**Teorema 1. (slide 7)**

Nu există nicio valoare *c* pentru care sa existe un algoritm în timp polinomial și care să ofere o soluție cu un factor de aproximare *c* pentru TSP, decât dacă **P=NP**.

Presupunem ca exista un algoritm aproximativ de factor *c* care sa rezolve TSP.

fie problema gasirii unui ciclu hamiltonian intr-un graf G. Din graful G pot obtine un graf G’ cu proprietatea ca V(G)=V(G’), E(G’)=VxV -> G’ graf complet w(e)=1 daca e din E(G) si w(e)=c\*n daca e nu este din E(G).

G’ este un graf complet, ponderat, cu n noduri (nodurile preluate din G) ci ponderile muchiilor vor fi 1 pentru muchiile provenite din G si c\*n pentru muchiile adaugate pana la saturatie.   
Am presupus ca exista un algoritm c-aproximativ pentru rezolvarea TSP(G’)

daca G contine un ciclu hamiltonian, atunci traseul optim al comis-voiajorului inb graful G’ va avea un cost total de *n* (va avea n muchii, muchiile vor fi exact cele mostenite din ciclul hamiltonian al lui G, deci fiecare muchie va avea ponderea 1). Atunci algorimtul nostru aproximativ va gasi in G’ un ciclu hamiltonian de cost cel mult *c\*n*.

**daca G nu contine ciclu hamiltonian**, atunci traseul optim al comis-voiajorului va contine cel putin o muchie care nu este din G, deci care are cost c\*n. In total costul traseului optim in G’ va fi >*c\*n* (>=*c\*n+n-1)*. Deci **algoritmul nostru nu va gasi un traseu >*c\*n*.**

problema: vreau sa gasesc un ciclu hamiltonian in G:

* pe baza lui G contruiesc pe G’ (**timp polinomial** ~O(n^2))
* rulez algoritmul c-aprximativ pe G’ (timp polinomial)
* daca rezultatul<=c\*n - return DA, G este hamiltonian
* altfel return NU, G nu este hamiltonian.

Algoritmul descris mai sus rezolva problema ciclului hamiltonian in timp polinomial! Dar HCP este NPC, deci daca o rezolv in timp polinomial, atunci P=NP!  
  
Observatie (slide 12)

**Fie G un graf complet, ponderat, care respectă regula triunghiului. Și fie v1, v2, v3, …., vk un lanț în graful G. Atunci avem len((v1,vk))≤len(v1, v2, v3, …., vk)**

pasul 0 k=3;

len((v1,v3))<=len(v1,v2,v3)=len((v1,v2))+len((v2,v3)) - adevarat din regula triunghiului

pasul k:

**len((v1,vk))<=**len(v1,vk-1,vk)=len((v1,vk-1))+len((vk-1,vk))<=len(v1,v2,...vk-1)+ len((vk-1,vk))=**len( v1,v2,....,vk)**

**Lema 3 (slide17):**

**Fie OPT costul soluției optime pentru TSP, iar MST - ponderea totală a unui Arbore parțial de cost minim pe baza aceluiași graf. Avem relația**

**OPT≥MST   
  
Demonstratie:**Pp ca OPT<MST.

OPT este costul unui ciclu hamiltonian in G iar MST este costul arborelui partial de cost minim din G.

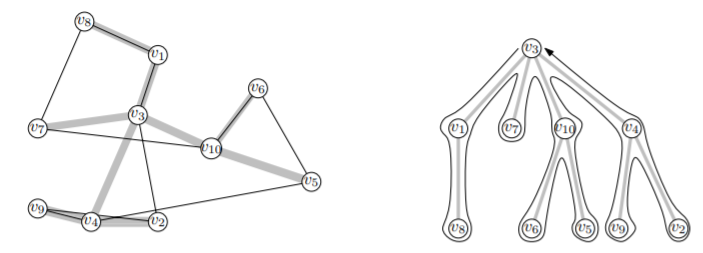
ciclu hamiltonian - contine toate nodurile (o singura data). Daca din acest ciclu scot o muchie, va ramane un lant de cost total <OPT. Lantul este un arbore (conex, n-1 muchii). Costul lantului (arborelui) obtinut <OPT < MST CONTRADICȚIE

**Teorema 4 (slide 20):**

**Algoritmul descris anterior (slides 18,19) este un algoritm 2-aproximativ pentru TSP**Demo:

fie OPT costul celui mai bun ciclu hamiltonian (OPT solutia optima pt TSP)

**Lema: Conturul unui MST (vezi figura de jos-dreapta) este de cost <=2\*OPT.**Demo Lema - conturul unui arbore implica traversarea fiecarei muchii exact de 2 ori. Costul conturului = 2\*MST<=2\*OPT

****

Revenind la teorema:

Pentru arborele din dreapta sus algoritmul nostru genereaza

𝛤={v3,v1,v8,v7,v10,v6,v5,v4,v9,v2,v3} (vezi desenul de sus-stanga)

Costul traseului 𝛤 va fi mai mic decat costul conturului de MST deoarece 𝛤 contine fie muchii din MST pe care le parcurge o singura data, fie muchii care “scurt-circuiteaza” conturul MST-ului (ex muchia v8-v7) care duce la o reducere a costului total (vezi observatie slide 12).  
**costul lui 𝛤**<=costul conturului**<=2\*OPT**