AAL

Autor : Mateusz Wasiak

Projekt na przedmiot Analiza Algorytmów (AAL)
Tytuł projektu: Urząd spraw czworakich

Opis problemu:
Urząd spraw czworakich przyjmuje petentów w czterech różnych sprawach. Obsługa każdego petenta zajmuje 5 minut i jest wykonywana przez urzędnika specjalizującego się w danej sprawie. Urząd zatrudnia sześciu urzędników, każdy specjalizuje się w dwóch różnych sprawach. W urzędzie
znajduje się jedno okienko, przy którym na początku dnia siedzi urzędnik specjalizujący się w pierwszej i drugiej sprawie. Do okienka ustawia się kolejka osób z różnymi sprawami. Gdy do okienka podchodzi osoba ze sprawą inną niż ta, w której specjalizuje się urzędnik, może zostać obsłużona na dwa sposoby: 1) urzędnik przy okienku dzwoni do urzędnika specjalizującego się w tej sprawie i załatwia tę konkretną sprawę telefonicznie, przez co obsługa wydłuża się o dodatkowe 5 minut, lub 2) urzędnik wychodzi i woła na swoje miejsce (na trwałe) urzędnika specjalizującego się w tej sprawie, co także zajmuje dodatkowe 5 minut. Należy znaleźć sposób obsługi petentów minimalizujący całkowity czas obsługi. Porządek ludzi w kolejce jest znany od początku i się nie zmienia.

Skrócony opis problemu:
W kolejce są zadania 4 różnych typów. W urzędzie jest 6 pracowników specjalizujących się w dwóch różnych zadaniach. Jeżeli przy okienku jest urzędnik, który specjalizuje się w danym zadaniu to wykonuje zadanie w 5 minut. Jeśli nie to może wykonać je w 10 minut, lub zawołać kogoś innego na swoje miejsce – trwa to 10 minut (razem z obsługą aktualnego zadania przez zawołanego pracownika). Należy uszeregować pracowników tak, by czas obsługi był jak najmniejszy.

Biblioteki zewnętrzne:
Projekt wykorzystuje do wprowadzania argumentów uruchomienia bibliotekę cxxopts.hpp.

Budowanie projektu:

Projekt jest budowane za pomocą programu cmake, który należy zainstalować.

Systemy z rodziny Linux

W terminalu:

- mkdir build && cd build # stworzenie i przejście do katalogu build
- cmake ../ # uruchomienie CMake
- make # kompilacja i konsolidacja projektu
- ./AAL # uruchomienie programu

Systemy Windows

UWAGA! W miejsce $GENERATOR należy wstawić jakim kompilatorem będzie kompilowany projekt. Przykładową wartością w to miejsce może być "Visual Studio 15 2017 Win64".

W środowisku Power Shell:

- mkdir build # stworzenie katalogu build
- cd build # przejście do katalogu build
- cmake -G "$GENERATOR" ../ # uruchomienie CMake

- cmake --build . # wywołanie kompilatora za pośrednictwem CMake

Tryby uruchomienia:

Tryb 1:

Uruchamiany poprzez uruchomienie w terminalu :
`./AAL –m1`

W tym trybie dane wejściowe są wprowadzane przy pomocy konsoli. Można przekierować wejście i/lub wyjście do pliku np.

`./AAL --m1 <./in.txt`

można również uruchomić program generatora i przekazać dane z niego do programu AAL :

`./Gen -n 10 -d 3 | ./AAL –m1`

W tym wypadku generowana jest instancja o długości 10 elementów (zadań) ziarno losowości 3.

Tryb 2:

Uruchamiany poprzez uruchomienie w terminalu :
`./AAL –m2 -n 10 -d 3`

Działa analogicznie do przypadku powyższego, z tym, że nie uruchamia programu Gen. Argument -d jest nieobowiązkowy.

Tryb 3:

Uruchamiany poprzez uruchomienie w terminalu :
`./AAL –m3 -n 1000000 -s 1000000 -k 100 -r 10`

Wykonuje pomiar czasu działania algorytmu dla 10 różnych instancji problemów. Pierwsza wylosowana dana ma 1000000 elementów, krok jest o 1000000 elementów. Wykonamy k=100 takich skoków.

Pomoc programu:

Program options:

--m1	Mode with reading from I/O
--m2	Mode with generation of data, -n option is obligatory
--m3	Full test mode, -n, -k, -s, -r options are obligatory
-n, --number arg	Get number of tasks
-d, --seed arg	Get seed for generator
-s, --step arg	Get step of tasks in next iterations
-k, --kproblem arg	Get number of problems
-r, --rinstance arg	Get number of problem instantions
-h, --help	Print help

Metody rozwiązania zadanego problemu:

Metoda Liniowa – wykorzystywana jako główny algorytm do szeregowania pracowników i obliczania optymalnego czasu obsługi.
Metoda ta najpierw sprawdza czy aktualne zadanie jest obsługiwane optymalnie. Jeśli tak to zwiększa ona czas o 5, nie zmienia pracowników i przechodzi do następnego zadania.

Następnie jesteśmy w przypadku takim, że pierwsze zadanie nie jest optymalne, więc sprawdzamy czy następne zadanie też nie jest optymalne. Jeśli również nie jest, to zmieniamy pracownika przed wykonaniem pierwszego zadania. Zwiększamy czas o 10 i sprawdzamy czy następne zadania są takie same jak pierwsze zadanie by szybciej obliczyć czas potrzebny do obsługi tych zadań. Trzecim krokiem jest sprawdzenie następnych zadań gdy pierwsze zadanie nie jest optymalne, a drugie jest optymalne. Najpierw dodamy do czasu 15 bo nie ważne czy zawołamy czy nie te dwa zadania które już rozważyliśmy wykonają się w 15 minut. Należy sprawdzić tu kilka różnych wariantów zadań w kolejce. Jeżeli występują zadania, które są takie same jak to na drrugim miejscu to iterujemy po nich aż nie znajdziemy innego. Zwiększając odpowiednio czas o 5 za każde zadanie. Następnie w zależności od tego jakie inne zadanie zostało po tym iterowaniu znalezione wykonujemy różne czynności. Jeśli jest to ta samo zadanie co pierwsze to zmieniamy pracownika. Jeśli jest to zadanie, które jest innym zadaniem obsługiwanym przez danego pracownika optymalnie to nie zmieniamy pracowników. Jeżeli jest to inne zadanie – czyli takie, które nie było na początku rozważane i nie jest zadaniem rozwiązywanym optymalnie przez danego pracownika to należy sprawdzić następne zadania. Jeżeli znów znajdziemy zadanie, które było 2 w kolejce to iterujemy po nim analogicznie do jak w poprzednim kroku. W momencie gdy znajdziemy inne zadanie to : Jeśli jest to to samo zadanie co pierwsze w kolejce to zmieniamy pracownika. W przeciwnym wypadku nie zmieniamy pracowników.

Złożoność: **O(n)**

Metoda Brutalna – wykorzystywana głównie do testowania następnej metody :
Metoda ta sprawdza czy dane zadanie może być wykonane przez aktualnie pracującego pracownika. Jeśli tak to pracownik wykonuje to zadanie, a czas wykonania zadań zwiększa się o 5. W przeciwnym wypadku uruchamiamy cztery razy rekurencyjnie tą samą metodę tylko z innymi zadaniami optymalnie wykonywanymi przechodząc już do następnego zadania. Sprawdzamy jaki czas wykonania jest najmniejszy i takie uszeregowanie pracowników jest optymalne. Czas zwiększa się o 10 (ponieważ albo zawołaliśmy innego pracownika lub wykonaliśmy to zadanie samodzielnie).

Opis plików

main.cpp – pętla główna programu obsługująca dane wejściowe
Gen.cpp – program główny generatora danych Gen
Worker.cpp, Worker.h – pliki klasy Worker, która jest odwzorowaniem pracownika urzędu
InputOutput.cpp InputOutput.h – pliki klasy InputOutput, która jest odpowiedzialna za obsługę wczytywania danych do algorytmów
Generator.cpp Generator.h – klasa do generowania danych
Algoritms.cpp Algoritms.h – klasa algorytmów zawierająca algorytm brutalny i liniowy
cxxopts.hpp – biblioteka, wykorzystywana do wczytywania argumentów