Politechnika Wrocławska

Projektowanie efektownych algorytmów

Dawid Słodkowski

1. Wstęp teoretyczny

Problem komiwojażera (ang. *travelling salesman problem*, w skrócie *TSP*) – problem obliczeniowy polegający na poszukiwaniu w grafie takiego cyklu, który zawiera wszystkie wierzchołki (każdy dokładnie raz) i ma jak najmniejszy koszt. Bardziej formalnie, problem komiwojażera polega na poszukiwaniu w grafie cyklu Hammiltona o najmniejszej wadze.

Problem ma liczne zastosowania w życiu codziennym. Najlepszym przykładem jest praca kuriera, który musi wyjechać z magazynu, zawieźć przesyłki w różne miejsca i wrócić do magazynu.

2. Pseudokod

K01: S_H.push (v) — *Odwiedzony wierzchołek dopisujemy do ścieżki*

K02: **Jeśli** S_H nie zawiera n wierzchołków, **to idź do** kroku K10 — *Jeśli brak ścieżki Hamiltona, przechodzimy do wyszukiwania*

K03: **Jeśli** nie istnieje krawędź z v do v_0, **to idź do** kroku K17 — *Jeśli ścieżka Hamiltona nie jest cyklem, odrzucamy ją*

K04: $d_H \leftarrow d_H$ + waga krawędzi z v do v_0 — *Uwzględniamy w sumie wagę ostatniej krawędzi cyklu*

K05: **Jeśli** $d_H \geq d$, **to idź do** kroku K08 — *Jeśli znaleziony cykl jest gorszy od bieżącego, odrzucamy go*

K06: $d \leftarrow d_H$ — *Zapamiętujemy sumę wag cyklu*

K07: Skopiuj stos S_H do stosu S — *oraz sam cykl Hamiltona*

K08: $d_H \leftarrow d_H$ - waga krawędzi z v do v_0 — *Usuwamy wagę ostatniej krawędzi z sumy*

K09: **Idź do** kroku K17

K10: *visited* [v] ← true — *Wierzchołek zaznaczamy jako odwiedzony, aby nie był ponownie wybierany przez DFS*

K11: **Dla** każdego sąsiada u wierzchołka v: **wykonuj** kroki K12...K15 — *Przechodzimy przez listę sąsiedztwa*

K12: **Jeśli** *visited* [u] = true, **to następny obieg pętli** K11 — *Omijamy wierzchołki odwiedzone*

K13: $d_H \leftarrow d_H$ + waga krawędzi z v do u — *Obliczamy nową sumę wag krawędzi ścieżki*

K14: TSP (n, *graf*, u, v_0, d, d_H, S, S_H, *visited*) — *Wywołujemy rekurencyjnie poszukiwanie cyklu*

K15: $d_H \leftarrow d_H$ - waga krawędzi z v do u — *Usuwamy wagę krawędzi z sumy*

K16: *visited* [v] ← false — *Zwalniamy bieżący wierzchołek*

K17: S_H.pop( ) — *Usuwamy bieżący wierzchołek ze ścieżki*

K18: **Zakończ**

3. Pomiar czasu

   Do pomiaru czasu wykorzystałem następującą formę pomiaru:

```
struct timeval start, end;
long mtime, secs, usecs;

gettimeofday(&start, NULL);
kod();
gettimeofday(&end, NULL);
secs  = end.tv_sec  - start.tv_sec;
usecs = end.tv_usec - start.tv_usec;
mtime = (((secs) * 100 + usecs/100.0) + 0.5);
printf("Czas: %ld ms\n", mtime);
```

Pomiar	Czas (ms)
1	42
2	9
3	9
4	9
5	10
6	10
7	17
8	10
9	15
10	9

4. Wnioski

Algorytm podziału i ograniczeń jest nieefektywny przy większej ilości wierzchołków. Istnieje kilka ścieżek optymalnych rozwiązania problemu w związku z tym mogą pojawiać się różne ścieżki końcowe.

5. Bibliografia

http://algorytmy.ency.pl/artykul/problem_komiwojazera

https://eduinf.waw.pl/inf/alg/001_search/0140.php