**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**.

Demonstratie:  
Presupunem ca exista un astfel de algoritm cu factor de aproximare *c*.

Fie G un graf simplu (neponderat). Se pune problema existentei unui ciclu hamiltonian in G. Aceasta problema este NPC.  
  
Construim un graf ponderat G’ pornind din G astfel:  
V(G’)=V(G).   
toate muchiile din G’ care se gasesc in G vor avea ponderea 1. Completam restul de muchii pana cand G’ este graf complet. Muchiile care nu provin din G vor avea ponderea *c\*n*.

Daca **G contine un ciclu hamiltonian**, atunci traseul optim al comis-voiajurului in graful G’ va fi de cost *n*. Iar **algoritmul nostru va oferi un traseu de cost total cel mult *c\*n***

Daca **G nu contine un ciclu hamiltonian.** Atunci traseul optim in G’ ar contine macar o muchie care nu provine din G. Deci costul total al traseului va fi >c\*n (>=c\*n+n-1). Atunci **algoritmul nostru va oferi un traseu de cost >c\*n**

Fie urmatorul algoritm ptru HC-Problem de G:

* Din G obtin graful G’. (complexitate O(n^2) - polinomial)
* gasesc un traseu de cost ALG care este c-aproximativ pentru problema TSP(G’) (complexitate - polinomial)
* Daca ALG<=c\*n atunci G este hamiltonian
  + altfel G nu este hamiltonian.

Algoritmul de mai sus rezolva HCP in timp polinomial (daca exista acel algoritm c-aproximativ pt TSP). HCP este problema NPC. Rezulta ca P=NP.   
  
**Observație 2 (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)**

Inductie:

pt k=3:

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

presupunem inegalitatea adevarata pentru toate lanturile de cel mult k-1 noduri. Vrem sa aratam ca este valabila si pt lanturi formate din *n* noduri

len((v1,v\_{k}))<=len(v1,v\_{k-1},v\_{k})=len((v1,v\_{k-1}))+len ((v\_{k-1},v\_{k})) <=len((v1,v2,...,v\_{k-1}))+len ((v\_{k-1},v\_{k}))=len(v1,v2,...v\_{k})  
  
**Lema 3 (slide 17):**

**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   
  
Demo:**

**Presupunem ca OPT<MST.  
OPT este costul unui ciclu (hamiltonian) ce contine toate cele n noduri (componenta conexa). Daca eliminam o muchie din acest ciclu rezulta un lant care contine toate nodurile. Costul lantului (arborelui) < OPT<MST** *Contradictie!*

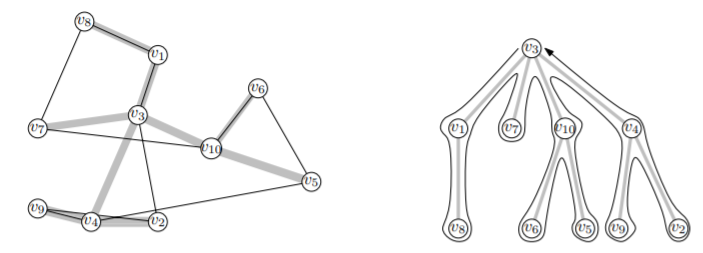
**Teorema 4:**

**Algoritmul descris anterior este un algoritm 2-aproximativ pentru TSP**

**Preludiu la demonstratie**

**Daca avem un arbore partial de cost minim, costul conturului arborelui (vezi figura din dreapta-jos) va fi <=2\*OPT**

**conturul arborelui este format parcurgand fiecare muchie din arbore de exact 2 ori. Costul conturului = 2\*MST<=2\*OPT**

****

**pentru exemplul de mai sus 𝛤={v3,v1,v8,v7,v10,v6,v5,v4,v9,v2,v3}**

**𝛤 - va fi lista nodurilor vizitate intr-un DFS al arborelui + nodul radacina. In lista 𝛤 vor exista muchii directe (care se regasesc si in arbore, ex:(v1,v8)) dar si muchii care scurt-circuiteaza arborele (ex: (v8,v7)). Aceste muchii care scurt-circuiteaza reduc costul total fata de costul conturului (vezi regula triunghiului).**

**costul lui 𝛤<=costul conturului<=2\*OPT**