Problem Straży Pożarnej

Dokumentacja projektu

1. **Wstęp**
   1. **Cel**

Celem algorytmu jest rozwiązanie problemu optymalnego rozmieszczenia jednostek straży pożarnej tak by możliwe było utrzymanie jak najmniejszej liczby placówek przy jednoczesnym pokryciu największej możliwej ilości miast.

* 1. **Problem**

Znalezienie optymalnego rozwiązanie niesie ze sobą bardzo dużą złożoność obliczeniową. Do znalezienia najlepszego rozwiązania trzeba by rozstrzygnąć wszystkie możliwe przypadki kombinacji położeń stacji w miastach. O ile przy bardzo małym zbiorze nie stanowiło by to problemu dla przeciętnego komputera, to w przypadku o wiele większej ilości miast mogło by znacząco wydłużyć działanie algorytmu.

* 1. **Algorytm**

Ze względu na bardzo szybko zwiększający się poziom skomplikowania wraz z rosnącą ilością miast do rozwiązania problemu posłuży algorytm heurystyczny zachłanny. Nie gwarantuje on otrzymania optymalnego wyniku ale jest optymalnym kompromisem między wydajnością a dokładnością.

* 1. **Działanie Algorytmu**

K1. Dla każdego miasta znajduje wszystkie połącznie które pokryła by ewentualna jednostka straży pożarnej znajdująca się w danym mieście i zapisuje do neighboursCLoserThanR.

K2. Przechodzi przez wszystkie miasta i wybiera te które ma największą ilość takich połączeń, jeżeli jest ich więcej to wybiera pierwsze na które trafi.

K3. Wybrane miasto zapisuje do zbioru wynikowego i we wszystkich miastach usuwa z neighboursCLoserThanR te które zostały właśnie pokryte.

K4. Krok K2 oraz K3 powtarza aż neighboursCLoserThanR będzie puste.

W ten sposób otrzymywana jest tablica zawierająca rozwiązanie problemu

1. **Uruchamianie**
   1. **Otwieranie projektu**

Solucja została stworzona w środowisku InteliJ IDEA i za pomocą tego IDE zalecane jest jej uruchomienie. W tym celu po włączeniu programu należy nacisnąć po kolei:

Open -> Znaleźć lokalizację folderu FireStations i go rozwinąć -> Wybrać tytułowy folder z etykietą IntelIJ -> Nacisnąć przycisk ok

InteliJ otworzy projekt i automatycznie pobierze niezbędne biblioteki

* 1. **Konfiguracja**

Projekt będzie od razu skonfigurowany i możliwy do użycia.

U góry po prawej w menu Run/Debug dostępne będą dwie konfiguracje środowiska:

MainClass -fi -fo oraz AllTests

* + 1. MainClass -fi -fo

To konfiguracja działania algorytmu. Po jej wybraniu i uruchomieniu, włączony zostanie algorytm który pobierze dane wejściowe z pliku „in.json” i wyniki zapisze do pliku „out.json”.

* + 1. AllTests

Druga to konfiguracja testów. Po jej wyborze automatycznie rozpocznie się testowanie programu.

* 1. **Uruchamianie Algorytmu**

Klasą startową programu jest klasa MainClass. W celu uruchomienia programu należy wybrać metodę main i podać dwa niezbędne parametry

„-fi” oraz „-fo”.

„-fi” to ścieżka do pliku wejściowego z danymi.

„-fo” to ścieżka do miejsca zapisu rozwiązania

Podstawowa ścieżka odczytu znajduje się pod adresem:

..\FireStations\src\main\resources\in.json

A zapisu pod:

..\FireStations\src\main\resources\out.json

Przykład podania przełączników:

-fi <ścieżka do pliku wejściowego> -fo <ścieżka do pliku wyjściowego>

-fi ..\FireStations\src\main\resources\in.json -fo ..\FireStations\src\main\resources\out.json

* 1. **Uruchamianie Testów**

W celu uruchomienia testów wystarczy wybrać konfigurację AllTests i włączyć program, reszta wykona się automatycznie.

1. **Budowa projektu**

Projekt składa się z 3 głównych elementów:

Kodu źródłowego algorytmu

Kodu źródłowego testów i pliki używane do testów

Pliku pom.xml

* 1. **Kod źródłowy algorytmu**

Kod znajduje się w folderze:

src -> main -> java

Podstawowa ścieżka plików zapisu i odczytu znajduje się w folderze:

stc -> main -> resources

* + 1. **Streszczenie Klas**

Każda klasa jest dokładnie opisana w kodzie więc tutaj zostanie przytoczona tylko skrótowo w celu ułatwienia zapoznania się z kodem

* MainClass
* StartArgs
* City
* Road
* ConnectsCitiesWithRoad
* JSonFireStationsIO
* SCPGreedy
  1. **Kod źródłowy testów**

Kod znajduje się w folderze

Src -> test -> java

Pliki używane do testowanie znajdują się w folderze

Src -> test -> resources

* + 1. **Streszczenie Klas**

Każda klasa jest dokładnie opisana w kodzie więc tutaj zostanie przytoczona tylko skrótowo w celu ułatwienia zapoznania się z kodem

* CityTest
* ConnectsCitiesWithRoadTest
* JSonFireStationsIOTest
  1. **Plik pom.xml**

Znajduje się bezpośrednio w głównym folderze. Zawiera niezbędną informację o konfiguracji projektu (bibliotekach itp.)

1. **Pliki Wejściowe i wyjściowe**
   1. **Plik wejściowy**

Plik wejściowy zapisany jest w formacie JSon i jego struktura wygląda następująco:

{  
"miasta" : ["A", "B", "C", "D", "E", "F", "G", "H", "I"],  
"drogi" : [  
{  
"miasta" : ["A", "B"],  
"czas\_przejazdu" : 2  
},  
{  
"miasta" : ["A", "C"],  
"czas\_przejazdu" : 3  
},  
{  
"miasta" : ["A", "D"],  
"czas\_przejazdu" : 4  
},  
{  
"miasta" : ["A", "E"],  
"czas\_przejazdu" : 1  
},  
{  
"miasta" : ["B", "F"],  
"czas\_przejazdu" : 9  
},  
{  
"miasta" : ["G", "F"],  
"czas\_przejazdu" : 9  
},  
{  
"miasta" : ["H", "I"],  
"czas\_przejazdu" : 9  
}  
],  
"timeout" : 5,  
"max\_czas\_przejazdu" : 10  
}

Posiada następujące zmienne

* „miasta” - ista miast, początkowo w każdym mieście znajduje się jednostka straży pożarnej; 0 < liczba miast < 1000000
* „drogi” - list dróg; pole „miasta” zawiera informację, jakie miasta łączy droga, a pole “czas\_przejazdu” jaki jest czas przejazdu drogą; 0 < “czas\_przejazu” < 100
* „max\_czas\_przejazdu” - liczba oznaczająca jaki jest maksymalny dopuszczalny czas przejazdu straży pożarnej zanim dotrze do celu; 0 < “max\_czas\_przejazu” < 10000
* „timeout” - czas w sekundach po jakim program musi zakończyć swoje działanie (tolerancja 1%); w tym czasie program powinien zapisać najlepsze rozwiązanie jakie udało mu się znaleźć; 0 < “timeout” < 10000

* 1. **Plik wyjściowy**

Plik wejściowy zapisany jest w formacie JSon i jego struktura wygląda następująco:

{"miasta":["B","G","H"]}

Posiada tylko jedną zmienną:

* „miasta” – przechowuje miasta w których zostaną postawione jednostki straży pożarnej

1. **Uwagi**
   1. **Wielowątkowość**

Podjęto próbę uruchomienia algorytmu w trybie wielowątkowym. Mechanizm ten zastosowano w momencie najbardziej wymagających obliczeń czyli podczas obliczania wszystkich połączeń danego miasta pokrytych przez straż pożarną. Jednak ze względu na specyfikę algorytmu nie przyniosło to oczekiwanych korzyści.

Po pierwsze proces ten wymagał by sztucznej kontroli na ilością wątków ponieważ ich za duża ilość mogła by prowadzić do błędów podczas wykonywania ze względu na ograniczone zasoby maszyny.

Po drugie zastosowanie wielowątkowości ma sens tylko w tedy kiedy występują przerwy w działaniu np. przez oczekiwanie na pobranie danych. W momencie kiedy jeden wątek czeka drugi pracuje na jego miejscu. W tym algorytmie takie sytuacje niemal nie występują. Dodatkowo obsługa wątków również wymaga zasobów komputera dlatego przy większej ich ilości program zaczynał zauważalnie zwalniać.

1. **a**