Uwagi

3: czego?
3: odnośnik literaturowy
5: liczba

Politechnika Śląska
Wydział Automatyki, Elektroniki i Informatyki

Podstawy Programowania Komputerów

Spedycja

autor	Hanna Podeszwa
prowadzący	dr inż. Krzysztof Simiński
rok akademicki	2018/2019
kierunek	informatyka
rodzaj studiów	SSI
semestr	1
termin laboratorium	środa, 13:45 – 15:15
sekcja	18
termin oddania sprawozdania	2019-01-11

1 Treść zadania

Napisać program wyszukujący najkrótsze możliwe trasy spedycyjne, wykorzystując algorytm Dijkstry. Miasta sąsiednie wraz z odległościami między nimi są zawarte w pliku tekstowym. Każda para miast jest w osobnej linii. Najkrótsze trasy spedycyjne zostaną zapisane do pliku, począwszy od miasta startowego, poprzez miasta sąsiednie (każda trasa w osobnej linii).

Program uruchamiany jest z linii poleceń z wykorzystaniem następujących przełączników:

- `-i` plik wejściowy z sąsiednimi miastami i odległościami między nimi
- `-o` plik wyjściowy z trasami spedycyjnymi
- `-s` miasto startowe

2 Analiza zadania

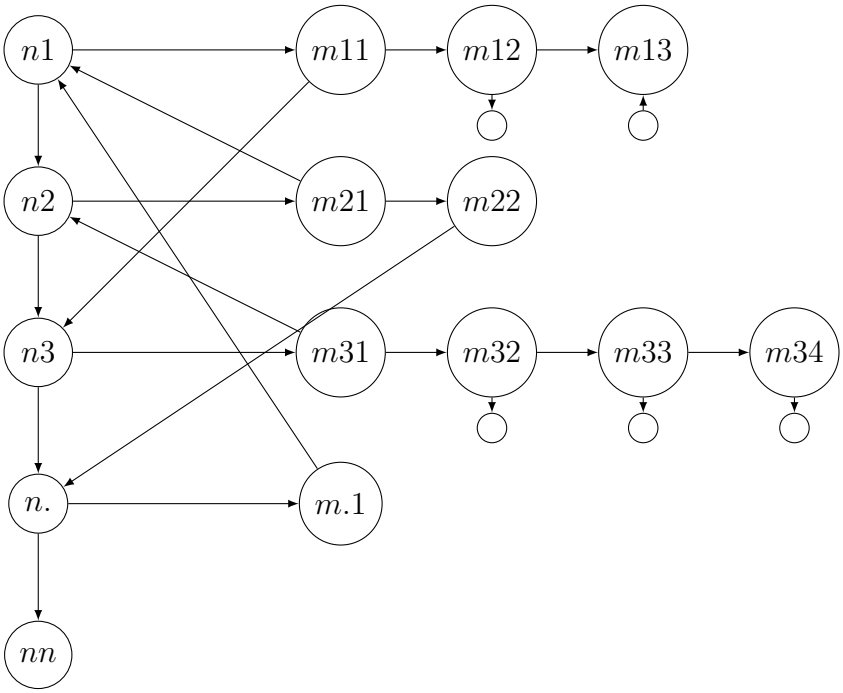
Zagadnienie przedstawia problem znalezienia najkrótszej możliwej trasy spedycyjnej pomiędzy miastem startowym, a wszystkimi innymi miastami zapisanymi w pliku.

2.1 Struktury danych

W programie wykorzystano listy jednokierunkowe do przechowywania miasta i dróg. Każdy element listy miast zawiera trzy wskaźniki: na nastepny element listy miast, na listę dróg oraz na miasto poprzednie, nazwę miasta, odległość od centrali oraz informację, czy dane miasto zostało już odwiedzone. Elementy listy dróg zawierają wskaźnik na odpowiednie miasto z listy miast oraz odległość między tym miastem, a miastem poprzednim. Rys. 1 przedstawia przykład [czego?]. Taka struktura danych umożliwia łatwe zastosowanie algorytmu Dijkstry.

2.2 Algorytmy

Program wyszukuje najkrótsze możliwe trasy spedycyjne, wykorzystując algorytm Dijkstry [odnośnik literaturowy] [1]. Zaczynając zawsze od miasta położonego najbliżej miasta startowego, porównuje wartość sumy wartości zmiennej odleglosc_od_centrali poprzedniego miasta i odległości między tymi miastami z aktualną wartością zmiennej odleglosc_od_centrali miasta następnego. Kolejnym krokiem jest relaksacja odległości od centrali wraz z ewentualnym zapamiętaniem adresu miasta poprzedniego. Utworzenie list i ich przejście jest wykonywane w średnim czasie $O(n^2 + m)$ (gdzie n oznacza liczba elementów listy miast, a m liczba elementów list dróg).



Rysunek 1: Przykład listy miast i listy dróg.

3 Specyfikacja zewnętrzna

Program jest uruchamiany z linii poleceń.

Należy przekazać do programu nazwy plików: wejściowego i wyjściowego oraz nazwę miasta startowego po odpowiednich przełącznikach (odpowiednio: `-i` dla pliku wejściowego, `-o` dla pliku wyjściowego i `-s` dla miasta startowego), np.

```
program -i miasta.txt -o trasy -s startowe
program -o trasy -s startowe -i miasta.txt
```

Pliki są plikami tekstowymi, ale mogą mieć dowolne rozszerzenie (lub go nie mieć.) Przełączniki mogą być podane w dowolnej kolejności. Uruchomienie programu bez żadnego parametru lub z parametrem `-h`

```
program
program -h
```

powoduje wyświetlenie krótkiej pomocy. Uruchomienie programu z nieprawidłowymi parametrami powoduje wyświetlenie komunikatu

```
Bledne argumenty
```

i wyświetlenie pomocy.

Podanie nieprawidłowej nazwy pliku powoduje wyświetlenie odpowiedniego komunikatu:

```
Nie udalo sie otworzyc pliku.
```

4 Specyfikacja wewnętrzna

Program został zrealizowany zgodnie z paradygmatem strukturalnym. W programie rozdzielono interfejs (komunikację z użytkownikiem) od logiki aplikacji (sortowania liczb).

4.1 Ogólna struktura programu

W funkcji głównej wywołana jest funkcja sprawdzArgumenty. Funkcja ta sprawdza, czy program został wywołany w prawidłowy sposób. Gdy program nie został wywołany prawidłowo, zostaje wypisany stosowny komunikat i program się kończy. Następnie wywoływana jest funkcja wczytajzPliku. Funkcja ta otwiera plik wejściowy, sczytuje miasta oraz odległości między nimi i umieszcza je w odpowiednich listach jednokierunkowych. Po sczytaniu wszystkich miast funkcja zamyka plik. W razie wystąpienia błędu funkcja zwraca **false**, w przeciwnym wypadku **true**. Następnie wywoływana jest funkcja Dijkstra. Funkcja przechodzi przez listy i zapisuje odpowiednie wartosci w zmiennych odleglosc_od_centrali i pMiastoPoprzednie. Następnie wywoływana jest funkcja wypisz_wynik.Funkcja zapisuje trasy spedycyjne i odległości od centrali do pliku wyjściowego. Po zapisaniu tras funkcja zamyka plik. Ostatnią funkcją programu jest funkcja usun zwalniająca pamięć.

4.2 Szczegółowy opis typów i funkcji

Szczegółowy opis typów i funkcji zawarty jest w załączniku.

5 Testowanie

Program został przetestowany na różnego rodzaju plikach. Pliki niepoprawne (zawierające zbyt małą ilość [liczba] miast lub niezawierające odległości) powodują zgłoszenie błędu. Plik pusty nie powoduje zgłoszenia błędu, ale utworzenie pustego pliku wynikowego (nie zostały podane żadne miasta ani odległości).

Program został sprawdzony pod kątem wycieków pamięci.

6 Wnioski

Program, wyszukujący najkrótsze możliwe trasy spedycyjne, jest programem prostym, chociaż wymaga samodzielnego zarządzania pamięcią. Najbardziej wymagające okazało się zastosowanie algorytmy Dijkstry. Szczególnie trudne było zapewnienie wpisywania prawidłowych wartości do zmiennej odleglosc_od_centrali , tak by wypisane trasy spedycyjne były możliwie najkrótsze. Przygotowanie tego projektu pozwoliło mi lepiej zrozumieć sposób działania list jednokierunkowych oraz wskaźników.

Literatura

[1] Adam Drozdek. *C++. Algorytmy i struktury danych.* Helion, Gliwice, 2001.

Dodatek
Szczegółowy opis typów i funkcji