2. Graceful labeling

a)1.5p alg && 0.5p complexitate

b)1.5p alg && 0.5p complexitate

Numerotare gratioasa(varianta)

O numerotare gratioasa a unui graf orientat G=(V,E)

n=card(V)

m=card(E)

presupune atribuirea pentru fiecare nod a unui numar unic intre 1 si n, a.i. pentru fiecare arc(u,v) diferenta dintre numar(u)-numar(v) sa fie unica si sa acopere multimea {1...m}.

a)Scrieti un alg. nedeterminist care sa verifice daca graful G poate avea o numerotare gratioasa. Calculati complexitatea temporala a algoritmului pentru cazul cel mai defavorabil

b)Scrieti un alg. determinist care sa verifice daca graful G poate avea o numerotare gratioasa. Estimati complexitatea ca la a

a)

ALG\_nedeterminist() //G=(V,E)

for\_each v∈V //generare pt fiecare nod un numar ==> O(n)

numar(v)=choice({1...n})

for\_each n∈V //verificare unicitate numere generate ==> O(n)

for\_each v∈V\{n} //O(n-1)

if numar(u) = numar(v) //O(1)

fail

else

succes

if succes //daca nodurile au numere diferite

for\_each (u,v)∈E //O(m)

if numar(u) - numar(v) ∈ {1...m} //O(1)

fail

for\_each (u',v')∈E\{(u,v)} //O(m-1)

if(numar(u)-numar(v)) = (numar(u')-numar(v')) //O(1)

fail

Complexitate temporala polinomiala : O(n)\*O(n-1) + O(m)\*O(m-1) = O(n^2-n-m^2-m)

b)

ALG\_determinist() //G=(V,E)

for\_each v∈V //atribuie random numerele nodurilor

for\_each u∈V\{v} //O(n)

if numar(u) = numar(v) //se verifica unicitatea nodurilor ==> O(n-1)

count = 0

else

count = 1

if count = 1

for\_each (u,v)∈E //O(m)

if numar(u) - numar(v) ∉ {1...m} //O(1)

for\_each (u',v')∈E\{(u,v)} //O(m-1)

if(numar(u) - numar(v)) = (numar(u')-numar(v')) //O(1)

count = 0

else

count = 1

Complexitate temporala polinomiala : O(n)\*O(n) + O(m)\*O(m-1) = O(n^2-m^2-m)

Varianta cu backtracking

//numar[]->vector cu noduri

functie\_verificare(graf, numar[]) //G=(V,E), n=card(V), m=card(E) ----> O(n^2)

for\_each v∈V //atribuie random numerele nodurilor -->O(n)

for\_each u∈V\{v} //O(n-1)

if numar(u) = numar(v) //se verifica unicitatea nodurilor ==> O(n-1)

count = 0

else

count = 1

if count = 1

for\_each (u,v)∈E //O(m)

if numar(u) - numar(v) ∉ {1...m} //O(1)

for\_each (u',v')∈E\{(u,v)} //O(m-1)

if(numar(u) - numar(v)) = (numar(u')-numar(v')) //O(1)

count = 0

else

count = 1

functie\_backtracking(graf, k, numar[]) ----> O(n^n) + O(k^n) = O(k^n)

if k==n

if functie\_verificare(graf, numar[])=1

count = 1

else

count = 0

i = 0

while(i < n)

numar[k]=i

i++

if functie\_backtracking(graf, k+1, numar)

count = 1

numar[i] = 0

i++

else

count = 0

i++

Bibliografie

https://www.geeksforgeeks.org/m-coloring-problem-backtracking-5/