawierają implementacje, które odrzucono:
  - klasa `algs.network.FlowNetworkAdjacencyList`,
  - klasa `algs.network.ShortestPathArray`.

1.5 Podejście

Środowisko testowe zakłada korzystanie z:

- NetBeans IDE 7.2.1

- Oracle JDK 1.7_u9
- TestNG 6.8

Specyficzne metodyki, narzędzia i kryteria satysfakcji są opisane w sekcji 2.

2 Szczegółowy plan testowania

2.1 Testowanie czarno-skrzynkowe

2.1.1 Wprowadzenie

Cel testów czarno-skrzynkowych nie różni się od wcześniejszych założeń. Testy czarno-skrzynkowe koncentrują się na weryfikacji wyników działania algorytmu FORDA-FULKERSONA dla określonych danych wejściowych.

2.1.2 Testowane elementy

Głównym obiektem testów czarno-skrzynkowych jest klasa `FordFulkerson` z projektu `ford-fulkerson-test-suite`, zawierająca implementację algorytmu FORDA-FULKERSONA. W trakcie testów bezpośrednio wykorzystywane są również inne klasy wchodzące do projektu `ford-fulkerson-test-suite`.

2.1.3 Testowana funkcjonalność

Zakres testów czarno-skrzynkowych obejmuje:

1. Działanie konstruktora klasy `FordFulkerson`.
2. Możliwość wykonania metody `compute()` klasy `FordFulkerson`.
3. Poprawność wyników działania implementacja algorytmu FORDA-FULKERSONA w klasie `FordFulkerson`. Weryfikacja polega na porównaniu wartości zwracanych przez `toString()` po wykonaniu metody `compute()`, do wartości zwracanych przez wyrocznię.

2.1.4 Testowana funkcjonalność — wyłączenia

Z testowania czarno-skrzynkowego wyłączone zostaną funkcjonalności:

- Z klas niewykorzystywanych ze względu na wybór implementacji:
  - `DFS_SearchList`,
  - `FlowNetworkAdjacencyList`,
  - `ShortestPathArray`.
- Z klas abstrakcyjnych, po których dziedziczą wykorzystywane klasy:
  - `FlowNetwork`
  - `Search`
- Z klas reprezentujących szczegóły implementacji:
  - `EdgeInfo`,
  - `DFS_SearchArray` – implementująca algorytm przeszukiwania DEPTH-FIRST SEARCH na implementacji sieci przepływu z klasy `FlowNetworkArray`, której instancja jest wymagana jako drugi argument konstruktura klasy `FordFulkerson`;
  - `VertexInfo`,
  - `VertexStructure`.

2.1.5 Podejście

Podstawą do projektu testów czarno-skrzynkowych jest wyróżnienie klas abstrakcji. Odpowiedni dobór klas pozwoli przetestować implementację algorytmu FORDA-FULKERSONA w rozsądnym zakresie, jednocześnie unikając

testowania totalnego. Zidentyfikowane zostały następujące klasy abstrakcji:

1. Ze względu na ilość węzłów:
   (a) Ujemna ilość węzłów.
   (b) Brak jakichkolwiek węzłów.
   (c) Dokładnie jeden węzeł, źródłowy, pośredni lub ujście.
   (d) Sieć przepływowa bez węzłów pośrednich.
   (e) Sieć przepływowa z jednym węzłem pośrednim – jako minimalna ilość węzłów pośrednich w sieci przepływowej z węzłami pośrednimi.
   (f) Sieć przepływowa z kilkoma węzłami pośrednimi – jako niegraniczna ilość węzłów w sieci przepływowej z węzłami pośrednimi.
   (g) Sieć przepływowa z $1e9$ węzłami pośrednimi – jako maksymalna ilość węzłów pośrednich w sieci przepływowej z węzłami pośrednimi.
   (h) Ilość węzłów mniejsza od zadeklarowanej.
   (i) Ilość węzłów większa od zadeklarowanej.
2. Ze względu na pętle:
   (a) W źródle.
   (b) W węźle pośrednim.
   (c) W ujściu.
   (d) Bez pętli.
3. Ze względu na krawędzie bezpośrednie pomiędzy źródłem, a ujściem:
   (a) Ze źródła do ujścia.
   (b) Z ujścia do źródła.
   (c) Bez krawędzi pomiędzy źródłem, a ujściem.
4. Ze względu na przepustowość krawędzi:
   (a) Dodatnia.
   (b) Zerowa.
   (c) Ujemna.
5. Ze względu na poprawność krawędzi:
   (a) Poprawne krawędzie.
   (b) Krawędzie zaczynające się w nieistniejącym węźle.
   (c) Krawędzie kończące się w nieistniejącym węźle.
6. Ze względu na istnienie krawędzi zwielokrotnionych:
   (a) Bez krawędzi zwielokrotnionych.
   (b) Z bezpośrednią zwielokrotnioną krawędzią pomiędzy źródłem, a ujściem.
   (c) Z krawędziami zwielokrotnionymi o przeciwnych zwrotach.

2.2 Testowanie biało-skrzynkowe

2.2.1 Wprowadzenie

Kryterium satysfakcji przeprowadzanych testów wyznaczane jest na podstawie współczynnika pokrycia kodu. Minimalną dopuszczalną wartością jest 90%. Część testów została dostarczona razem z systemem.

2.2.2 Testowane elementy

Lista poszczególnych klas poddawanych testowaniu oraz pokrycie kodu zapewnione przez testy zawarte wraz z kodem źródłowym.

Pakiet algs.network	
Klasa	**Aktualne pokrycie**
FordFulkerson	100%
FlowNetwork	100%
FlowNetworkAdjacencyList	75%
FlowNetworkArray	94%
Search	100%
DFS_SearchList	100%
DFS_SearchArray	100%
EdgeInfo	100%
VertexInfo	100%
VertexStructure	93%

W oddzielnym załączniku sprawozdanie/raport/index.html znajduje się stan pokrycia przed utworzeniem nowego zestawu testów.

2.3 Testowana funkcjonalność — wyłączenia

W zakres testowania biało-skrzynkowego nie wchodzi część metod dostarczanych przez klasy testowane:

- FlowNetworkArray.getCost()

2.3.1 Podejście

Testowanie zostanie przeprowadzone techniką testowania strukturalnego. Głównym kryterium zaliczenia testów będzie współczynnik pokrycia linii kodu, którego wartość musi przekraczać 90%. Dodatkowo do wytycznych środowiska testowego ustalonych globalnie użyte zostaną:

- Cobertura 1.9.4.1

2.4 Testowanie z wykorzystaniem obiektów pozornych

2.4.1 Wprowadzenie

Celem testów integracyjnych jest sprawdzenie poprawności danych przekazywanych między modułami. Wykorzystanie obiektów pozornych znacznie upraszcza ten proces, dając możliwość testowania ”jednocześnie” tylko jednego modułu.

2.4.2 Testowane elementy

Punktem wejściowym testów jest klasa FordFulkerson zawierająca implementację algorytmu Forda-Fulkersona.

2.4.3 Podejście

Testy zostaną przeprowadzone przy podejściu z góry do dołu, oraz integracji w szerz. Jako punkt wejścia obrana została klasa FordFulkerson.

2.4.4 Zadania testowania

Zadanie	Osoba odpowiedzialna
Projekt testów	MM
Przygotowanie przypadków testowych	MM
Implementacja testów	MM
Uruchomienie testów i weryfikacja wyników	MM, TC
Akceptacja wyników przebiegu testowania	TC

3 Projekt testów

3.1 Testowanie czarno-skrzynkowe

Poniżej opisane są przypadki testowe wybrane do realizacji w ramach testów czarno-skrzynkowych.

3.1.1 Sieć bez węzłów

Sieć bez żadnych węzłów nie istnieje, nie da się dla niej wyznaczyć maksymalnego przepływu.

3.1.2 Sieć z 1 węzłem

Za szczególny przypadek sieci przepływu można uznać sieć składającą się z dokładnie jednego węzła. Niezależnie od pozostałych parametrów, taka sieć powinna zostać odrzucona jako nieprawidłowa. Wyróżnić można warianty złożone z:

- Samego źródła.
- Węzła pośredniego.
- Ujścia.
- Wspólnego źródła i ujścia.
- Wspólnego źródła i ujścia z pętlą.

Szczególną uwagę należy zwrócić na ostatni z wymienionych wariantów, który jako jedyny posiadający krawędź, którą można traktować jako krawędź ze źródła do ujścia.

Można wyróżnić cztery schematy sieci z pojedynczym węzłem, każdy w czterech wariantach: bez pętli, z pętlą o dodatniej/ujemnej/zerowej przepustowości, co przekładałoby się na szesnaście przypadków testowych. W ramach planu ograniczono się do pięciu najistotniejszych kombinacji.

3.1.3 Sieć bez węzłów pośrednich

Sieć złożona ze źródła i ujścia, bez jakichkolwiek węzłów pośrednich. Wyróżnione zostały następujące przypadki testowe dla topologii sieci z węzłami:

- Nie połączonymi żadną krawędzią.
- Połączonymi pojedynczą krawędzią skierowaną od źródła do ujścia o dodatniej przepustowości.
- Połączonymi pojedynczą krawędzią skierowaną od źródła do ujścia o dodatniej przepustowości z pętlą o dodatniej przepustowości w ujściu.
- Połączonymi pojedynczą krawędzią skierowaną od źródła do ujścia o dodatniej przepustowości z pętlą o ujemnej przepustowości w źródle.
- Połączonymi pojedynczą krawędzią skierowaną od źródła do ujścia o zerowej przepustowości.
- Połączonymi pojedynczą krawędzią skierowaną od źródła do ujścia o ujemnej przepustowości.

- Połączonymi pojedynczą krawędzią skierowaną od ujścia do źródła o dodatniej przepustowości.
- Połączonymi pojedynczą krawędzią skierowaną od ujścia do źródła o ujemnej przepustowości.
- Z wieloma krawędziami skierowanymi od żródła do ujścia.
- Z wieloma krawędziami skierowanymi od ujścia do źródła.
- Z wieloma krawędziami skierowanymi w różnych stronach.

Można bez problemu zdefiniować ponad sto wariantów topologii sieci przepływowej bez węzłów pośrednich. Trzy ze względu na rodzaj przepustowości krawędzi (dodatnia/ujemna/zerowa), siedem ze względu na istnienie i rodzaj pętli (brak/w źródle z dodatnią przepustowością/w źródle z ujemną przepustowością/w źródle z zerową przepustowością/w ujściu z dodatnią przepustowościa/w ujściu z ujemną przepustowościa/w ujściu z zerową przepustowościa), oraz sześć wariantów połączeń pomiędzy krawędziami (bez krawędzi, pojedyncza od źródła do ujścia, pojedyncza od ujścia do źródła, zwielokrotniona ze źródła do ujścia, zwielokrotniona z ujścia do źródła, z pomieszanymi zwrotami). Do przetestowania zostało wybranych 11 najbardziej reprezentatywnych topologii sieci bez węzłów pośrednich.

3.1.4 Sieć z 1 węzłem pośrednim

Prosta sieć przepływu zbudowana ze źródła, jednego węzła pośredniego i ujścia. Wyróżnione zostały następujące przypadki testowe w zależności od topologii:

- Połączone pojedynczymi krawędziami skierowanymi ze źródła do węzłą pośredniego i z węzła pośredniego do ujścia.
- Z pojedynczymi krawędziami skierowanymi z ujścia do węzła pośredniego i z węzła pośredniego do źródła.
- Z pojedynczymi krawędziami skierowanymi z węzła pośredniego do ujścia i z węzła pośredniego do źródła.
- Z pojedynczymi krawędziami skierowanymi ze źródła do węzła pośredniego i z ujścia do węzła pośredniego.
- Połączone pojedynczymi krawędziami skierowanymi ze źródła do węzła pośredniego i z węzła pośredniego do źródła z dodatkową krawędzią o dodatniej przepustowości ze źródła do ujścia.
- Połączone pojedynczymi krawędziami skierowanymi ze źródła do węzła pośredniego i z węzła pośredniego do źródła z pętlą o dodatniej przepustowości w węźle pośrednim.
- Połączone zwielokrotnionymi krawędziami ze źródła do węzła pośredniego i z węzła pośredniego do źródła, z mieszanymi zwrotami.
- Połączone pojedynczą krawędzią skierowaną ze źródła do węzła pośredniego, bez krawędzi do ujścia.
- Połączone pojedynczą krawędzia skierowaną ze źródła do węzła pośredniego, oraz krawędzią ze źródłą do ujścia.
- Połączone pojedynczą krawędzia skierowaną z węzła pośredniego do ujścia, bez połączenia ze źródłem.

W przypadku sieci z jednym węzłem pośrednim można wyróżnić nawet ponad tysiąc różnych rodzajów topologii sieci. Dla dwóch krawędzi istnieje już dziewięć kombinacji rodzajów przepustowości (dodatnia/ujemna/zerowa), trzy możliwe położenia pętli (w źródle/węźle pośrednim/ujściu), trzy rodzaje przepustowości w pętli (dodatni/ujemny/zerowy), cztery możliwe kombinacje zwrotów krawędzi, oraz cztery warianty zwielokrotnienia krawędzi w topologii sieci. Doliczenie się takiej liczby możliwych rodzajów topologii sieci, nie wymagało nawet uwzględnienia możliwości istnienia krawędzi bezpośredniej ze źródła do ujścia, czy braku którejś krawędzi. Ostatecznie zdecydowano się zatem na dziesięć różnych przypadków testowych.

3.1.5 Sieć z 2 węzłami pośrednimi połączonymi równolegle

W tej sekcji jako warte przetestowania wyróżnioniono następujące schematy sieci z 2 węzłami pośrednimi połączonymi równolegle:

- Z krawędziami o dodatniej przepustowości ze źródła do obu węzłów pośrednich i z obu węzłów pośrednich do ujścia.
- Z dodatkową krawędzią o dodatniej przepustowości z pierwszego węzła pośredniego do drugiego węzła pośredniego.
- Z dodatkową krawędzią o dodatniej przepustowości z drugiego węzła pośredniego do pierwszego węzła pośredniego.
- Z dodatkowymi krawędziami o dodatniej przepustowości z pierwszego węzła pośredniego do drugiego węzła pośredniego i z drugiego węzła pośredniego do pierwszego węzła pośredniego.[1]
- Z dodatkową krawędzią o dodatniej przepustowości ze źródła do ujścia.
- Z dodatkową krawędzia o dodatniej przepustowości z ujścia do źródła.
- Z dodatkową krawędzią o dodatniej przepustowości z ujścia do pierwszego węzła pośredniego.
- Z dodatkową krawędzią o dodatniej przepustowości z pierwszego węzła pośredniego do źródłą.
- Z dodatkowymi krawędziami o dodatniej przepustowości z ujścia do pierwszego węzła pośredniego i z pierwszego węzłą pośredniego do źródła.

Przypadki testowe dla sieci z 2 węzłami pośrednimi zostały pomyślane przede wszystkim dla przetestowania zachowania oprogramowania dla różnych schematów połączeń, stąd chociażby brak w tej sekcji klas abstrakcji z pętlami.

3.1.6 Sieć z 2 węzłami pośrednimi połączonymi szeregowo

W tym wypadku jako istotne przypadki testowe zdefiniowano topologie sieci z 2 węzłami pośrednimi połączonymi szeregowo:

- Z krawędziami o dodatniej przepustowości ze źródła do pierwszego węzła pośredniej, z pierwszego węzła pośredniego do drugiego węzła pośredniego i z drugiego węzła pośredniego do ujścia.
- Sieć z 2 węzłami pośrednimi połączonymi szeregowo z wieloma krawędziami pomiędzy węzłami pośrednimi.
- Sieć z 2 węzłami pośrednimi połączonymi szeregowo z krawędzią o zerowej przepustowości pomiędzy węzłami pośrednimi.
- Sieć z 2 węzłami pośrednimi połączonymi szeregowo z krawędzią o ujemnej przepustowości pomiędzy węzłami pośrednimi.
- Sieć z 2 węzłami pośrednimi połączonymi szeregowo z krawędzią z drugiego węzłą pośredniego do pierwszego węzła pośredniego.[2]
- Sieć z 2 węzłami pośrednimi połączonymi szeregowo z krawędzią od pierwszego węzła pośredniego do ujścia.
- Sieć z 2 węzłami pośrednimi połączonymi szeregowo z krawędzią od źródła do drugiego węzła pośredniego.

Dodanie schematów z połączeniami szeregowymi zasadniczo wyczerpuje możliwości budowy schematów połączeń, każdy bardziej złożony (prawidłowy) schemat połączeń będzie bazował na pewnej kombinacji powyższych wariacji na temat połączeń szeregowych i równoległych.

3.1.7 Sieć z nieistniejącymi węzłami

Liczba węzłów określona w danych wejściowych jest większa niż liczba zdefiniowanych węzłów.

[1] Potencjalne zagrożenie stworzeniem złożonej pętli.

[2] Ujemna przepustowość pomiędzy pierwszym a drugim węzłem pośrednim nie jest tożsama z krawędzią o dodatniej przepustowości od drugiego do pierwszego węzła pośredniego. Ujemna przepustowość krawędzi jest niepoprawna z definicji, natomiast krawędź skierowana w przeciwną stronę jest jak najbardziej poprawna z punktu widzenia definicji, choć taka sieć wciąż nie miałaby żadnej przepustowości.

3.1.8 Sieć z ujemną liczbą węzłów

W danych wejściowych liczba węzłów w sieci jest określona za pomocą liczby ujemnej.

3.1.9 Sieć z $1e9$ węzłów

Test dla bardzo dużej (najlepiej granicznej) wartości liczby węzłów.

3.2 Testowanie biało-skrzynkowe

3.2.1 Klasa testowana `algs.network.VertexStructure`

Testowana metoda	`String toString()`
Opis	Wypisuje listę węzłów wychodzących i wchodzących.
Przypadki testowe	1. Struktura bez zdefiniowanych krawędzi wejscia/wyjścia. 2. Struktura z pojedynczą krawędzią wejścia/wyjścia. 3. Struktura z wieloma krawędziami wejścia wyjścia.
Wykonawca	MO

3.2.2 Klasa testowana `algs.network.FlowNetworkArray`

Testowana metoda	`FlowNetworkArray(int sourceIndex, int sinkIndex, int numVertices)`
Opis	Konstruktor minimalnej struktury sieci. Inicjalizuje tylko niezbędne zmienne.
Przypadki testowe	1. Wartości oczekiwane. 2. Wartości ujemne. 3. `sinkIndex < sourceIndex` 4. `sinkIndex - sourceIndex > numVertices`
Wykonawca	MO

Testowana metoda	`FlowNetworkArray(int numVertices, int srcIndex, int sinkIndex, Iterator<Edges> edges)`
Opis	Konstruktor struktury reprezentującej graf przpływu.
Przypadki testowe	1. Wartości oczekiwane. 2. Wartości ujemne. 3. `sinkIndex - sourceIndex > numVertices` 4. `sinkIndex < sourceIndex` 5. Pusta kolekcja krawędzi.
Wykonawca	MO

Testowana metoda	`void validate()`
Opis	Metoda weryfikuje czy informacje na temat sieci są poprawne. Zwracany jest wyjątek, `IllegalStateException` w dwóch przypadkach: 1. Przepływ krawędzi jest większy niż jej przepustowość. 2. Liczba krawędzi wchodzących jest różna od krawędzi wychodzących.
Przypadki testowe	1. Wartości oczekiwane. 2. Wartości ujemne. 3. `sinkIndex - sourceIndex > numVertices` 4. `sinkIndex < sourceIndex` 5. Pusta kolekcja krawędzi.
Wykonawca	MO

3.3 Testowanie z wykorzystaniem obiektów pozornych

3.4 Klasa testowana `algs.network.FlowNetworkArray`

Testowana metoda	`void populate()`
Opis	Metoda pomocnicza przy wypełnianiu.
Przypadki testowe	1. Struktura z ośmioma krawędziami.
Wykonawca	MM

3.4.1 Klasa testowana `algs.network.FordFulkerson`

Testowana metoda	`Boolean compute()`
Opis	Wyznacza maksymalny przepływ.
Przypadki testowe	1. Struktura z ośmioma krawędziami.
Wykonawca	MM

4 Realizacja testów

4.1 Testowanie czarno-skrzynkowe

4.1.1 Sieć bez węzłów

Wszystkie testy na sieci bez węzłów zrealizowano w klasie `rw.blackbox.EmptyNetworkTest`.

TestCase	`rw.blackbox.EmptyNetworkTest.testCase1()`
ID	C.1.1.1
Dane wejściowe	• liczba węzłów w sieci: `0`, • indeks źródła: 2, • indeks ujścia: 5, • krawędzie: NULL;
Oczekiwany wynik	`0`
Wykonawca	RW

TestCase	`rw.blackbox.EmptyNetworkTest.testCase2()`
ID	C.1.1.2
Dane wejściowe	• liczba węzłów w sieci: `0`, • indeks źródła: `0`, • indeks ujścia: `0`, • krawędzie: NULL;
Oczekiwany wynik	`Exception`
Wykonawca	RW

4.1.2 Sieć z 1 węzłem

Wszystkie testy zrealizowane w klasie `rw.blackbox.JustOneElementTest`.

TestCase	`rw.blackbox.JustOneElementTest.testCase1()`
ID	C.1.2.1
Dane wejściowe	• liczba węzłów w sieci: 1, • indeks źródła: `0`, • indeks ujścia: 1, • krawędzie: NULL;
Oczekiwany wynik	`0`
Wykonawca	RW

TestCase	`rw.blackbox.JustOneElementTest.testCase2()`
ID	C.1.2.2
Dane wejściowe	• liczba węzłów w sieci: 1, • indeks źródła: 2, • indeks ujścia: 1, • krawędzie: NULL;
Oczekiwany wynik	`Exception`
Wykonawca	RW

TestCase	`rw.blackbox.JustOneElementTest.testCase3()`
ID	C.1.2.3
Dane wejściowe	• liczba węzłów w sieci: 1, • indeks źródła: 1, • indeks ujścia: 0, • krawędzie: NULL;
Oczekiwany wynik	`Exception`
Wykonawca	RW

TestCase	`rw.blackbox.JustOneElementTest.testCase4()`
ID	C.1.2.4
Dane wejściowe	• liczba węzłów w sieci: 1, • indeks źródła: 0, • indeks ujścia: 0, • krawędzie: NULL;
Oczekiwany wynik	`0`
Wykonawca	RW

TestCase	`rw.blackbox.JustOneElementTest.testCase5()`
ID	C.1.2.5
Dane wejściowe	• liczba węzłów w sieci: 1, • indeks źródła: 0, • indeks ujścia: 1, • krawędzie: `{(0, 0, 5)}`
Oczekiwany wynik	`0`
Wykonawca	RW

4.1.3 Sieć bez węzłów pośrednich

Testy na sieciach bez węzłów pośrednich zrealizowano w klasie `rw.blackbox.JustSourceAndSinkTest`

TestCase	`rw.blackbox.JustSourceAndSinkTest.testCase1()`
ID	C.1.3.1
Dane wejściowe	• liczba węzłów w sieci: 2, • indeks źródła: 0, • indeks ujścia: 1, • krawędzie: NULL
Oczekiwany wynik	`0`
Wykonawca	RW

TestCase	`rw.blackbox.JustSourceAndSinkTest.testCase2()`
ID	C.1.3.2
Dane wejściowe	• liczba węzłów w sieci: 2, • indeks źródła: `0`, • indeks ujścia: 1, • krawędzie: `{(0, 1, 9)}`
Oczekiwany wynik	`9`
Wykonawca	RW

TestCase	`rw.blackbox.JustSourceAndSinkTest.testCase3a()`
ID	C.1.3.3
Dane wejściowe	• liczba węzłów w sieci: 2, • indeks źródła: `0`, • indeks ujścia: 1, • krawędzie: `{(0, 1, 57956756);(1, 1, 6363)}`
Oczekiwany wynik	`57956756`
Wykonawca	RW

TestCase	`rw.blackbox.JustSourceAndSinkTest.testCase3b()`
ID	C.1.3.4
Dane wejściowe	• liczba węzłów w sieci: 2, • indeks źródła: `0`, • indeks ujścia: 1, • krawędzie: `{(0, 1, 12);(1, 1, 2)}`
Oczekiwany wynik	`Exception`
Wykonawca	RW

TestCase	`rw.blackbox.JustSourceAndSinkTest.testCase4a()`
ID	C.1.3.5
Dane wejściowe	• liczba węzłów w sieci: 2, • indeks źródła: `0`, • indeks ujścia: 1, • krawędzie: `{(0, 1, 346723);(0, 0, -1623474)}`
Oczekiwany wynik	`346723`
Wykonawca	RW

TestCase	`rw.blackbox.JustSourceAndSinkTest.testCase4b()`
ID	C.1.3.6
Dane wejściowe	• liczba węzłów w sieci: 2, • indeks źródła: `0`, • indeks ujścia: 1, • krawędzie: `{(0, 1, 4);(0, 0, -2)}`
Oczekiwany wynik	`Exception`
Wykonawca	RW

TestCase	`rw.blackbox.JustSourceAndSinkTest.testCase5()`
ID	C.1.3.7
Dane wejściowe	• liczba węzłów w sieci: 2, • indeks źródła: `0`, • indeks ujścia: 1, • krawędzie: `{(0, 1, 0)}`
Oczekiwany wynik	`0`
Wykonawca	RW

TestCase	`rw.blackbox.JustSourceAndSinkTest.testCase6a()`
ID	C.1.3.8
Dane wejściowe	• liczba węzłów w sieci: 2, • indeks źródła: `0`, • indeks ujścia: 1, • krawędzie: `{(0, 1, -5)}`
Oczekiwany wynik	`0`
Wykonawca	RW

TestCase	`rw.blackbox.JustSourceAndSinkTest.testCase6b()`
ID	C.1.3.9
Dane wejściowe	• liczba węzłów w sieci: 2, • indeks źródła: `0`, • indeks ujścia: 1, • krawędzie: `{(0, 1, -173848)}`
Oczekiwany wynik	`0`
Wykonawca	RW

TestCase	`rw.blackbox.JustSourceAndSinkTest.testCase7()`
ID	C.1.3.10
Dane wejściowe	• liczba węzłów w sieci: 2, • indeks źródła: `0`, • indeks ujścia: 1, • krawędzie: `{(1, 0, 5)}`
Oczekiwany wynik	`0`
Wykonawca	RW

TestCase	`rw.blackbox.JustSourceAndSinkTest.testCase8a()`
ID	C.1.3.11
Dane wejściowe	• liczba węzłów w sieci: 2, • indeks źródła: `0`, • indeks ujścia: 1, • krawędzie: `{(1, 0, -4)}`
Oczekiwany wynik	`0`
Wykonawca	RW

TestCase	`rw.blackbox.JustSourceAndSinkTest.testCase8b()`
ID	C.1.3.12
Dane wejściowe	• liczba węzłów w sieci: 2, • indeks źródła: `0`, • indeks ujścia: 1, • krawędzie: `{(1, 0, -1639273)}`
Oczekiwany wynik	`Exception`
Wykonawca	RW

TestCase	`rw.blackbox.JustSourceAndSinkTest.testCase9a()`
ID	C.1.3.13
Dane wejściowe	• liczba węzłów w sieci: 2, • indeks źródła: `0`, • indeks ujścia: 1, • krawędzie: `{(0,1,12);(0,1,2);(0,1,4)}`
Oczekiwany wynik	`18`
Wykonawca	RW

TestCase	`rw.blackbox.JustSourceAndSinkTest.testCase9b()`
ID	C.1.3.14
Dane wejściowe	• liczba węzłów w sieci: 2, • indeks źródła: `0`, • indeks ujścia: 1, • krawędzie: `{(1,0,9);(1,0,3);(1,0,15)}`
Oczekiwany wynik	`Exception`
Wykonawca	RW

TestCase	`rw.blackbox.JustSourceAndSinkTest.testCase10a()`
ID	C.1.3.15
Dane wejściowe	• liczba węzłów w sieci: 2, • indeks źródła: `0`, • indeks ujścia: 1, • krawędzie: `{(0,1,-16);(0,1,-5);(0,1,-3)}`
Oczekiwany wynik	`0`
Wykonawca	RW

TestCase	`rw.blackbox.JustSourceAndSinkTest.testCase10b()`
ID	C.1.3.16
Dane wejściowe	• liczba węzłów w sieci: 2, • indeks źródła: `0`, • indeks ujścia: 1, • krawędzie: `{(1,0,-7);(1,0,-1)}`
Oczekiwany wynik	`Exception`
Wykonawca	RW

TestCase	`rw.blackbox.JustSourceAndSinkTest.testCase11a()`
ID	C.1.3.17
Dane wejściowe	• liczba węzłów w sieci: 2, • indeks źródła: `0`, • indeks ujścia: 1, • krawędzie: `{(0,1,4);(1,0,3)}`
Oczekiwany wynik	`4`
Wykonawca	RW

TestCase	`rw.blackbox.JustSourceAndSinkTest.testCase11b()`
ID	C.1.3.18
Dane wejściowe	• liczba węzłów w sieci: 2, • indeks źródła: `0`, • indeks ujścia: 1, • krawędzie: `{(0,1,6);(1,0,11);(0,1,7);(1,0,2)}`
Oczekiwany wynik	`13`
Wykonawca	RW

TestCase	`rw.blackbox.JustSourceAndSinkTest.testCase11c()`
ID	C.1.3.19
Dane wejściowe	• liczba węzłów w sieci: 2, • indeks źródła: `0`, • indeks ujścia: 1, • krawędzie: `{(0,1,15);(1,0,9);(0,1,7);(1,0,4)}`
Oczekiwany wynik	`Exception`
Wykonawca	RW

4.1.4 Sieć z 1 węzłem pośrednim

Testy na sieciach z jednym węzłem pośrednim zostały zgrupowane w klasie `rw.blackbox.SingleVertexTest`.

TestCase	`rw.blackbox.SingleVertexTest.testCase1a()`
ID	C.1.4.1
Dane wejściowe	• liczba węzłów w sieci: 3, • indeks źródła: `0`, • indeks ujścia: 2, • krawędzie: `{(0,1,512);(1,2,126)}`
Oczekiwany wynik	`126`
Wykonawca	RW

TestCase	`rw.blackbox.SingleVertexTest.testCase1b()`
ID	C.1.4.2
Dane wejściowe	• liczba węzłów w sieci: 3, • indeks źródła: `0`, • indeks ujścia: 2, • krawędzie: `{(0,1,104526);(1,2,75269);(1,0,1523)}`
Oczekiwany wynik	`75269`
Wykonawca	RW

TestCase	`rw.blackbox.SingleVertexTest.testCase2()`
ID	C.1.4.3
Dane wejściowe	• liczba węzłów w sieci: 3, • indeks źródła: `0`, • indeks ujścia: 2, • krawędzie: `{(0,1,104526);(1,2,75269);(1,0,1523)}`
Oczekiwany wynik	`75269`
Wykonawca	RW

TestCase	`rw.blackbox.SingleVertexTest.testCase3()`
ID	C.1.4.4
Dane wejściowe	• liczba węzłów w sieci: 3, • indeks źródła: `0`, • indeks ujścia: 2, • krawędzie: `{(1,0,325);(1,2,12)}`
Oczekiwany wynik	`0`
Wykonawca	RW

TestCase	`rw.blackbox.SingleVertexTest.testCase4()`
ID	C.1.4.5
Dane wejściowe	• liczba węzłów w sieci: 3, • indeks źródła: `0`, • indeks ujścia: 2, • krawędzie: `{(0,1,9);(2,1,13)}`
Oczekiwany wynik	`0`
Wykonawca	RW

TestCase	`rw.blackbox.SingleVertexTest.testCase5()`
ID	C.1.4.6
Dane wejściowe	• liczba węzłów w sieci: 3, • indeks źródła: `0`, • indeks ujścia: 2, • krawędzie: `{(0,1,107209);(1,2,75269);(0,2,8301)}`
Oczekiwany wynik	`83570`
Wykonawca	RW

TestCase	`rw.blackbox.SingleVertexTest.testCase6a()`
ID	C.1.4.7
Dane wejściowe	• liczba węzłów w sieci: 3, • indeks źródła: `0`, • indeks ujścia: 2, • krawędzie: `{(0,1,73);(1,2,17);(1,1,345)}`
Oczekiwany wynik	`17`
Wykonawca	RW

TestCase	`rw.blackbox.SingleVertexTest.testCase6b()`
ID	C.1.4.8
Dane wejściowe	• liczba węzłów w sieci: 3, • indeks źródła: `0`, • indeks ujścia: 2, • krawędzie: `{(0,1,73);(1,2,17);(1,1,345)}`
Oczekiwany wynik	`Exception`
Wykonawca	RW

TestCase	`rw.blackbox.SingleVertexTest.testCase7a()`
ID	C.1.4.9
Dane wejściowe	• liczba węzłów w sieci: 3, • indeks źródła: `0`, • indeks ujścia: 2, • krawędzie: `{(0,1,5);(1,2,7);(1,0,12);(2,1,3)}`
Oczekiwany wynik	`5`
Wykonawca	RW

TestCase	`rw.blackbox.SingleVertexTest.testCase7b()`
ID	C.1.4.10
Dane wejściowe	• liczba węzłów w sieci: 3, • indeks źródła: `0`, • indeks ujścia: 2, • krawędzie: `{(0,1,16);(1,2,8);(2,1,2);(1,2,13)}`
Oczekiwany wynik	`Exception`
Wykonawca	RW

TestCase	`rw.blackbox.SingleVertexTest.testCase7c()`
ID	C.1.4.11
Dane wejściowe	• liczba węzłów w sieci: 3, • indeks źródła: `0`, • indeks ujścia: 2, • krawędzie: `{(0,1,16);(1,2,8);(1,0,5);(2,1,2);(1,2,13)}`
Oczekiwany wynik	`16`
Wykonawca	RW

TestCase	`rw.blackbox.SingleVertexTest.testCase8()`
ID	C.1.4.12
Dane wejściowe	• liczba węzłów w sieci: 3, • indeks źródła: `0`, • indeks ujścia: 2, • krawędzie: `{(0,1,3)}`
Oczekiwany wynik	`0`
Wykonawca	RW

TestCase	`rw.blackbox.SingleVertexTest.testCase9()`
ID	C.1.4.13
Dane wejściowe	• liczba węzłów w sieci: 3, • indeks źródła: `0`, • indeks ujścia: 2, • krawędzie: `{(0,1,13);(0,2,5)}`
Oczekiwany wynik	`5`
Wykonawca	RW

TestCase	`rw.blackbox.SingleVertexTest.testCase10()`
ID	C.1.4.14
Dane wejściowe	• liczba węzłów w sieci: 3, • indeks źródła: `0`, • indeks ujścia: 2, • krawędzie: `{(0,1,13);(0,2,5)}`
Oczekiwany wynik	`5`
Wykonawca	RW

TestCase	`rw.blackbox.SingleVertexTest.testCase10()`
ID	C.1.4.15
Dane wejściowe	• liczba węzłów w sieci: 3, • indeks źródła: `0`, • indeks ujścia: 2, • krawędzie: `{(1,2,7)}`
Oczekiwany wynik	`0`
Wykonawca	RW

4.1.5 Sieć z 2 węzłami pośrednimi połączonymi równolegle

Testy operujące na sieciach z 2 węzłami pośrednimi połączonymi równolegle zostały zebrane w klasie `rw.blackbox.TwoParalle`

TestCase	`rw.blackbox.TwoParallelVerticesTest.testCase1()`
ID	C.1.5.1
Dane wejściowe	• liczba węzłów w sieci: 4, • indeks źródła: `0`, • indeks ujścia: 3, • krawędzie: `{(0,1,1);(0,2,4);(1,3,2);(2,3,3)}`
Oczekiwany wynik	4
Wykonawca	RW

TestCase	`rw.blackbox.TwoParallelVerticesTest.testCase2()`
ID	C.1.5.2
Dane wejściowe	• liczba węzłów w sieci: 4, • indeks źródła: `0`, • indeks ujścia: 3, • krawędzie: `{(0,1,7);(0,2,1);(1,3,5);(2,3,3);(1,2,9)}`
Oczekiwany wynik	8
Wykonawca	RW

TestCase	`rw.blackbox.TwoParallelVerticesTest.testCase3()`
ID	C.1.5.3
Dane wejściowe	• liczba węzłów w sieci: 4, • indeks źródła: `0`, • indeks ujścia: 3, • krawędzie: `{(0,1,3);(0,2,6);(1,3,10);(2,3,4);(2,1,2)}`
Oczekiwany wynik	9
Wykonawca	RW

TestCase	`rw.blackbox.TwoParallelVerticesTest.testCase4()`
ID	C.1.5.4
Dane wejściowe	• liczba węzłów w sieci: 4, • indeks źródła: `0`, • indeks ujścia: 3, • krawędzie: `{(0,1,17);(0,2,11);(1,3,5);(2,3,3);(2,1,9),(1,2,6)}`
Oczekiwany wynik	8
Wykonawca	RW

TestCase	`rw.blackbox.TwoParallelVerticesTest.testCase5()`
ID	C.1.5.5
Dane wejściowe	• liczba węzłów w sieci: 4, • indeks źródła: `0`, • indeks ujścia: 3, • krawędzie: `{(0,1,5);(0,2,3);(1,3,4);(2,3,9);(0,3,3)}`
Oczekiwany wynik	`10`
Wykonawca	RW

TestCase	`rw.blackbox.TwoParallelVerticesTest.testCase6()`
ID	C.1.5.6
Dane wejściowe	• liczba węzłów w sieci: 4, • indeks źródła: `0`, • indeks ujścia: 3, • krawędzie: `{(0,1,2);(0,2,5);(1,3,8);(2,3,1);(3,0,4)}`
Oczekiwany wynik	3
Wykonawca	RW

TestCase	`rw.blackbox.TwoParallelVerticesTest.testCase7()`
ID	C.1.5.7
Dane wejściowe	• liczba węzłów w sieci: 4, • indeks źródła: `0`, • indeks ujścia: 3, • krawędzie: `{(0,1,1);(0,2,9);(1,3,12);(2,3,4);(3,1,3)}`
Oczekiwany wynik	5
Wykonawca	RW

TestCase	`rw.blackbox.TwoParallelVerticesTest.testCase8()`
ID	C.1.5.8
Dane wejściowe	• liczba węzłów w sieci: 4, • indeks źródła: `0`, • indeks ujścia: 3, • krawędzie: `{(0,1,8);(0,2,3);(1,3,5);(2,3,7);(2,1,6)}`
Oczekiwany wynik	8
Wykonawca	RW

TestCase	`rw.blackbox.TwoParallelVerticesTest.testCase9()`
ID	C.1.5.9
Dane wejściowe	• liczba węzłów w sieci: 4, • indeks źródła: `0`, • indeks ujścia: 3, • krawędzie: `{(0,1,7);(0,2,8);(1,3,2);(2,3,3);(3,1,11),(1,0,8)}`
Oczekiwany wynik	5
Wykonawca	RW

4.1.6 Sieć z 2 węzłami pośrednimi połączonymi szeregowo

Testy wykorzystujące sieci z 2 węzłami pośrednimi znajdują się z kolei w klasie `rw.blackbox.TwoSerialVerticesTest`.

TestCase	`rw.blackbox.TwoSerialVerticesTest.testCase1()`
ID	C.1.6.1
Dane wejściowe	• liczba węzłów w sieci: 4, • indeks źródła: `0`, • indeks ujścia: 3, • krawędzie: `{(0,1,5);(1,2,3);(2,3,7)}`
Oczekiwany wynik	15
Wykonawca	RW

TestCase	`rw.blackbox.TwoSerialVerticesTest.testCase2a()`
ID	C.1.6.2
Dane wejściowe	• liczba węzłów w sieci: 4, • indeks źródła: `0`, • indeks ujścia: 3, • krawędzie: `{(0,1,15);(1,2,1);(2,3,21);(1,2,5);(1,2,3)}`
Oczekiwany wynik	15
Wykonawca	RW

TestCase	`rw.blackbox.TwoSerialVerticesTest.testCase2b()`
ID	C.1.6.3
Dane wejściowe	• liczba węzłów w sieci: 4, • indeks źródła: `0`, • indeks ujścia: 3, • krawędzie: `{(0,1,15);(1,2,1);(2,3,21);(1,2,5);(1,2,7);(1,2,3)}`
Oczekiwany wynik	`Exception`
Wykonawca	RW

TestCase	`rw.blackbox.TwoSerialVerticesTest.testCase3()`
ID	C.1.6.4
Dane wejściowe	• liczba węzłów w sieci: 4, • indeks źródła: `0`, • indeks ujścia: 3, • krawędzie: `{(0,1,3);(1,2,0);(2,3,4)}`
Oczekiwany wynik	`0`
Wykonawca	RW

TestCase	`rw.blackbox.TwoSerialVerticesTest.testCase4()`
ID	C.1.6.5
Dane wejściowe	• liczba węzłów w sieci: 4, • indeks źródła: `0`, • indeks ujścia: 3, • krawędzie: `{(0,1,7);(1,2,-1);(2,3,8)}`
Oczekiwany wynik	`0`
Wykonawca	RW

TestCase	`rw.blackbox.TwoSerialVerticesTest.testCase5()`
ID	C.1.6.6
Dane wejściowe	• liczba węzłów w sieci: 4, • indeks źródła: `0`, • indeks ujścia: 3, • krawędzie: `{(0,1,12);(1,2,4);(2,3,15);(2,1,6)}`
Oczekiwany wynik	`4`
Wykonawca	RW

TestCase	`rw.blackbox.TwoSerialVerticesTest.testCase6()`
ID	C.1.6.7
Dane wejściowe	• liczba węzłów w sieci: 4, • indeks źródła: `0`, • indeks ujścia: 3, • krawędzie: `{(0,1,9);(1,2,7);(2,3,2);(2,1,5)}`
Oczekiwany wynik	7
Wykonawca	RW

TestCase	`rw.blackbox.TwoSerialVerticesTest.testCase7()`
ID	C.1.6.8
Dane wejściowe	• liczba węzłów w sieci: 4, • indeks źródła: `0`, • indeks ujścia: 3, • krawędzie: `{(0,1,11);(1,2,3);(2,3,10);(0,2,6)}`
Oczekiwany wynik	9
Wykonawca	RW

4.1.7 Sieć z nieistniejącymi węzłami

4.1.8 Sieć z ujemną liczbą węzłów

Częściowo zrealizowano w klasie `rw.blackbox.NegativeSizeNetworkTest`. Nie jest możliwe stworzenie sieci z ujemną ilością węzłów.

4.1.9 Sieć z $1e9$ węzłów

Ograniczono wielkość testowanej sieci do $1e3$ węzłów, zrealizowano w klasie `rw.blackbox.HugeFlowNetworkTest`. Ręczne stworzenie sieci przepływu tej wielkości jest nierealne. Metoda generateEdges z klasy FlowNetworkGenerator ma złożoność rzędu $O(n^3)$, co znacząco ogranicza możliwości w tym zakresie.

4.2 Testowanie biało-skrzynkowe

4.2.1 Klasa testująca `algs.network.ToStringTest`

Klasa `algs.network.ToStringTest` zawiera testy dla metody `algs.network.VertexStructure.toString()`.

TestCase	`algs.network.ToStringTest.testEmpty()`
ID	B.1.1.1
Dane wejściowe	Struktura bez zdefiniowanych krawędzi wejścia/wyjścia.
Oczekiwany wynik	”`forward:List[0]`, `backward:List[0]`”
Wykonawca	MO

TestCase	`algs.network.ToStringTest.testOneEdge()`
ID	B.1.1.2
Dane wejściowe	1. Krawędź wyjścia `(1, 2, 1)`. 2. Krawędź wejścia `(2, 1, 1)`.
Oczekiwany wynik	”`forward:List[1]: [0] -> [1] 0/1 @ 0,` `backward:List[1]: [0] -> [1] 0/1 @ 0`”
Wykonawca	MO

TestCase	`algs.network.ToStringTest.testThreeEdges()`
ID	B.1.1.3
Dane wejściowe	1. Trzy krawędzie wejścia `{(0, 3, 1), (1, 3, 1), (2, 3, 1)}`. 2. Trzy krawędzie wyjścia `{(3, 4, 1), (3, 5, 1), (3, 6, 1)}`.
Oczekiwany wynik	”`forward:List[3]: [0] -> [1] 0/1 @ 0, backward:List[3]: [0] -> [1] 0/1 @ 0`”
Wykonawca	MO

4.2.2 Klasa testująca `algs.network.MinimalNetworkConstructorTest`

Klasa `algs.network.MinimalNetworkConstructorTest` zawiera testy dla minimalnego konstruktora `algs.network.FlowNe`
`sourceIndex, int sinkIndex, int numVertices)`.

TestCase	`algs.network.MinimalNetworkConstructorTest.validArgumentTest()`
ID	B.1.2.1
Dane wejściowe	1. `sourceIndex = 0` 2. `sinkIndex = 1` 3. `numVertices = 2`
Oczekiwany wynik	1. `sourceIndex = 0` 2. `sinkIndex = 1` 3. `numVertices = 2`
Wykonawca	MO

TestCase	`algs.network.MinimalNetworkConstructorTest.invalidArgumentsTest()`
ID	B.1.2.2
Dane wejściowe	1. `sourceIndex = -1` 2. `sinkIndex = -2` 3. `numVertices = -2`
Oczekiwany wynik	1. `IllegalArgumentException`
Wykonawca	MO

TestCase	`algs.network.MinimalNetworkConstructorTest.sinkBeforeSourceTest()`
ID	B.1.2.3
Dane wejściowe	1. `sourceIndex = 1` 2. `sinkIndex = 0` 3. `numVertices = 2`
Oczekiwany wynik	1. `IllegalArgumentException`
Wykonawca	MO

TestCase	`algs.network.MinimalNetworkConstructorTest.tooFewVerticesTest()`
ID	B.1.2.4
Dane wejściowe	1. `sourceIndex = 0` 2. `sinkIndex = 7` 3. `numVertices = 2`
Oczekiwany wynik	1. `IllegalArgumentException`
Wykonawca	MO

4.2.3 Klasa testująca `algs.network.NetworkConstructorTest`

Klasa `algs.network.MinimalNetworkConstructorTest` zawiera testy dla konstruktora `algs.network.FlowNetworkArray(int numVertices, int sourceIndex, int sinkIndex, Iterator<EdgeInfo> edges)`.

TestCase	`algs.network.NetworkConstructorTest.validArgumentsTest()`
ID	B.1.3.1
Dane wejściowe	1. `numVertices = 4` 2. `sourceIndex = 0` 3. `sinkIndex = 3` 4. `edges = {(0, 1, 3); (1, 2, 2); (2, 3, 2); (0, 2, 3)}`
Oczekiwany wynik	1. `numVertices = 4` 2. `sourceIndex = 0` 3. `sinkIndex = 3` 4. `edges = {(0, 1, 3); (1, 2, 2); (2, 3, 2); (0, 2, 3)}`
Wykonawca	MO

TestCase	`algs.network.NetworkConstructorTest.invalidNumVerticesTest()`
ID	B.1.3.2
Dane wejściowe	1. `numVertices = -4` 2. `sourceIndex = 0` 3. `sinkIndex = 3` 4. `edges = {(0, 1, 3); (1, 2, 2); (2, 3, 2); (0, 2, 3)}`
Oczekiwany wynik	1. `IllegalArgumentException`
Wykonawca	MO

TestCase	`algs.network.NetworkConstructorTest.invalidSourceIndexTest()`
ID	B.1.3.3
Dane wejściowe	1. `numVertices = 4` 2. `sourceIndex = -1` 3. `sinkIndex = 3` 4. `edges = {(0, 1, 3); (1, 2, 2); (2, 3, 2); (0, 2, 3)}`
Oczekiwany wynik	1. `IllegalArgumentException`
Wykonawca	MO

TestCase	`algs.network.NetworkConstructorTest.invalidSinkIndexTest()`
ID	B.1.3.4
Dane wejściowe	1. `numVertices = 4` 2. `sourceIndex = 0` 3. `sinkIndex = -3` 4. `edges = {(0, 1, 3); (1, 2, 2); (2, 3, 2); (0, 2, 3)}`
Oczekiwany wynik	1. `IllegalArgumentException`
Wykonawca	MO

TestCase	`algs.network.NetworkConstructorTest.tooFewVerticesTest()`
ID	B.1.3.5
Dane wejściowe	1. `numVertices = 2` 2. `sourceIndex = 0` 3. `sinkIndex = 3` 4. `edges = {(0, 1, 3); (1, 2, 2); (2, 3, 2); (0, 2, 3)}`
Oczekiwany wynik	1. `IllegalArgumentException`
Wykonawca	MO

TestCase	`algs.network.NetworkConstructorTest.sinkBeforeSourceTest()`
ID	B.1.3.6
Dane wejściowe	1. `numVertices = 4` 2. `sourceIndex = 3` 3. `sinkIndex = 0` 4. `edges = {(0, 1, 3);(1, 2, 2); (2, 3, 2); (0, 2, 3)}`
Oczekiwany wynik	1. `IllegalArgumentException`
Wykonawca	MO

TestCase	`algs.network.NetworkConstructorTest.sinkBeforeSourceTest()`
ID	B.1.3.7
Dane wejściowe	1. `numVertices = 4` 2. `sourceIndex = 0` 3. `sinkIndex = 3` 4. `edges = pusty Iterator`
Oczekiwany wynik	1. `numVertices = 4` 2. `sourceIndex = 0` 3. `sinkIndex = 3` 4. `edges = pusta tablica EdgeInfo[4][4]`
Wykonawca	MO

4.2.4 Klasa testująca `algs.network.IllegalStateExceptionTest`

Klasa `algs.network.IllegalStateExceptionTest` zawiera testy weryfikujące poprawne wywoływanie wyjątków przez metodę `algs.network.FlowNetworkArray.validate()`.

TestCase	`algs.network.IllegalStateExceptionTest.moreFlowThanCapacityTest()`
ID	B.1.4.1
Dane wejściowe	1. `krawędź (1, 2, 1)` 2. `przepływ = 2`
Oczekiwany wynik	1. `IllegalStateException`
Wykonawca	MO

TestCase	`algs.network.IllegalStateExceptionTest.flowConservationTest()`
ID	B.1.4.2
Dane wejściowe	1. `krawędź (1, 2, 1)` 2. `przepływ = 2`
Oczekiwany wynik	1. `IllegalStateException`
Wykonawca	MO

4.3 Testowanie z wykorzystaniem obiektów pozornych

4.3.1 Klasa testująca `algs.network.ComputeTest`

Klasa `algs.network.ComputeTest` zawiera testy dla metody `algs.network.FordFulkerson.compute()`.

TestCase	`algs.network.ToStringTest.testFindAugmentingPath()`
ID	P.1.1.1
Dane wejściowe	numVertices = 6; srcIndex = 0; sinkIndex = 5;
Oczekiwany wynik	Wywołanie 4 razy metody findAugmentingPath.
Wykonawca	MM

4.3.2 Klasa testująca `algs.network.PopulateTest`

Klasa `algs.network.PopulateTest` zawiera testy dla metody `algs.network.FlowNetworkArray.populate(Iterator<EdgeInfo>edges)`.

TestCase	`algs.network.PopulateTest.testNumberOfCalls_hasNext()`
ID	P.1.2.1
Dane wejściowe	iterator po tablicy ośmiu wierzchołków.
Oczekiwany wynik	8 wywołań metody next
Wykonawca	MM

5 Wykonanie testów

5.1 Testowanie czarno-skrzynkowe

Testy czarno-skrzynkowe są załączone w zestawie `BlackBoxTests.xml` z pakietu `rw.blackbox`.

5.1.1 Sieć bez węzłów

Wszystkie testy na sieci bez węzłów zrealizowano w klasie `EmptyNetworkTest`.

TestCase	`rw.blackbox.EmptyNetworkTest.testCase1()`
ID	C.1.1.1
Oczekiwany wynik	`0`
Uzyskany wynik	`0`
Rezultat	TEST ZALICZONY

TestCase	`rw.blackbox.EmptyNetworkTest.testCase1()`
ID	C.1.1.2
Oczekiwany wynik	`Exception`
Uzyskany wynik	`Exception`
Rezultat	TEST ZALICZONY

5.1.2 Sieć z 1 węzłem

Wszystkie testy zrealizowane w klasie `JustOneElementTest`.

TestCase	`rw.blackbox.JustOneElementTest.testCase1()`
ID	C.1.2.1
Opis	Samego źródła.
Oczekiwany wynik	`0`
Uzyskany wynik	`0`
Rezultat	TEST ZALICZONY

TestCase	`rw.blackbox.JustOneElementTest.testCase2()`
ID	C.1.2.2
Opis	Węzła pośredniego.
Oczekiwany wynik	`Exception`
Uzyskany wynik	`Exception`
Rezultat	TEST ZALICZONY

TestCase	`rw.blackbox.JustOneElementTest.testCase3()`
ID	C.1.2.3
Opis	Ujścia.
Oczekiwany wynik	`Exception`
Uzyskany wynik	`Exception`
Rezultat	TEST ZALICZONY

TestCase	`rw.blackbox.JustOneElementTest.testCase4()`
ID	C.1.2.4
Oczekiwany wynik	`0`
Uzyskany wynik	`0`
Rezultat	TEST ZALICZONY

TestCase	`rw.blackbox.JustOneElementTest.testCase5()`
ID	C.1.2.5
Oczekiwany wynik	`0`
Uzyskany wynik	`0`
Rezultat	TEST ZALICZONY

5.1.3 Sieć bez węzłów pośrednich

Testy na sieciach bez węzłów pośrednich zrealizowano w klasie `JustSourceAndSinkTest`.

TestCase	`rw.blackbox.JustSourceAndSinkTest.testCase1()`
ID	C.1.3.1
Oczekiwany wynik	`0`
Uzyskany wynik	`0`
Rezultat	TEST ZALICZONY

TestCase	`rw.blackbox.JustSourceAndSinkTest.testCase2()`
ID	C.1.3.2
Oczekiwany wynik	`9`
Uzyskany wynik	`9`
Rezultat	TEST ZALICZONY

TestCase	`rw.blackbox.JustSourceAndSinkTest.testCase3a()`
ID	C.1.3.3
Oczekiwany wynik	`57956756`
Uzyskany wynik	`57956756`
Rezultat	TEST ZALICZONY

TestCase	`rw.blackbox.JustSourceAndSinkTest.testCase3b()`
ID	C.1.3.4
Oczekiwany wynik	`Exception`
Uzyskany wynik	`brak wyjątku`
Rezultat	TEST NIE ZALICZONY

TestCase	`rw.blackbox.JustSourceAndSinkTest.testCase4a()`
ID	C.1.3.5
Oczekiwany wynik	`346723`
Uzyskany wynik	`346723`
Rezultat	TEST ZALICZONY

TestCase	`rw.blackbox.JustSourceAndSinkTest.testCase4b()`
ID	C.1.3.6
Oczekiwany wynik	`Exception`
Uzyskany wynik	`brak wyjątku`
Rezultat	TEST NIE ZALICZONY

TestCase	`rw.blackbox.JustSourceAndSinkTest.testCase5()`
ID	C.1.3.7
Oczekiwany wynik	`0`
Uzyskany wynik	`0`
Rezultat	TEST ZALICZONY

TestCase	`rw.blackbox.JustSourceAndSinkTest.testCase6a()`
ID	C.1.3.8
Oczekiwany wynik	`0`
Uzyskany wynik	`0`
Rezultat	TEST ZALICZONY

TestCase	`rw.blackbox.JustSourceAndSinkTest.testCase6b()`
ID	C.1.3.9
Oczekiwany wynik	`Exception`
Uzyskany wynik	`Exception`
Rezultat	TEST ZALICZONY

TestCase	`rw.blackbox.JustSourceAndSinkTest.testCase7()`
ID	C.1.3.10
Oczekiwany wynik	`0`
Uzyskany wynik	`0`
Rezultat	TEST ZALICZONY

TestCase	`rw.blackbox.JustSourceAndSinkTest.testCase8a()`
ID	C.1.3.11
Oczekiwany wynik	`0`
Uzyskany wynik	`0`
Rezultat	TEST ZALICZONY

TestCase	`rw.blackbox.JustSourceAndSinkTest.testCase8b()`
ID	C.1.3.12
Oczekiwany wynik	`Exception`
Uzyskany wynik	`brak wyjątku`
Rezultat	TEST NIE ZALICZONY

TestCase	`rw.blackbox.JustSourceAndSinkTest.testCase9a()`
ID	C.1.3.13
Oczekiwany wynik	`18`
Uzyskany wynik	`4`
Rezultat	TEST NIE ZALICZONY

TestCase	`rw.blackbox.JustSourceAndSinkTest.testCase9b()`
ID	C.1.3.14
Oczekiwany wynik	`Exception`
Uzyskany wynik	`brak wyjątku`
Rezultat	TEST NIE ZALICZONY

TestCase	`rw.blackbox.JustSourceAndSinkTest.testCase10a()`
ID	C.1.3.15
Oczekiwany wynik	`0`
Uzyskany wynik	`0`
Rezultat	TEST ZALICZONY

TestCase	`rw.blackbox.JustSourceAndSinkTest.testCase10b()`
ID	C.1.3.16
Oczekiwany wynik	`Exception`
Uzyskany wynik	`brak wyjątku`
Rezultat	TEST NIE ZALICZONY

TestCase	`rw.blackbox.JustSourceAndSinkTest.testCase11a()`
ID	C.1.3.17
Oczekiwany wynik	`4`
Uzyskany wynik	`4`
Rezultat	TEST ZALICZONY

TestCase	`rw.blackbox.JustSourceAndSinkTest.testCase11b()`
ID	C.1.3.18
Oczekiwany wynik	`13`
Uzyskany wynik	`7`
Rezultat	TEST NIE ZALICZONY

TestCase	`rw.blackbox.JustSourceAndSinkTest.testCase11c()`
ID	C.1.3.19
Oczekiwany wynik	`Exception`
Uzyskany wynik	`brak wyjątku`
Rezultat	TEST NIE ZALICZONY

5.1.4 Sieć z 1 węzłem pośrednim

Testy na sieciach z jednym węzłem pośrednim zostały zgrupowane w klasie SingleVertexTest.

TestCase	`rw.blackbox.SingleVertexTest.testCase1a()`
ID	C.1.4.1
Oczekiwany wynik	`126`
Uzyskany wynik	`126`
Rezultat	TEST ZALICZONY

TestCase	`rw.blackbox.SingleVertexTest.testCase1b()`
ID	C.1.4.2
Oczekiwany wynik	`75269`
Uzyskany wynik	`75269`
Rezultat	TEST ZALICZONY

TestCase	`rw.blackbox.SingleVertexTest.testCase2()`
ID	C.1.4.3
Oczekiwany wynik	`0`
Uzyskany wynik	`0`
Rezultat	TEST ZALICZONY

TestCase	`rw.blackbox.SingleVertexTest.testCase3()`
ID	C.1.4.4
Oczekiwany wynik	`0`
Uzyskany wynik	`0`
Rezultat	TEST ZALICZONY

TestCase	`rw.blackbox.SingleVertexTest.testCase4()`
ID	C.1.4.5
Oczekiwany wynik	`0`
Uzyskany wynik	`0`
Rezultat	TEST ZALICZONY

TestCase	`rw.blackbox.SingleVertexTest.testCase5()`
ID	C.1.4.6
Oczekiwany wynik	`83570`
Uzyskany wynik	`83570`
Rezultat	TEST ZALICZONY

TestCase	`rw.blackbox.SingleVertexTest.testCase6a()`
ID	C.1.4.7
Oczekiwany wynik	`17`
Uzyskany wynik	`17`
Rezultat	TEST ZALICZONY

TestCase	`rw.blackbox.SingleVertexTest.testCase6b()`
ID	C.1.4.8
Oczekiwany wynik	`Exception`
Uzyskany wynik	`brak wyjątku`
Rezultat	TEST NIE ZALICZONY

TestCase	`rw.blackbox.SingleVertexTest.testCase7a()`
ID	C.1.4.9
Oczekiwany wynik	`5`
Uzyskany wynik	`5`
Rezultat	TEST ZALICZONY

TestCase	`rw.blackbox.SingleVertexTest.testCase7b()`
ID	C.1.4.10
Oczekiwany wynik	`Exception`
Uzyskany wynik	`brak wyjątku`
Rezultat	TEST NIE ZALICZONY

TestCase	`rw.blackbox.SingleVertexTest.testCase7c()`
ID	C.1.4.11
Oczekiwany wynik	16
Uzyskany wynik	13
Rezultat	TEST NIE ZALICZONY

TestCase	`rw.blackbox.SingleVertexTest.testCase8()`
ID	C.1.4.12
Oczekiwany wynik	`0`
Uzyskany wynik	`0`
Rezultat	TEST ZALICZONY

TestCase	`rw.blackbox.SingleVertexTest.testCase9()`
ID	C.1.4.13
Oczekiwany wynik	5
Uzyskany wynik	5
Rezultat	TEST ZALICZONY

TestCase	`rw.blackbox.SingleVertexTest.testCase10()`
ID	C.1.4.12
Oczekiwany wynik	`0`
Uzyskany wynik	`0`
Rezultat	TEST ZALICZONY

5.1.5 Sieć z 2 węzłami pośrednimi połączonymi równolegle

Testy operujące na sieciach z 2 węzłami pośrednimi połączonymi równolegle zostały zebrane w klasie `TwoParallelVerticesTest`.

TestCase	`rw.blackbox.TwoParallelVerticesTest.testCase1()`
ID	C.1.5.1
Oczekiwany wynik	4
Uzyskany wynik	4
Rezultat	TEST ZALICZONY

TestCase	`rw.blackbox.TwoParallelVerticesTest.testCase2()`
ID	C.1.5.2
Oczekiwany wynik	8
Uzyskany wynik	8
Rezultat	TEST ZALICZONY

TestCase	`rw.blackbox.TwoParallelVerticesTest.testCase2()`
ID	C.1.5.3
Oczekiwany wynik	8
Uzyskany wynik	8
Rezultat	TEST ZALICZONY

TestCase	`rw.blackbox.TwoParallelVerticesTest.testCase2()`
ID	C.1.5.4
Oczekiwany wynik	9
Uzyskany wynik	9
Rezultat	TEST ZALICZONY

TestCase	`rw.blackbox.TwoParallelVerticesTest.testCase3()`
ID	C.1.5.5
Oczekiwany wynik	9
Uzyskany wynik	9
Rezultat	TEST ZALICZONY

TestCase	`rw.blackbox.TwoParallelVerticesTest.testCase4()`
ID	C.1.5.6.
Oczekiwany wynik	8
Uzyskany wynik	8
Rezultat	TEST ZALICZONY

TestCase	`rw.blackbox.TwoParallelVerticesTest.testCase5()`
ID	C.1.5.7.
Oczekiwany wynik	8
Uzyskany wynik	8
Rezultat	TEST ZALICZONY

TestCase	`rw.blackbox.TwoParallelVerticesTest.testCase6()`
ID	C.1.5.8
Oczekiwany wynik	3
Uzyskany wynik	3
Rezultat	TEST ZALICZONY

TestCase	`rw.blackbox.TwoParallelVerticesTest.testCase7()`
ID	C.1.5.9
Oczekiwany wynik	5
Uzyskany wynik	5
Rezultat	TEST ZALICZONY

TestCase	`rw.blackbox.TwoParallelVerticesTest.testCase8()`
ID	C.1.5.9
Oczekiwany wynik	8
Uzyskany wynik	8
Rezultat	TEST ZALICZONY

TestCase	`rw.blackbox.TwoParallelVerticesTest.testCase9()`
ID	C.1.5.10
Oczekiwany wynik	5
Uzyskany wynik	5
Rezultat	TEST ZALICZONY

5.1.6 Sieć z 2 węzłami pośrednimi połączonymi szeregowo

Testy wykorzystujące sieci z 2 węzłami pośrednimi znajdują się z kolei w klasie `TwoSerialVerticesTest`.

TestCase	`rw.blackbox.TwoSerialVerticesTest.testCase1()`
ID	C.1.6.1
Oczekiwany wynik	`3`
Uzyskany wynik	`3`
Rezultat	TEST ZALICZONY

TestCase	`rw.blackbox.TwoSerialVerticesTest.testCase2()`
ID	C.1.6.2
Oczekiwany wynik	`15`
Uzyskany wynik	`3`
Rezultat	TEST NIE ZALICZONY

TestCase	`rw.blackbox.TwoSerialVerticesTest.testCase3()`
ID	C.1.6.3
Oczekiwany wynik	`0`
Uzyskany wynik	`0`
Rezultat	TEST ZALICZONY

TestCase	`rw.blackbox.TwoSerialVerticesTest.testCase4()`
ID	C.1.6.4
Oczekiwany wynik	`0`
Uzyskany wynik	`0`
Rezultat	TEST ZALICZONY

TestCase	`rw.blackbox.TwoSerialVerticesTest.testCase5()`
ID	C.1.6.5
Oczekiwany wynik	`4`
Uzyskany wynik	`4`
Rezultat	TEST ZALICZONY

TestCase	`rw.blackbox.TwoSerialVerticesTest.testCase6()`
ID	C.1.6.6
Oczekiwany wynik	7
Uzyskany wynik	7
Rezultat	TEST ZALICZONY

TestCase	`rw.blackbox.TwoSerialVerticesTest.testCase7()`
ID	C.1.6.7
Oczekiwany wynik	9
Uzyskany wynik	9
Rezultat	TEST ZALICZONY

5.1.7 Sieć z nieistniejącymi węzłami

5.1.8 Sieć z ujemną liczbą węzłów

Częściowo zrealizowano w klasie NegativeSizeNetworkTest Nie jest możliwe stworzenie sieci z ujemną ilością węzłów.

5.1.9 Sieć z $1e9$ węzłów

Ograniczono wielkość testowanej sieci do $1e3$ węzłów, zrealizowano w klasie HugeFlowNetworkTest Ręczne stworzenie sieci przepływu tej wielkości jest nierealne. Metoda generateEdges z klasy FlowNetworkGenerator ma złożoność rzędu $O(n^3)$, co znacząco ogranicza możliwości w tym zakresie.

5.2 Testowanie biało-skrzynkowe

5.2.1 Test case `ToStringTest`

`ToStringTest`	
Data	08.01.13 15:31
Ilość wykonanych testów	3
Ilość porażek	0
Uwagi	Brak
Wykonawca	MO

5.2.2 Test case `MinimalnetworkConstructorTest`

`MinimalNetworkConstructorTest`	
Data	08.01.13 15:33
Ilość wykonanych testów	4
Ilość porażek	4
Uwagi	Brak walidacji danych wejściowych. Błędy w kodzie. Id testów które nie przeszły: • B.1.2.1 • B.1.2.2 • B.1.2.3 • B.1.2.4
Wykonawca	MO

5.2.3 Test case `NetworkConstructorTest`

`NetworkConstructorTest`	
Data	08.01.13 15:42
Liczba wykonanych testów	7
Liczba porażek	5
Uwagi	Brak walidacji danych wejściowych. Id testów które nie przeszły: • B.1.3.2 • B.1.3.3 • B.1.3.4 • B.1.3.5 • B.1.3.6
Wykonawca	MO

5.2.4 Test case `IllegalStateExceptionTest`

`IllegalStateExceptionTest`	
Data	08.01.13 16:05
Liczba wykonanych testów	2
Ilość porażek	0
Uwagi	Brak
Wykonawca	MO

5.3 Testowanie z wykorzystaniem obiektów pozornych

5.3.1 Test case PopulateTest

PopulateTest	
Data	08.01.13 15:31
Ilość wykonanych testów	1
Ilość porażek	0
Uwagi	Brak
Wykonawca	MM

5.3.2 Test case ComputeTest

ComputeTest	
Data	08.01.13 18:43
Ilość wykonanych testów	1
Ilość porażek	0
Uwagi	Brak
Wykonawca	MM

6 Ocena rezultatów testów

6.1 Testowanie czarno-skrzynkowe

Testy czarnoskrzynkowe można uruchomić z poziomu testSuite'a BlackBoxText.xml w pakiecie rw.blackbox.

6.1.1 Sieć bez węzłów pośrednich

Testy na sieciach bez węzłów pośrednich zrealizowano w klasie JustSourceAndSinkTest.

- asdasdkajslkfjhasldkjfhalkjshdfkjahsdfkljahsdlkfjhasldkjfh. `testCase3b` Oczekiwany wynik: Exception Uzyskany wynik: brak wyjątku TEST NIE ZALICZONY
  Program nie podnosi wyjątku w związku z istnieniem pętli w grafie.
  `testCase8b` Oczekiwany wynik: Exception Uzyskany wynik: brak wyjątku TEST NIE ZALICZONY
  Algorytm poprawnie nie znajduje niezerowego przepływu maksymalnego w sieci bez ścieżki o dodatniej przepustowości od źródła do ujścia. Nie jest podnoszony wyjątek w związku z występowaniem krawędzi o ujemnej przepustowości.
- Z wieloma krawędziami skierowanymi od źródła do ujścia. `testCase9a` Oczekiwany wynik: 18 Uzyskany wynik: 4 TEST NIE ZALICZONY
  `testCase9b` Oczekiwany wynik: Exception Uzyskany wynik: brak wyjątku TEST NIE ZALICZONY

Algorytm nie odczytuje poprawnie przepustowości ze zwielokrotnionych krawędzi. Nie jest podnoszony wyjątek w związku z występowaniem zwielokrotnionych krawędzi.
`testCase10b` Oczekiwany wynik: Exception Uzyskany wynik: brak wyjątku
Algorytm poprawnie nie znajduje niezerowego przepływu maksymalnego w sieci bez ścieżki o dodatniej przepustowości od źródła do ujścia. Nie jest podnoszony wyjątek mimo występowania krawędzi o ujemnej przepustowości i krawędzi zwielokrotnionych.
`testCase11b` Oczekiwany wynik: 13 Uzyskany wynik: 7 TEST NIE ZALICZONY
`testCase11c` Oczekiwany wynik: Exception Uzyskany wynik: brak wyjątku TEST NIE ZALICZONY
Algorytm poprawnie znajduje przepływ maksymalny w sieci z dodatkową krawędzią o tym samym kierunku, a przeciwnym zwrocie. W przypadku zwielokrotnienia, którejkolwiek z tych krawędzi, algorytm zwraca nieprawidłową wartość przepływu, nie podnosząc wyjątku w związku z występowaniem krawędzi o ujemnej przepustowości.

6.1.2 Sieć z 1 węzłem pośrednim

Testy na sieciach z jednym węzłem pośrednim zostały zgrupowane w klasie SingleVertexTest.

- asdasdasdqweqwe
  `testCase6b` Oczekiwany wynik: Exception Uzyskany wynik: brak wyjątku
  Algorytm poprawnie znajduje maksymalny przepływ w sieci. Algorytm nie informuje o istnieniu pętli w sieci.
- Połączone zwielokrotnionymi krawędziami ze źródła do węzła pośredniego i z węzła pośredniego do źródła, z mieszanymi zwrotami.
  `testCase7b` Oczekiwany wynik: Exception Uzyskany wynik: brak wyjątku TEST NIE ZALICZONY
  `testCase7c` Oczekiwany wynik: 16 Uzyskany wynik: 13 TEST NIE ZALICZONY

6.1.3 Sieć z 2 węzłami pośrednimi połączonymi szeregowo

Testy wykorzystujące sieci z 2 węzłami pośrednimi znajdują się z kolei w klasie TwoSerialVerticesTest.

- Sieć z 2 węzłami pośrednimi połączonymi szeregowo z wieloma krawędziami pomiędzy węzłami pośrednimi. `testCase2a` Oczekiwany wynik: 15 Uzyskany wynik: 3 TEST NIE ZALICZONY
  `testCase2b` Oczekiwany wynik: Exception Uzyskany wynik: brak wyjątku TEST NIE ZALICZONY
  Zwraca niepoprawną wartość. Konstruktor ani funkcja validate() z klasy FlowNetworkArray nie podnoszą wyjątków.

6.1.4 Sieć z nieistniejącymi węzłami

6.1.5 Sieć z ujemną liczbą węzłów

Częściowo zrealizowano w klasie NegativeSizeNetworkTest Nie jest możliwe stworzenie sieci z ujemną ilością węzłów.

6.1.6 Sieć z $1e9$ węzłów

Ograniczono wielkość testowanej sieci do $1e3$ węzłów, zrealizowano w klasie HugeFlowNetworkTest Ręczne stworzenie sieci przepływu tej wielkości jest nierealne. Metoda generateEdges z klasy FlowNetworkGenerator ma złożoność rzędu $O(n^3)$, co znacząco ogranicza możliwości w tym zakresie.

6.2 Testowanie biało-skrzynkowe

6.2.1 Raporty błędów

TestCase	`algs.network.MinimalNetworkConstructorTest.validArgumentTest()`
ID	B.1.2.1
Data wykonania	08.01.13 15:33
Opis błędu	Niewłaściwie przypisywane wartości
Dane wejściowe	1. sourceIndex = 0 2. sinkIndex = 1 3. numVertices = 2
Otrzymany wynik	1. sourceIndex = 2 2. sinkIndex = 1 3. numVertices = 0
Oczekiwany wynik	1. sourceIndex = 0 2. sinkIndex = 1 3. numVertices = 2
Wykonawca	MO

TestCase	`algs.network.MinimalNetworkConstructorTest.invalidArgumentTest()`
ID	B.1.2.2
Data wykonania	08.01.13 15:33
Opis błędu	Możliwe jest stworzenie obiektu o niedopuszczalnych polach
Dane wejściowe	1. sourceIndex = -1 2. sinkIndex = -2 3. numVertices = -2
Otrzymany wynik	1. sourceIndex = -1 2. sinkIndex = -2 3. numVertices = -2
Oczekiwany wynik	1. `IllegalArgumentException`
Wykonawca	MO

TestCase	`algs.network.MinimalNetworkConstructorTest.sinkBeforeSourceTest()`
ID	B.1.2.3
Data wykonania	08.01.13 15:33
Opis błędu	Możliwe jest stworzenie obiektu o niedopuszczalnych polach
Dane wejściowe	1. sourceIndex = 1 2. sinkIndex = 0 3. numVertices = 2
Otrzymany wynik	1. sourceIndex = 1 2. sinkIndex = 0 3. numVertices = 2
Oczekiwany wynik	1. `IllegalArgumentException`
Wykonawca	MO

TestCase	`algs.network.MinimalNetworkConstructorTest.tooFewVerticesTest()`
ID	B.1.2.4
Data wykonania	08.01.13 15:42
Opis błędu	Możliwe jest stworzenie obiektu o niedopuszczalnych polach
Dane wejściowe	1. sourceIndex = 0 2. sinkIndex = 7 3. numVertices = 2
Otrzymany wynik	1. sourceIndex = 0 2. sinkIndex = 7 3. numVertices = 2
Oczekiwany wynik	1. `IllegalArgumentException`
Otrzymany wynik	sourceIndex = 0; sinkIndex = 7; numVertices = 2
Oczekiwany wynik	IllegalArgumentException
Wykonawca	MO

TestCase	`algs.network.MinimalNetworkConstructorTest.invalidNumVerticesTest()`
ID	B.1.3.2
Data wykonania	08.01.13 15:42
Opis błędu	Możliwe jest stworzenie sieci o ujemniej liczbie węzłów
Dane wejściowe	1. numVertices = -4 2. sourceIndex = 0 3. sinkIndex = 3 4. edges = {(0, 1, 3); (1, 2, 2); (2, 3, 2); (0, 2, 3)}
Otrzymany wynik	1. numVertices = -4 2. sourceIndex = 0 3. sinkIndex = 3 4. edges = {(0, 1, 3); (1, 2, 2); (2, 3, 2); (0, 2, 3)}
Oczekiwany wynik	1. `IllegalArgumentException`
Wykonawca	MO

TestCase	`algs.network.MinimalNetworkConstructorTest.invalidSourceIndexTest()`
ID	B.1.3.3
Data wykonania	08.01.13 15:42
Opis błędu	Możliwe jest stworzenie sieci o ujemniej liczbie węzłów
Dane wejściowe	1. numVertices = 4 2. sourceIndex = -1 3. sinkIndex = 3 4. edges = {(0, 1, 3); (1, 2, 2); (2, 3, 2); (0, 2, 3)}
Otrzymany wynik	1. numVertices = 4 2. sourceIndex = -1 3. sinkIndex = 3 4. edges = {(0, 1, 3); (1, 2, 2); (2, 3, 2); (0, 2, 3)}
Oczekiwany wynik	1. `IllegalArgumentException`
Wykonawca	MO

TestCase	`algs.network.MinimalNetworkConstructorTest.invalidSinkIndexTest()`
ID	B.1.3.4
Data wykonania	08.01.13 15:42
Opis błędu	Możliwe jest stworzenie sieci o ujemniej liczbie węzłów
Dane wejściowe	1. numVertices = 4 2. sourceIndex = 0 3. sinkIndex = -3 4. edges = {(0, 1, 3); (1, 2, 2); (2, 3, 2); (0, 2, 3)}
Otrzymany wynik	1. numVertices = 4 2. sourceIndex = 0 3. sinkIndex = -3 4. edges = {(0, 1, 3); (1, 2, 2); (2, 3, 2); (0, 2, 3)}
Oczekiwany wynik	1. `IllegalArgumentException`
Wykonawca	MO

TestCase	`algs.network.MinimalNetworkConstructorTest.tooFewVerticesTest()`
ID	B.1.3.5
Data wykonania	08.01.13 15:42
Opis błędu	Możliwe jest stworzenie sieci o ujemniej liczbie węzłów
Dane wejściowe	1. numVertices = 2 2. sourceIndex = 0 3. sinkIndex = 3 4. edges = {(0, 1, 3); (1, 2, 2); (2, 3, 2); (0, 2, 3)}
Otrzymany wynik	1. numVertices = 2 2. sourceIndex = 0 3. sinkIndex = 3 4. edges = {(0, 1, 3); (1, 2, 2); (2, 3, 2); (0, 2, 3)}
Oczekiwany wynik	1. `IllegalArgumentException`
Wykonawca	MO

TestCase	`algs.network.MinimalNetworkConstructorTest.sinkBeforeSourceTest()`
ID	B.1.3.6
Data wykonania	08.01.13 15:42
Opis błędu	Możliwe jest stworzenie sieci o ujemniej liczbie węzłów
Dane wejściowe	1. numVertices = 4 2. sourceIndex = 3 3. sinkIndex = 0 4. edges = {(0, 1, 3);(1, 2, 2); (2, 3, 2); (0, 2, 3)}
Otrzymany wynik	1. numVertices = 4 2. sourceIndex = 3 3. sinkIndex = 0 4. edges = {(0, 1, 3);(1, 2, 2); (2, 3, 2); (0, 2, 3)}
Oczekiwany wynik	1. `IllegalArgumentException`

6.2.2 Przebieg procesu testowania

Proces testowania przebiegł zgodnie z planem. Błędy zostały skutecznie wykryte. W efekcie uzyskane zostało 100% pokrycia instrukcji.

6.3 Testowanie z wykorzystaniem obiektów pozornych

6.4 Ocena testów

Testy skupiły się na krotności wywołań metod mockowanych obiektów i na tym polu nie zaobserwowano uchybień ze strony programistów, tj. testy osiągnęły przewidywane rezultaty.

Historia dokumentu

Data	Wersja	Autor	Szczegóły
2012-12-21	0.1.0	TC	Szablon dokumentu.
2012-12-22	0.2.0	RW	Dodano plan testowania biało-skrzynkowego.
2012-12-22	0.2.1	MO	Dodano plan testowania czarno-skrzynkowego.
2012-12-25	0.2.2	TC	Połączono plany. Dodano plan ogólny. Rozróżnionio plany szczegółowe.
2012-12-28	0.2.3	MM	Dodano plan testowania z mockami.
2012-12-29	0.3	TC, MM, MO, RW	Zmergowano projekty testowania biało-, czarno--skrzynkowego oraz z wykorzystaniem mocków.
2013-01-02	0.4	RW	Dodano realizację testów czarno-skrzynkowych.
2013-01-02	0.4.1	MO	Dodano realizację testów biało-skrzynkowych.
2013-01-03	0.4.2	MO	Dodano realizację testów z mockami.
2013-01-05	0.5	TC, MM, MO, RW	Dodano opis wykonania testów.
2013-01-07	0.6	TC, MM, MO, RW	Dodano oceny rezultatów testów.
2013-01-08	1.0	TC	Sprawdzono. Zatwierdzono.