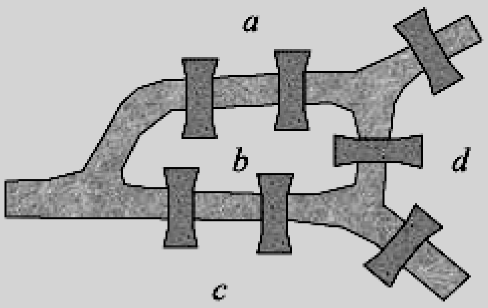
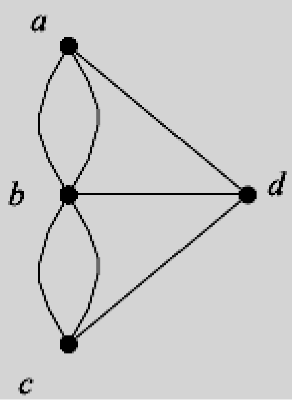
Grafy eulerowskie

Nazwa „eulerowski” pochodzi stąd, iż Euler w 1736 r. rozwiązał „problem mostów królewieckich”. Pytano, czy można przejść dokładnie raz przez każdy z siedmiu mostów (rys. po lewej) tak, aby powrócić do punktu wyjścia. Zauważmy, że problem jest równoważny stwierdzeniu, czy graf pokazany na rys. po prawej stronie jest eulerowski.





Cykl Eulera – cykl zawierający każdą krawędź dokładnie raz.

Tw. Eulera (1736)

Graf G ma cykl Eulera, iff G jest spójny, $|V| > 1$ i stopień każdego wierzchołka jest liczbą parzystą.

Idea algorytmu Hong Tuy (wyznaczania cyklu Eulera).

Uogólnienie – Problem chińskiego listonosza (1962).

Problem chińskiego listonosza

Dana jest sieć ulic oraz poczta. Aby listonosz dostarczył korespondencję musi przejść wzdłuż każdej ulicy co najmniej raz i powrócić do punktu wyjścia. Formułując problem w języku grafów, pytamy o najkrótszą zamkniętą marszrutę w grafie G utworzonym na podstawie sieci ulic, w którym wagi krawędzi odpowiadają długościom ulic.

Znany jest efektywny algorytm rozwiązujący ten problem, lecz my rozważymy dwa przypadki:

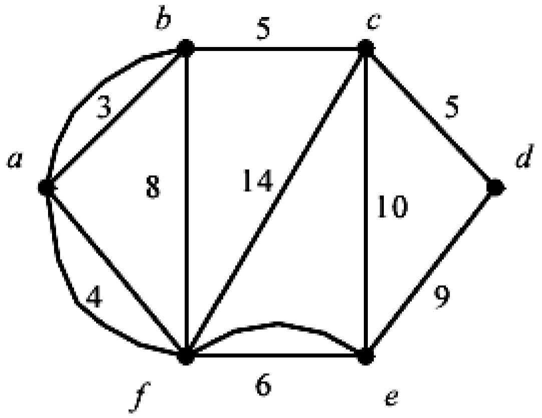
Przypadek 1: graf G jest eulerowski. Wówczas każdy cykl Eulera jest optymalnym rozwiązaniem, które można znaleźć korzystając np. z algorytmu Fleury'ego.

Problem chińskiego listonosza

Przypadek 2: graf *G* jest półeulerowski. Znajdujemy ścieżkę Eulera łączącą dwa wierzchołki nieparzystego stopnia *u* i *v*. Następnie szukamy najkrótszej drogi z *u* do *v*. Łącząc obie drogi otrzymujemy rozwiązanie.

Przykład

1) Droga Eulera:
   $b \to a \to f \to b \to c \to$
   $d \to e \to c \to f \to e$
2) Najkrótsza droga z *e* do *b*
   $e \to f \to a \to b$
3) „Trasa listonosza":
   $b \to a \to f \to b \to c \to$
   $d \to e \to c \to f \to e \to f \to a \to b$

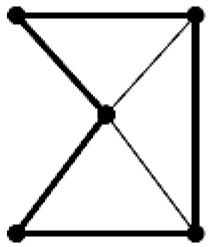


Dla grafów krawędziowych albo sieci problem jest wielomianowy (graf mieszany – NP-trudny).

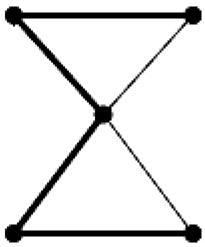
Grafy hamiltonowskie

Def. *Cykl* (*droga*) *Hamiltona* jest to cykl (droga), w którym każdy wierzchołek grafu występuje dokładnie raz. Graf jest *hamiltonowski* (*półhamiltonowski*), o ile posiada cykl (drogę) Hamiltona.

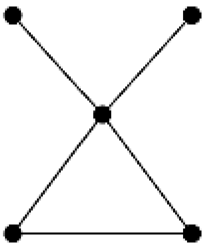
Przykład



Graf hamiltonowski



Graf półhamiltonowski



Graf nie jest ani hamiltonowski ani półhamiltonowski

Uwaga Problem polegający na stwierdzeniu czy dany graf *G* jest hamiltonowski jest NP-zupełny. Oznacza to, że nie są znane efektywne (działające w czasie wielomianowym) algorytmy rozwiązujące ten problem. Nie jest również znane twierdzenie podające warunki konieczne i dostateczne na to, aby *G* był hamiltonowski.

Geneza

W roku 1859 Hamilton opracował grę logiczną, która składała się z drewnianego regularnego dwunastościanu, w którym punkty narożne (wierzchołki) opisano nazwami sławnych miast. Celem gry było znalezienie trasy przebiegającej wzdłuż krawędzi figury w taki sposób, aby przebiegała przez każde z miast dokładnie jeden raz i kończyła się w miejscu startu - czyli tworzyła cykl.

<u>Tw</u>. Diraca (1952)

Jeżeli graf *G* ma $n \geq 3$ wierzchołków oraz $(\forall v \in V)\left(\deg(v) \geq \frac{n}{2}\right)$, to *G* ma cykl Hamiltona.

<u>Tw</u>. Ore (1960)

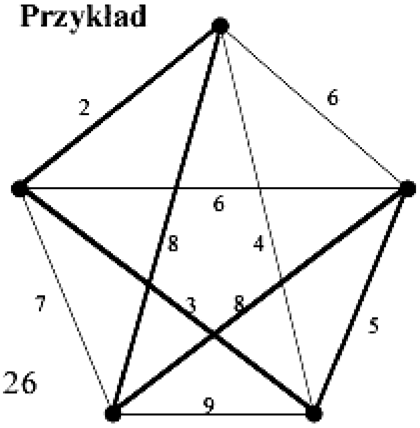
Jeżeli graf *G* ma $n \geq 3$ wierzchołków oraz $(\forall u,v \in V)(\deg(u) + \deg(v) \geq n)$, to *G* ma cykl Hamiltona.

Problem komiwojażera

Dany jest zbiór miast. Komiwojażer chce odwiedzić wszystkie miasta (każde dokładnie raz) i powrócić do punktu wyjścia. Problem polega na znalezieniu najkrótszej trasy o tej własności.

Zdefiniujemy powyższy problem w języku teorii grafów. Niech będzie dany graf pełny G. Zakładamy, że z każdą krawędzią e_i jest skojarzona jej waga (długość) oznaczana dalej przez w_i. Rozwiązaniem problemu komiwojażera jest taki cykl Hamiltona, którego suma wag krawędzi jest minimalna.

Przykład



$\Sigma = 26$

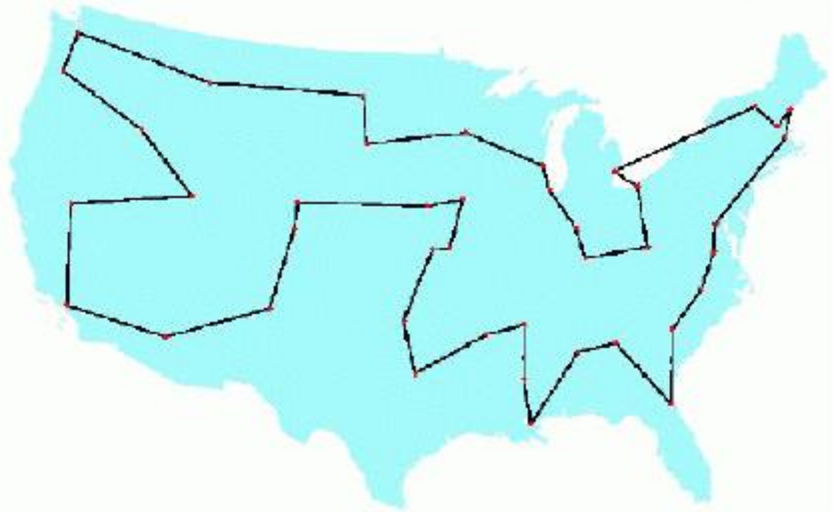
Uwagi

- problem komiwojażera jest NP-trudny, co oznacza, że nie są znane algorytmy o wielomianowej złożoności obliczeniowej rozwiązujące ten problem (przypuszczalnie takie nie istnieją)
- w praktyce jesteśmy zmuszeni posługiwać się wielomianowymi algorytmami przybliżonymi, tzn. takimi, które szybko znajdują rozwiązanie, które jest w przybliżeniu równe optymalnemu

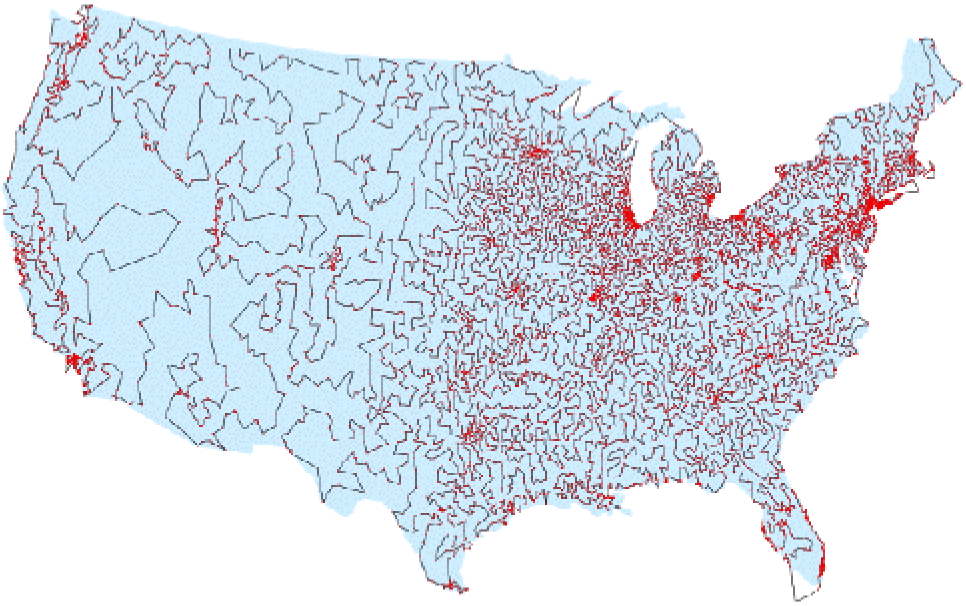
Przykład Jednym z możliwych algorytmów dokładnych jest sprawdzenie wszystkich możliwych cykli Hamiltona i wybranie najkrótszego. Wadą takiego podejścia jest to, że liczba cykli jest zbyt duża, gdyż dla n-wierzchołkowego grafu wynosi $(n!)/2$. Stąd, jeśli dysponujemy komputerem sprawdzającym milion permutacji na sekundę, to:

- $n = 10 \Rightarrow$ ilość cykli = $(10!)/2 = 1814400 \Rightarrow$ czas obliczeń = 1.8 s
- $n = 20 \Rightarrow$ ilość cykli = $(20!)/2 \approx 10^{18} \Rightarrow$ czas obliczeń $\approx$ 40 tys. lat

Historia problemu komiwojażera



George Dantzig, Ray Fulkerson i Selmer Johnson (1954) zaprezentowali optymalne rozwiązanie problemu komiwojażera dla 49 amerykańskich miast.



Rozwiązanie obejmujące 13549 miast amerykańskich, uzyskane w 1998 roku.

Problem

Dane:

Sieć $\boldsymbol{G_c}=(\boldsymbol{V},\boldsymbol{E};\boldsymbol{c})$, gdzie $\boldsymbol{G}=(\boldsymbol{V},\boldsymbol{E})$ – graf pełny krawędziowy,

$\boldsymbol{c}:\boldsymbol{E}\to\boldsymbol{R}^{+}$ - macierz odległości.

Wynik:

Cykl (droga) komiwojażera.

Uwaga:

G jest grafem pełnym (jeżeli nie, to dodajemy brakujące krawędzie i obciążamy wagą $M=n\max\{c_{ij}\}+1$.

Założenie:

Macierz *C* spełnia warunek trójkąta.

Twierdzenie.

Jeżeli macierz odległości spełnia warunek trójkąta, to optymalne rozwiązanie problemu komiwojażera jest cyklem Hamiltona (zawiera każdy wierzchołek dokładnie raz).

Uwaga:

Jeżeli *C* nie spełnia warunku trójkąta, wówczas:

a) c_{ij} zastępujemy przez

$c_{ij}^{*}=$ **długość najkrótszej drogi z *i* do *j*.**

b) rozwiązujemy problem komiwojażera z macierzą *C.**

c) zastępujemy krawędzie wyznaczonego cyklu przez najkrótsze drogi.

Problem komiwojażera jako zadanie programowania liniowego

Niech $x_{ij} = \begin{cases} 1, \textit{komiwojażer jedzie z i do j}, \\ 0, \textit{w pp.} \end{cases}$

Zminimalizować:

$$\sum_i \sum_j x_{ij} c_{ij},$$

przy ograniczeniach:

$$\sum_i x_{ij} = 1,\ j = 1, 2, ..., n,$$

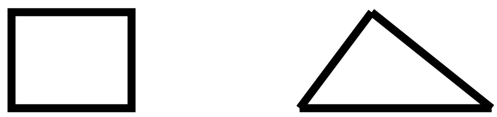
$$\sum_j x_{ij} = 1,\ i = 1, 2, ..., n.$$

Powyższy problem jest zadaniem przydziału pracy (PP - wielomianowym).
Ograniczenia należy uzupełnić:

$$\forall Q \subset V, \sum_{i \in Q} \sum_{j \in V \setminus Q} x_{ij} \geq 1.$$

Uwagi:

1. Ograniczenia dotyczą wszystkich podzbiorów ($2^{|V|}$), stąd NP-trudność.
2. Ograniczenia eliminują przypadek.



3. Zbiór rozwiązań PP zawiera rozwiązania komiwojażera, więc optymalne rozwiązanie PP jest dolnym ograniczeniem optymalnej drogi komiwojażera.

Algorytmy przybliżone (konstrukcyjne)

1. Startują z pojedynczej krawędzi (wierzchołka i w każdej iteracji powiększają rozwiązanie częściowe, aż do osiągnięcia cyklu Hamiltona.
2. Poprawiają rozwiązanie przez wymianę pewnego podzbioru krawędzi.
3. Wyznaczają podgraf częściowy (nie będący rozwiązaniem dopuszczalnym) i transformują go w cykl Hamiltona.

Algorytm włączania

Szkic algorytmu:

- wybierz dowolny wierzchołek v_0 grafu jako początkowy wierzchołek cyklu
- załóżmy, że został wyznaczony fragment cyklu zawierający wierzchołki $v_0,...,v_k$. W celu rozszerzenia takiego częściowego rozwiązania na $(k+1)$-elementowy fragment cyklu wykonywane są dwa kroki:
    - *krok wyboru*: wybierz wierzchołek v spośród wierzchołków nie należących do cyklu
    - *krok włączania*: określ miejsce w dotychczas utworzonym fragmencie cyklu, w które należy wstawić wierzchołek v

Kryteria wyboru

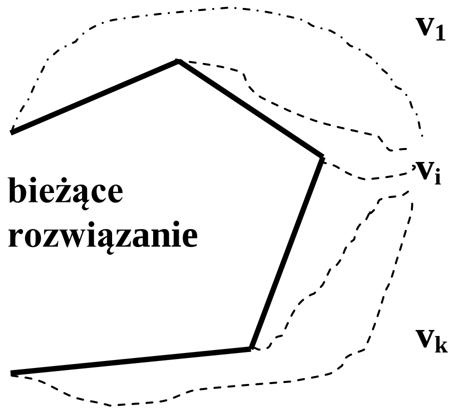
Przykładowe kryteria stosowane podczas kroku wyboru to:

- wybór losowego wierzchołka spośród wierzchołków nie włączonych jeszcze do cyklu
- wierzchołek leżący najbliżej dotychczas zdefiniowanego fragmentu cyklu
- dla każdego wierzchołka wyznaczyć koszt jego włączenia do cyklu w najbardziej korzystnym dla niego miejscu i wybrać wierzchołek o minimalnym koszcie
- wybór wierzchołka znajdującego się najdalej od cyklu

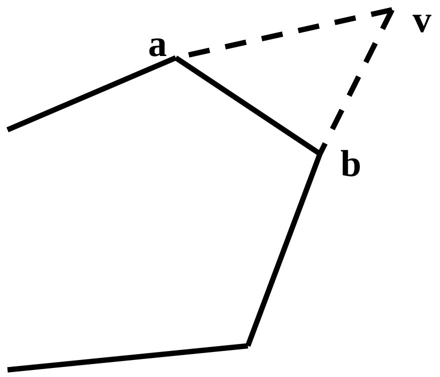
Uwagi:

- brak teoretycznych analiz porównujących powyższe podejścia
- testy komputerowe wskazują, że włączanie najdalszego wierzchołka okazuje się być techniką najskuteczniejszą w praktyce

Wybór „najbliższego” („najdalszego”) wierzchołka (sąsiada) od bieżącego rozwiązania



Włączanie



$$\min\{c_{a,v} + c_{b,v} - c_{a,b}\}$$

Algorytmy najbliższego (najdalszego sąsiada) mają oszacowanie równe 2, a złożoność $O(n^2)$. Lepsze wyniki (eksperymentalne) otrzymano dla „najdalszego” sąsiada.

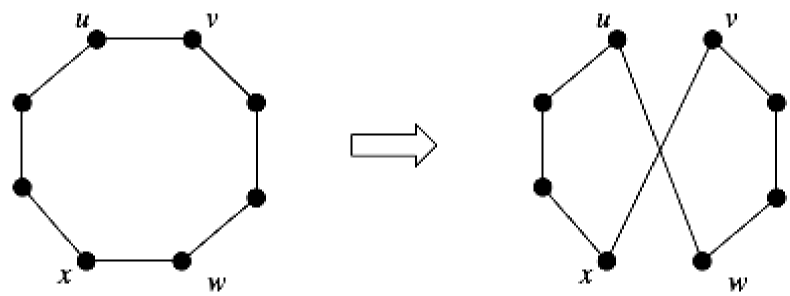
Poprawianie rozwiązania przez wymianę krawędzi

(okrojona wersja algorytmu optymalnego Lina-Kernighana)

- Załóżmy, że mamy pewien cykl komiwojażera C dla grafu G złożony z krawędzi $e_1,\dots,e_n$
- wybieramy dwie krawędzie e_i, e_j, które nie są sąsiednie
- oznaczmy $e_i=\{u,v\}$ oraz $e_j=\{w,x\}$
- cykl C przekształcamy w cykl C' w taki sposób, że

  $$C' = (C\setminus\{e_i,e_j\})\cup\{\{u,w\}, \{v,x\}\},$$

  gdzie wierzchołki u,w należą do różnych składowych podgrafu $C\setminus\{e_i,e_j\}$



przed **(………,u,v,a………b,w,x,…………..)**

po **(……….u,w,b……...a,v,x,…………..)**

Algorytm 2-opt

- przy oznaczeniach z poprzedniego slajdu możemy napisać, że koszt nowego cyklu C' wynosi

  $$w(C') = w(C) - w(\{u,v\}) - w(\{w,x\}) + w(\{u,w\}) + w(\{v,x\})$$
- zatem, jeśli $w(\{u,v\}) + w(\{w,x\}) > w(\{u,w\}) + w(\{v,x\})$, to nowe rozwiązanie jest lepsze od dotychczasowego
- w celu wyznaczenia cyklu C' algorytm szuka takiej pary krawędzi e_i, e_j, aby maksymalnie obniżyć koszt nowego bieżącego rozwiązania
- jeśli zmniejszenie wagi cyklu C nie jest możliwe, to C jest rozwiązaniem 2-*optymalnym* i algorytm kończy działanie
- powyższa procedura przekształcania bieżącego cyklu C jest powtarzana dopóki możliwa jest redukcja wagi cyklu dzięki opisanej wcześniej wymianie krawędzi

Liczba krawędzi do wymiany $n(n-3)/2$

Algorytm 3-opt

- algorytm 3-optymalny jest uogólnieniem algorytmu 2-optymalnego
- w danej iteracji usuwane są trzy krawędzie x, y, z z cyklu C
- następnie do cyklu dodawane są trzy krawędzie e_1, e_2, e_3 w taki sposób, aby graf $C' = (C \setminus \{x, y, z\}) \cup \{e_1, e_2, e_3\}$ tworzył cykl
- istnieje wiele możliwości wyboru krawędzi e_1, e_2, e_3
- algorytm zmiany bieżącego cyklu można opisać następująco:

$d := +\infty$;
dla każdej możliwej trójki krawędzi x, y, z należącej do C:
    dla wszystkich e_1, e_2, e_3 t.ż. $(C \setminus \{x, y, z\}) \cup \{e_1, e_2, e_3\}$ tworzy cykl:
        jeśli $w((C - \{x, y, z\}) \cup \{e_1, e_2, e_3\}) < d$ to
            $d := w((C \setminus \{x, y, z\}) \cup \{e_1, e_2, e_3\})$;
jeśli $w(C) > d$ to utwórz nowy bieżący cykl C; (*)

- algorytm kończy działanie, gdy warunek (*) jest fałszywy
- jeśli w-k (*) jest spełniony, to podane kroki są powtarzane dla nowego cyklu

Złożoność O(n^3). Algorytm 3-opt jest „lepszy” od 2-opt o 1-3%.

Algorytm *r*-opt

- w każdej iteracji usuwamy r łuków (ich zbiór oznaczmy przez A) z bieżącego cyklu C
- do zbioru dróg $C - A$ dodajemy r łuków (zbiór dodanych łuków oznaczmy przez B
- w każdej iteracji badamy wszystkie dopuszczalne zbiory A i B i wybieramy takie rozwiązanie, że $w((C\backslash A)\cup B)$ jest minimalne
- przez dopuszczalne zbiory rozumiemy takie, że $(C\backslash A)\cup B$ jest cyklem

Uwaga *Rozwiązanie r-opt jest również rozwiązaniem* $(r - 1)$*-optymalnym.*

Efektywność

- liczba możliwych wyborów krawędzi A wynosi $O(n!/(r!(n-r)!))$
- zbiór $C\backslash A$ można uzupełnić do cyklu na $O(2^{r-1}(r-1)!)$ sposobów
- złożoność całego algorytmu wynosi zatem

$$O(p(n)2^{r-1}(r-1)!\binom{n}{r})$$

Uwagi:

- w praktyce algorytm 3-opt okazuje się dużo lepszy niż 2-opt, jednak algorytm 4-opt nie jest znacząco lepszy niż 3-opt
- wynik końcowy zależy od wyboru cyklu początkowego – nie jest prawdą, że lepszy cykl początkowy prowadzi do znalezienia lepszego rozwiązania, jednak testy komputerowe wskazują, że warto wybierać jako punkt startowy dla algorytmu cykl o możliwie małej wadze

Algorytmy: 4,5,….-OPT nie są „znacząco" lepsze od 3-OPT.

<u>Uwaga</u>. Jeśli cykl Hamiltona nie jest optymalny, to różni się od optymalnego pewną liczbą krawędzi. Wystarczy znaleźć ten zbiór (2^n) i dokonać wymiany.

<u>Algorytm</u> Lina-Kernighana (optymalny):
Wykonać algorytmy 2,3,…,*n*-optymalny.

Algorytmy przekształcające podgraf częściowy w cykl Hamiltona.

Idea

Algorytm w pierwszej fazie wyznacza minimalne drzewo rozpinające grafu. Następnie, wierzchołki stopnia nieparzystego w drzewie są łączone krawędziami w pary tak, aby waga grafu indukowanego przez drzewo rozpinające i krawędzie łączące wierzchołki stopnia nieparzystego w tym drzewie była możliwie najmniejsza. Wszystkie wierzchołki tak wyznaczonego grafu mają stopień parzysty, więc istnieje w nim cykl Eulera. W ostatnim kroku algorytmu cykl ten jest wyznaczany i przekształcany w cykl Hamiltona, który jest jednocześnie cyklem wynikowym.

1.

Algorytm 1.

Krok 1*: wyznaczyć minimalne drzewo spinające *T* w grafie *G

***Krok 2*: Uzupełnić drzewo *T* do podgrafu Eulera *S*. Można to zrobić na dwa sposoby:**

Wariant 1**: Powielić krawędzie drzewa;**

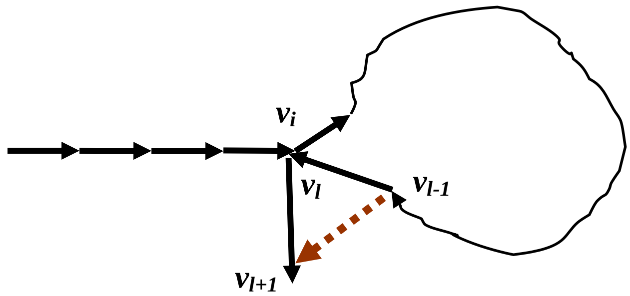
Wariant 2: **Wyznaczyć minimalne skojarzenie pomiędzy wierzchołkami nieparzystego stopnia drzewa *T*.**

***Krok 3*: Wyznaczyć cykl Eulera *C* w grafie *S*.**

***Krok* 4: Przekształcić cykl Eulera *C* w cykl Hamiltona *H*.**

Wyjaśnienie – Krok 4: Przekształcenie cyklu Eulera w cykl Hamiltona

Przeglądamy (zaczynając od pierwszego wierzchołka cykl Eulera. Niech v_i będzie pierwszym powtarzającym się w cyklu wierzchołkiem.



W miejsce (v_{l-1},v_l) i (v_i,v_{l+1}) wstawiamy (v_{l-1},v_{l+1}) – linia przerywana.

Funkcja odległości spełnia warunek trójkąta, więc po modyfikacji długość cyklu się nie zwiększy.

Złożoność

Algorytm 1.

Krok 1*: wyznaczyć minimalne drzewo spinające *T* w grafie *G
(alg. chciwości O(mlnm), Dijkstry $O(n^2)$);

***Krok 2*: Uzupełnić drzewo *T* do podgrafu Eulera *S*. Można to zrobić na dwa sposoby:**

Wariant 1: **Powielić krawędzie drzewa;** ***$O(n^2)$***

Wariant 2: **Wyznaczyć minimalne skojarzenie pomiędzy wierzchołkami nieparzystego stopnia drzewa *T*.**
(algorytm Christophidesa $O(n^3)$)

Krok 3*: Wyznaczyć cykl Eulera *C* w grafie *S*.** ***(Hong Tuy $O(n^2)$)

***Krok* 4: Przekształcić cykl Eulera *C* w cykl Hamiltona *H*.**
$O(n^2)$

Podsumowanie:

Wariant 1: ***$O(n^2)$;***
Wariant 2: ***$O(n^3)$;***

Oszacowanie

C* **- optymalna droga komiwojażera;**
***c*(*A*) – suma wag krawędzi należących do A.**

Krok 1*: wyznaczyć minimalne drzewo spinające *T* w grafie *G

$c(T) \le c(C^*)$ **{po odrzuceniu dowolnej krawędzi z *C** otrzymujemy jedno z drzew}**

***Krok 2*: Uzupełnić drzewo *T* do podgrafu Eulera *S*. Można to zrobić na dwa sposoby:**

Wariant 1: **Powielić krawędzie drzewa;**

$c(S_1) \le 2c(C^*)$

Wariant 2: **Wyznaczyć minimalne skojarzenie pomiędzy wierzchołkami nieparzystego stopnia drzewa *T*.**

$c(S_2) \le 3/2c(C^*)$

***Krok 3*: Wyznaczyć cykl Eulera *C* w grafie *S*.**
{długość się nie zwiększa}

***Krok* 4: Przekształcić cykl Eulera *C* w cykl Hamiltona *H*.**
{z warunku trójkąta, długość się nie zwiększa}

Wariant 1: $c(H_1) \le 2c(C^*)$
Wariant 1: $c(H_2) \le 3/2c(C^*)$

Lemat (Wariant 2**).**
(Waga minimalnego skojarzenia pomiędzy wierzchołkami nieparzystego stopnia w drzewie *T*)$\le 1/2c(C^*)$.

D-d.
***N* – graf pełny na wierzchołkach nieparzystego stopnia w T (ma parzystą liczbę wierzchołków.**
1. długość drogi kom. *K* (w *N*), $c(K) \le c(C^*)$ (łatwo pokazać z war. trójkąta).
2. *K* ma parzystą liczbę wierzchołków i krawędzi. Biorąc, co 2-gą krawędź z *K*, otrzymamy dwa pełne skojarzenia wierzchołków z *N*. Przynajmniej jedno z nich ma wagę $\le c(K)/2$, a więc i od $c(C^*)/2$. Wobec tego, minimalne skojarzenie ma wagę $\le c(C^*)/2$.

Warto zaznaczyć, że do dziś (2011r) nie jest znany algorytm o mniejszym współczynniku aproksymacji i stworzenie takiego algorytmu jest jednym z najważniejszych problemów otwartych w dziedzinie algorytmów aproksymacyjnych. W 2010r. udało się uzyskać lepsze algorytmy (aktualny rekord to 13/9=1,(4)) dla szczególnego przypadku, gdy W jest metryką najkrótszych ścieżek w nieskierowanym grafie bez wag.

DFS – procedura przeglądania w głąb grafu (odwiedzane wierzchołki umieszczane są na stosie), drzewo DFS.

1. Algorytm 2 – przybliżony.

Algorytm A(*G*);

<u>begin</u>

wyznaczyć minimalne drzewo spinające grafu (procedura DFS);
jeżeli $v_1,v_2,\ldots,v_n$, są kolejno odwiedzanymi wierzchołkami (w DFS), to utwórz cykl Hamiltona: $C=(v_1,v_2,\ldots,v_n, v_1)$;

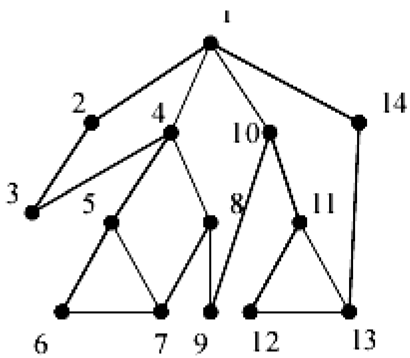
end.

Przykład

1) Załóżmy, że znaleziono następujące drzewo spinające pewnego grafu pełnego *G*

2) kolejność odwiedzania wierzchołków przez *DFS*

3) wyznaczamy cykl Hamiltona



Oszacowanie – proste!

PROBABILISTYCZNY ALGORYTM PRZYBLIŻONY

Założenia:

- duża liczba miast,
- miasta są położone wewnątrz prostokąta *X*

ALGORYTM:

Krok 1: Podzielić X na prostokąty zawierające podobną liczbę miast (rozkład!);
Krok 2: Wyznaczyć optymalną (przybliżoną) drogę komiwojażera dla każdego prostokąta;
Krok 3: Połączyć rozwiązania z prostokątów w jeden cykl komiwojażera (można to zrobić na wiele sposobów (np. losowo)

Uogólnienia:

1. różne grafy,
2. wielu komiwojażerów,
3. problemy trasowania pojazdów.