3. Vertex cover → set cover

Sa se reduca problema vertex cover la problema set cover.

Set cover: Dandu-se o multime de S elemente si N submultimi S1,S2,… Sn, se pot gasi k submultimi a.i. reuniunea lor sa dea multimea S?

Sa se scrie functia de reducere in C++ si rezolvarea pentru problema set cover in C++, a.i. sa se poata rezolva problema vertex cover pe baza acestei functii de reducere si pe baza rezolvarii problemei set cover.

Rezolvarea codului s-a facut in Python intrucat implementarea a fost mai facila.

K\_ACOPERIRE =<p SET COVER

K\_acoperire : Fie G=(V,E) un graf neorientat. Multimea V'⊂V se numeste K\_acoperire a lui G daca:

-> card(V')=k

-> oricare (u,v)∈E avem u∈V' &| v∈V'

multime

|

G=(V,E)------>S,(Si)i,h

| \ \

| \ \

| multime submultimi pentru S

| arce

|

multime

noduri

a)ALGORITMUL DETERMINIST TRACTABIL

- se pleaca de la un graf G si un numar k, graficul G avand o k\_acoperire

- se considera S=E (multimea muchiilor)

- se formeaza Si subseturi unde i=1,n (n-numarul de noduri)

- se constata ca h=k

- ALG\_Set\_Cover(G,k) //G=(V,E) -> n=card(V) & m=card(E)

S = E //multime de arce

h = k //multimea de Si care alcatuiesc S-ul

for\_each vi∈V //i=1,n

for\_each ej∈E //j=1,m

if vi este capat al lui ej

Si = ej //se adauga in submultimea coresunzatoare fiecarui

//nod muchiile ce pleaca din nodul respectiv

return S,(Si)i,h

b)ECHIVALENTA IESIRILOR

- demonstrez ca G are o k\_acoperire <=> S are o subcolectie cu cardinalul h

"=>" Fie V'⊂V se numeste K\_acoperire a lui G ; V'={vi1, vi2, vi3, ..., vik}

-> card(V')=k

-> oricare (u,v)∈E avem u∈V' &| v∈V'

Fie S'={xi1, xi2, xi3, ..., xik} U {yj / ej are un singur capat in V'} ; S'=⊂S

- avem k valori de x in S'

- daca ej are ambele capete in V' => yj!∈S'; xa,xb∈S' => ej=(va,vb) => ej este a j-a muchie din submultimea Si

- daca ej are un singur capat in V'; fie va un capat => yj∈S'; xa∈S' => ultima muchie din multimea Si este a j-a muchie

"<=" Fie S'⊂S a.i. card S'=k, k=h, ultima muchie din S' este a k-a la numar, S'={xi1, xi2, xi3, ..., xik} U {yj1, yj2, yj3, ..., yjk} <=> h=k

- aratam ca V'={vi1, vi2, vi3, ..., vik}⊂V este k\_acoperire a lui G

- 0<j<m-1 : consideram arcul ej

- yj∈S' => un singur x este in S':fie xa acel x => nodul este in V' si este singurul capat al lui ej in V'

- yj!∈S' => exista 2 valori de x in S, xa, xb => va, vb∈V' => ej are ambele capete in V'

===> V' este o k\_acoperire a lui G

Am aratat ca:

(S,(Si)i,h) se poate obtine din (G,k) folosind un algoritm determinist tractabil si ca G are o k\_acoperire <=> S are o subcolectie Si cu k elemente => k\_acoperire =<p Set\_cover