Piotr Dąbrowski Z34609

Patryk Pankiewicz Z34756

Sprawozdanie nr 2

Prowadzący: doc. dr inż. Dariusz Bursztynowski
Temat: Czasy zakończenia zadań w grafie czynności ze stałymi czasami realizacji zadań.

Opis:

W łukowym planie (grafie) czynności dla każdej czynności c jest podany czas jej realizacji określony stałą wartością t_c. Dla każdego zdarzenia w planie wyznaczyć:

1. najwcześniejszy możliwy czas osiągnięcia zdarzenia,
2. najpóźniejszy możliwy czas osiągnięcia zdarzenia.

1. Założenia Programu.

1.1. Dane wejściowe.

Dane wejściowe będą stanowiły dwa pliki tekstowe:

Vertices.txt - zawierający numery wierzchołków grafu które reprezentują zdarzenia (milestones).

Edges.txt - zawierający krawędzie (łuki) grafu skierowanego reprezentujące czynności. Sama krawędź będzie reprezentowana przez wartość wskazujące na numer wierzchołka początkowego i końcowego. Waga każdej krawędzi jest równa długości czasu realizacji czynności(t_c), jeżeli w grafie występują czynności pozorne to ich waga jest równa 0.

Komentarz nr 1.:

Jak widać założenia odnośnie danych wejściowych uległy zmianie w stosunku do sprawozdania nr 1, gdzie miał być to jeden plik zawierający trzy grupy liczb naturalnych dla każdej czynności: - Numer czynności - Numery poprzedników czynności - Czas trwania czynności. Jednak w tym modelu konieczne byłoby budowanie całego grafu wraz z uwzględnieniem czynności pozornych co nie jest zadaniem trywialnym i stanowi treść innego zadania ("Łukowy graf czynności dla zadanego grafu węzłowego"). Z tego powodu zrezygnowaliśmy z tego podejścia i założyliśmy że cały graf jest już dany (wraz z czynnościami pozornymi).

1.2. Dane wyjściowe.

Dane wyjściowe będzie stanowił plik tekstowy *Results.txt* zawierający trzy liczby naturalne dla każdej czynności:
- Numer czynności.
- Najwcześniejszy możliwy czas zakończenia czynności.
- Najpóźniejszy możliwy czas zakończenia czynności.

1.3. Założenia implementacyjne.

Program będzie podzielony na części realizujące następujące zadania:

- ➢ Wczytania danych z plików tekstowych oraz ich walidacja pod względem syntaktycznym.
- ➢ Sprawdzenia poprawności grafu i założeń(acykliczność grafu) oraz przygotowanie danych do użycia algorytmu.
- ➢ Algorytm obliczający najwcześniejsze i najpóźniejsze możliwe czasy osiągnięcia zdarzenia.
- ➢ Zapis wyników do pliku tekstowego.

1.4. Obsługa wyjątków.

Analizowane grafy z założenia muszą być acykliczne. Wystąpienie niedozwolonego cyklu w grafie jest sprawdzane na etapie sortowania topologicznego wierzchołków. W przypadku wystąpienia cyklu program wypisze na ekranie informację.

Analizowane grafy powinny być także spójne. Ten wyjątek nie będzie obsługiwany. Zakłada się, że dane wejściowe od użytkownika nie będą dopuszczały występowania grafów niespójnych.

2. Wybór i opis algorytmów.

W programie wykorzystane zostaną następujące algorytmy:

- **Sortowanie (numerowanie) topologiczne wierzchołków grafu wg algorytmu Kahna -** wykorzystane do sprawdzenia acykliczności grafu i przygotowania danych do dalszych obliczeń.
- **Metoda ścieżki krytycznej (CPM)** wykorzystana do wyznaczania najwcześniejszych możliwych czasów osiągnięcia zdarzeń oraz najpóźniejszych możliwych czasów osiągnięcia zdarzeń.

Oznaczenia stosowane w opisie algorytmów:

EST - najwcześniejszy możliwy termin rozpoczęcia danego zadania

EFT - najwcześniejszy możliwy termin zakończenia danego zadania

LFT - najpóźniejszy dopuszczalny termin zakończenia danego zadania

LST - najpóźniejszy dopuszczalny termin rozpoczęcia danego zadania

s_i - najwcześniejszy możliwy termin wystąpienia zdarzenia (wierzchołka o indeksie i)

l_i - najpóźniejszy możliwy termin wystąpienia zdarzenia (wierzchołka o indeksie i)

t_c - długość czasu realizacji danego zadania

p_{ij} - waga krawędzi między wierzchołkami i oraz j, $p_{ij}=t_c$

n - liczba wierzchołków grafu

V - zbiór wierzchołków grafu

E - zbiór krawędzi graf

2.1. Prezentacja algorytmów:

2.1.1. Sortowanie (numerowanie) topologiczne wierzchołków grafu wg algorytmu Kahna.[3]

Dzięki wykorzystaniu tego algorytmu można uzyskać poprawną kolejność wykonywania czynności (wierzchołków z pliku Vertices.txt) oraz sprawdzić, czy w grafie nie występują cykle. Każdy łukowy graf czynności musi być acykliczny, w przypadku wprowadzenia przez użytkownika grafu zawierającego cyklu program wyświetli stosowne ostrzeżenie.

Schemat działania algorytmu:

1. Dla każdego wierzchołka policz liczbę łuków dochodzących. Liczbę odwiedzonych wierzchołków zainicjalizuj jako 0.
2. Wybierz wszystkie wierzchołki, do których nie dochodzą żadne łuki i dodaj je do kolejki wierzchołków do przetworzenia.
3. Usuń odwiedzony wierzchołek z kolejki wierzchołków do przetworzenia i dodaj go do kolejki wynikowej.
   Zwiększ o 1 liczbę odwiedzonych wierzchołków.

   Zmniejsz o 1 liczbę łuków dochodzących dla wszystkich sąsiadów usuniętego wcześniej wierzchołka.

   Jeśli liczba łuków dochodzących sąsiedniego wierzchołka wynosi 0, dodaj go do kolejki wierzchołków do przetworzenia.

4. Powtarzaj punkt 3 do momentu opróżnienia kolejki wierzchołków do przetworzenia.
5. Jeśli liczba odwiedzonych wierzchołków nie jest równa liczbie wierzchołków, to sortowanie topologiczne dla danego grafu nie jest możliwe (w grafie istnieje cykl).

Złożoność obliczeniowa sortowania topologicznego: O(|V|+|E|).

2.1.2. Metoda ścieżki krytycznej. [1]

Wyznaczenie najwcześniejszych terminów zdarzeń.

Algorytm zaczyna się od pierwszego wierzchołka. Jako najwcześniejszy możliwy termin pierwszego zdarzenia na początku algorytmu przyjmuje się 0. Najwcześniejsze możliwe terminy pozostałych zadań to maksimum z sumy najwcześniejszego możliwego terminu poprzednika oraz czasu trwania czynności między poprzednikiem a danym zdarzeniem.

$s_i:=0$
for (j:=2, j<=n; j++) do
$s_j=max_{i:(i,,j)\in E}(s_i + p_{ij})$

Złożoność obliczeniowa: O(|E|).

Wyznaczenie najpóźniejszych terminów zdarzeń.

Algorytm zaczyna się od ostatniego wierzchołka. Jako najpóźniejszy możliwy termin wystąpienia ostatniego zdarzenia na początku algorytmu przyjmuje się najwcześniejszy możliwy termin wystąpienia ostatniego zdarzenia. Najpóźniejsze możliwe terminy wystąpienia pozostałych zdarzeń to minimum z różnicy najpóźniejszego możliwego terminu wystąpienia poprzednika oraz czasu trwania czynności.

$l_n:=s_n$
for (i:=n-1, i>=1; i--) do
$l_i=min_{j:(i,,j)\in E}(l_j - p_{ij})$

Złożoność obliczeniowa: O(|E|).

Wyznaczenie najwcześniejszych możliwych czasów osiągnięcia zdarzeń oraz najpóźniejszych możliwych czasów osiągnięcia zdarzeń.

$EFT(i,j)=s_i+p_{ij}$

$LFT(i,j)=l_j$

3. Struktury danych.

Graf reprezentowany będzie przez klasę **Graph** zawierającą listę wierzchołków nodes.

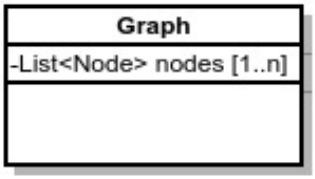

Diagram nr 1- klasa Graph.

Node(Wierzchołek) - klasa abstrakcyjna zawierająca pola:
number - numer identyfikujący wierzchołek wczytany z pliku.
connections - lista krawędzi incydentnych z danym wierzchołkiem.

Milestone(Zdarzenie) - klasa reprezentująca zdarzenie, rozszerzająca klasę Node.
Składa się z pól:
numberAfterSort - przechowują numer nadany przez algorytm sortowania topologicznego.
name - opcjonalne pole przechowujące nazwę.
EST, EFT, LST, LFT - pola przechowujące wyniki obliczone algorytmem CPM

Edge(Krawędź) - klasa abstrakcyjna zawierająca pola:
start- wskazanie na wierzchołek początkowy.
end - wskazanie na wierzchołek końcowy.
weight - waga

Action(Czynność) - klasa reprezentująca czynność, rozszerzająca klasę Edge.
składa się z pola:
name - pole przechowujące nazwę czynności.

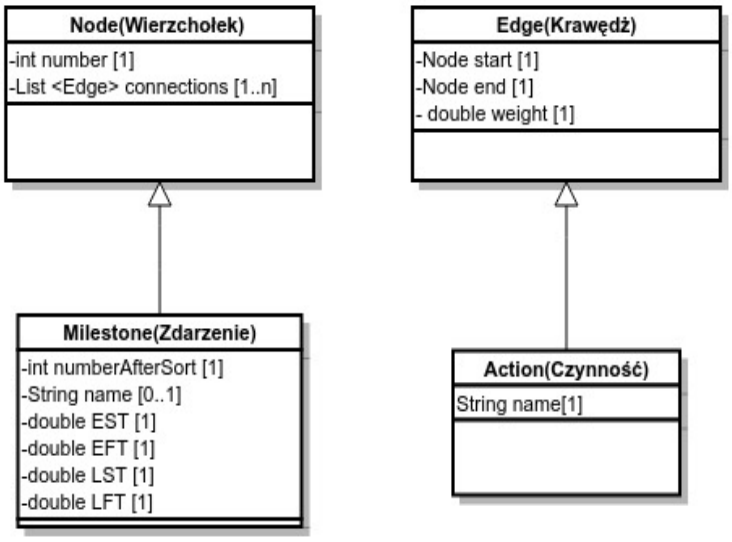

Diagram nr 2 klasy: Node, Edge, Milestone i Action.

4. Projekt testów.

Poprawność działania algorytmu dla przykładowych grafów będzie realizowana poprzez sprawdzenie zgodności wyników z wynikami otrzymanymi w programie *Microsoft Project Professional 2013.*

Z oszacowanej złożoności czasowej użytych algorytmów wynika że na szybkość ich działania będzie zależeć od liczby wierzchołków oraz od ilości krawędzi w grafie. Z tego względu grafy generowane do testów będą dobierane metodą prób i błędów.

Parametrami opisujące generowane grafy będą: liczba wierzchołków oraz gęstość. Przewidujemy rozpatrywanie szczególnych przypadków grafów jak np. graf będący drzewem.

5. Bibliografia.

[1] Pieńkosz K., Materiały wykładowe do przedmiotu Podstawy Badań Operacyjnych, Warszawa 2016

[2] K. Pieńkosz, J. Wojciechowski, Grafy i sieci, PWN, Warszawa 2013

[3] http://www.geeksforgeeks.org/topological-sorting-indegree-based-solution/