SEMESTRÁLNÍ PRÁCE

z předmětu X36TIN – Teoretická informatika

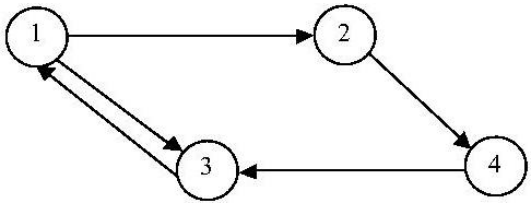
Jméno: Martin Plicka
St. skupina: 2/65
Cvičení: Úterý - 9.15
Zadání: 32 – Page Hopping

1.Zadání (anglicky)

Page Hopping

It was recently reported that, on the average, only 19 clicks are necessary to move from any page on the World Wide Web to any other page. That is, if the pages on the web are viewed as nodes in a graph, then the average path length between arbitrary pairs of nodes in the graph is 19.

Given a graph in which all nodes can be reached from any starting point, your job is to find the average shortest path length between arbitrary pairs of nodes. For example, consider the following graph. Note that links are shown as directed edges, since a link from page a to page b does not imply a link from page b to page a.



The length of the shortest path from node 1 to nodes 2, 3, and 4 is 1,1, and 2 respectively. From node 2 to nodes 1, 3 and 4, the shortest paths have lengths of 3, 2, and 1. From node 3 to nodes 1, 2, and 4, the shortest paths have lengths of 1, 2, and 3. Finally, from node 4 to nodes 1, 2, and 3 the shortest paths have lengths of 2, 3, and 1. The sum of these path lengths is 1 + 1 + 2 + 3 + 2 + 1 + 1 + 2 + 3 + 2 + 3 + 1 = 22. Since there are 12 possible pairs of nodes to consider, we obtain an average path length of 22/12, or 1.833 (accurate to three fractional digits).

Input

The input data will contain multiple test cases. Each test case will consist of an arbitrary number of pairs of integers, a and b, each representing a link from a page numbered a to a page numbered b. Page numbers will always be in the range 1 to 100. The input for each test case will be terminated with a pair of zeroes, which are not to be treated as page numbers. An additional pair of zeroes will follow the last test case, effectively representing a test case with no links, which is not to be processed. The graph will not include self-referential links (that is, there will be no direct link from a node to itself), and at least one path will exist from each node in the graph to every other node in the graph.

Output

For each test case, determine the average shortest path length between every pair of nodes, accurate to three fractional digits. Display this length and the test case identifier (they're numbered sequentially starting with 1) in a form similar to that shown in the sample output below.

Sample Input	Output for the Sample Input
1 2 2 4 1 3 3 1 4 3 0 0 1 2 1 4 4 2 2 7 7 1 0 0 0 0	Case 1: average length between pages = 1.833 clicks Case 2: average length between pages = 1.750 clicks

2.Zadání (stručný překlad)

U zadaného grafu určete průměrnou nejkratší cestu přes všechny uzly.

Vstup obsahuje více testovaných případů. Každý případ obsahuje libovolný počet párů čísel **a b** reprezentujících odkaz ze stránky **a** na **b**. Čísla stránek jsou z rozsahu 1..100. Každý testovaný případ je zakončen párem 0 0, který není interpretován. Další pár nul je za posledním testovaným případem a přestavuje prázdný test, který nebude interpretován. Graf nebude mít smyčky a existuje alespoň jedna cesta mezi každým párem uzlů.

Výstupem bude průměrná délka nejkratších cest mezi všemi uzly zobrazená na 3 desetinné číslice a číslo testovaného případu, jak uvádí tabulka v zadání.

3.Ovládání programu

Vstup programu se určí podle parametrů z příkazové řádky. Při volání bez parametru se čte ze standardního vstupu (tj. může být zadán z klávesnice nebo přesměrován z výstupu jiného procesu). Pokud je zadán jeden parametr, vstup se čte ze souboru zadaného tímto parametrem. V případě neexistujícího souboru či zadání více parametrů je vypsáno chybové hlášení.

4.Algoritmus

Pro řešení problému byl použit Floyd-Warshallův algoritmus pro zjištění minimálních cest pro všechny páry uzlů v orientovaném grafu. Ten funguje na principu postupného přidávání počtu vnitřních uzlů do požadované nejkratší cesty. Její délka bude maximálně (n – 1). Navíc byla použita varianta s redukovanými nároky na počet matic uložených v paměti. Při k-tém průchodu maticí nejsou totiž hodnoty prvků d_{ik} a d_{kj} z předchozího průchodu ovlivněny aktuálním průchodem, proto je možno použít pouze jednu matici n x n.

Popis v pseudokódu

```
FLOYD-WARSHALL-1(W)
    D :=W
        for k := 1 to n do
            for i := 1 to n do
                for j := 1 to n do
dij := min(dij ; dik + dkj)
return D
```

Vstupem algoritmu je **W**, matice w-délek grafu. Každý prvek w_{ij} udává w-ohodnocení hrany (i,j). V našem případě nabývá hodnot 0 (diagonální prvky), 1 (odkaz ze stránky **i** na **j**) nebo 999 (označuje nekonečno, tj. odkaz z **i** na **j** neexistuje).

Výstupem je pak **D**, matice w-vzdáleností, kde hodnota každého prvku d_{ij} udává délku nejkratší cesty z bodu **i** do bodu **j**. Pokud po provedení algoritmu je stále některý prvek d_{ij} roven 999 (tedy nekonečno), znamená to, že zadaný graf není silně souvislý a nevyhovuje zadání úlohy.

Jelikož zadání umožňuje některá čísla uzlů přeskočit (viz druhý příklad v zadání), je nutno při čtení čísel každému uzlu pomocí pomocného pole přiřadit nový index v matici podle pořadí výskytu ve vstupním souboru.

Po provedení F-W algoritmu je nutno všechny prvky matice D mimo hlavní diagonálu sečíst a spočítat průměr – požadovaný výsledek algoritmu.

Algoritmus se opakuje pro každý testovaný případ, dokud se ve vstupu neobjeví samostatný pár 0 0.

5.Popis implemetace

Proměnné

Nejdůležitější proměnné programu jsou následující:

```
#define PAGEMAX 100
```

int max;

- Uchovává velikost matice D

int pocet_pripadu;

- Obsahuje aktuální číslo testovaného případu

int matice[PAGEMAX*PAGEMAX];

- Pro uložení matice D. 2-rozměrná matice je do lineárního pole mapována. Právě používaná část a tedy velikost matice je určena proměnnou *max*.

int nahrada[PAGEMAX];

- Obsahuje tabulku nahrazení pro další výskyty stejných čísel uzlů na vstupu. Každé číslo nového uzlu na vstupu se nahradí prvním dostupným (dle aktuální hodnoty *max*).

Funkce

void error (char * text, int row, int col)

- Vypíše zadané chybové hlášení, případně doplněné o pozici chyby (řádek, prvek) na vstupu, ukončí program.

int check_string (char * string)

- Zkontroluje správný formát řetězce před převodem na číslo, vrátí 1 v případě chyby.

int mapuj (int z, int na)

- Mapuje indexy dvojrozměrného pole (*z*, *na*) na index lineárního pole.

void print ()

- Funkce použitá při ladění pro výpis matice D.

void init ()

- Maže obsah matice D (proměnná *matice[]*) a tabulky náhrad (*nahrada[]*). Připravuje program pro testování dalšího případu.

int min(int x,int y)

- Určí minimum ze zadaných operandů. Čísla $x \geq 999$ určují hodnotu „nekonečno“.

void algoritmus()

- Provede vlastní výpočet. Nejprve provede Floyd-Warshallův algoritmus, následně spočítá aritmetický průměr z prvků mimo hlavní diagonálu matice D a vypíše jej.

int zarad (int cislo)

- Přiřadí první volný index matice D danému uzlu a vrátí jej. Pokud bude funkce volána znovu se stejným parametrem, bude vráceno stejné číslo. Funkce tak umožňuje „přeskočit“ nepoužitá čísla uzlů zadaných na vstupu.

int main(int argc, char *argv[])

- Provádí čtení, ošetření správnosti a zpracovávání vstupních hodnot každého testovaného případu. Volá funkci pro provedení algoritmu.

6.Složitost

Časová

Složitost čtení je závislá na počtu uzlů grafu n. Pro počet načtených dvojic m platí $m \leq n(n-1)$. Pak složitost čtení vstupu pro jeden případ je $O(n^2)$.

Složitost Floyd-Warshallova algoritmu je $O(n^3)$ (tři vnořené cykly o délce n).

Složitost výpočtu průměrně hodnoty je opět $O(n^2)$, jelikož je nutno sečíst n(n-1) prvků a následně je vydělit.

Výsledná asymptotická časová složitost algoritmu je tedy $O(n^3)$.

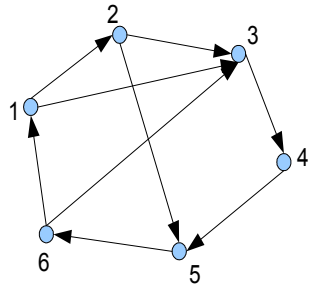
Paměťová

Pro uložení matice W a tedy D je potřeba n^2 paměťových míst. Tento požadavek je dominantní oproti ostatním proměnným. A tedy paměťová složitost algoritmu je $\Theta(n^2)$.

V případě mého řešení bylo za podmínky zadání max. 100 uzlů a z důvodu rychlosti a jednoduchosti implementace použito staticky definované pole o rozměru 100x100. Není tak třeba dynamicky alokovat paměť při nalezení dalšího čísla uzlu na vstupu. Za předpokladu, že proměnná typu Int má velikost 4B, toto pole zabere v paměti cca 40kB (100x100x4). Pokud by zadání dovolovalo vyšší počet uzlů, bylo by nutno paměť alokovat dynamicky.

7.Testy

Aplikace byla kromě příkladů v zadání testována i na těchto grafech:

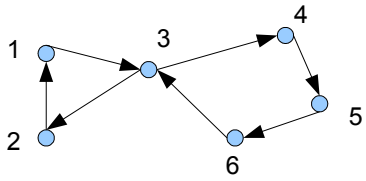


VSTUP:

1 2 2 4 1 3 3 1 4 3 0 0
1 2 1 4 4 2 2 7 7 1 0 0
1 2 2 3 3 4 4 5 5 6 6 1 1 3 6 3 2 5 0 0
2 1 1 3 3 2 3 4 4 5 5 6 6 3 0 0
0 0

VÝSTUP:

Pripad c. 1: Prumerna vzdalenost mezi strankami = 1.833 kliknuti
Pripad c. 2: Prumerna vzdalenost mezi strankami = 1.750 kliknuti
Pripad c. 3: Prumerna vzdalenost mezi strankami = 2.200 kliknuti
Pripad c. 4: Prumerna vzdalenost mezi strankami = 2.500 kliknuti



8.Závěr

Úloha je příkladem na jeden z častých problémů teorie grafů, tedy hledání minimálních cest mezi všemi páry uzlů.

Pro řešení dané úlohy lze použít i jiné grafové algoritmy, např. Dijkstrův algoritmus použitý na všechny uzly grafu. Ale při využití jiných algoritmů dosáhneme srovnatelné nebo i horší časové složitosti. Floyd - Warshallův algoritmus je navíc jednoduchý a snadno implementovatelný.

Zdroje

Josef Kolář: Teoretická informatika - skriptum