Tema Cocheci Cristiana 252

Lista submisii:

<https://leetcode.com/problems/possible-bipartition/submissions/1079764884/>

<https://leetcode.com/submissions/detail/1077672104/>

<https://leetcode.com/submissions/detail/1079761164/>

<https://leetcode.com/submissions/detail/1080512980/>

<https://leetcode.com/submissions/detail/1080574776/>

<https://leetcode.com/submissions/detail/1079791649/>

<https://www.infoarena.ro/job_detail/3161575?action=view-source>

<https://www.infoarena.ro/job_detail/3160425?action=view-source>

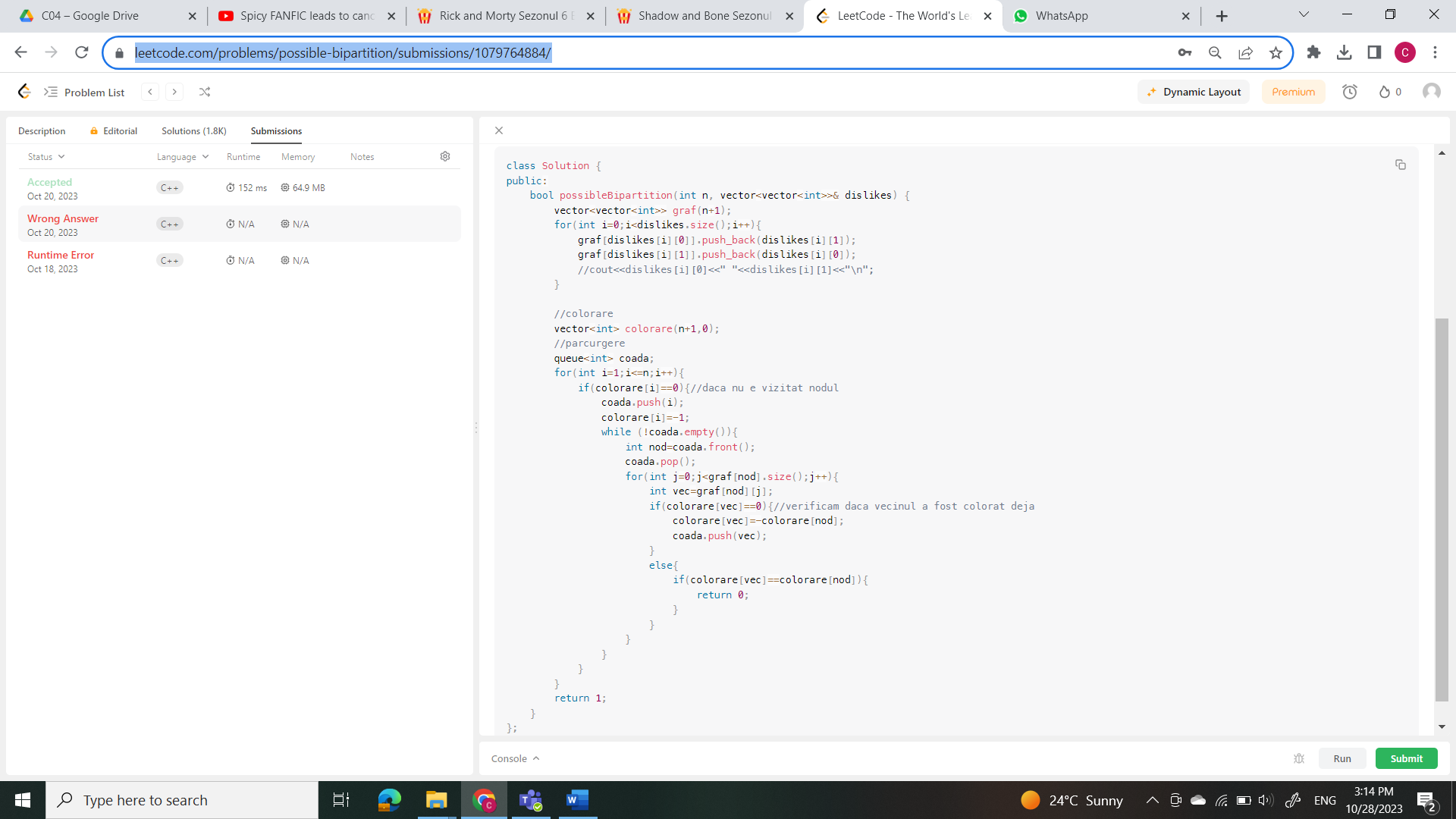
<https://codeforces.com/contest/1881/submission/229445509>

<https://codeforces.com/contest/1144/submission/229452986>

1.- <https://leetcode.com/problems/possible-bipartition/>

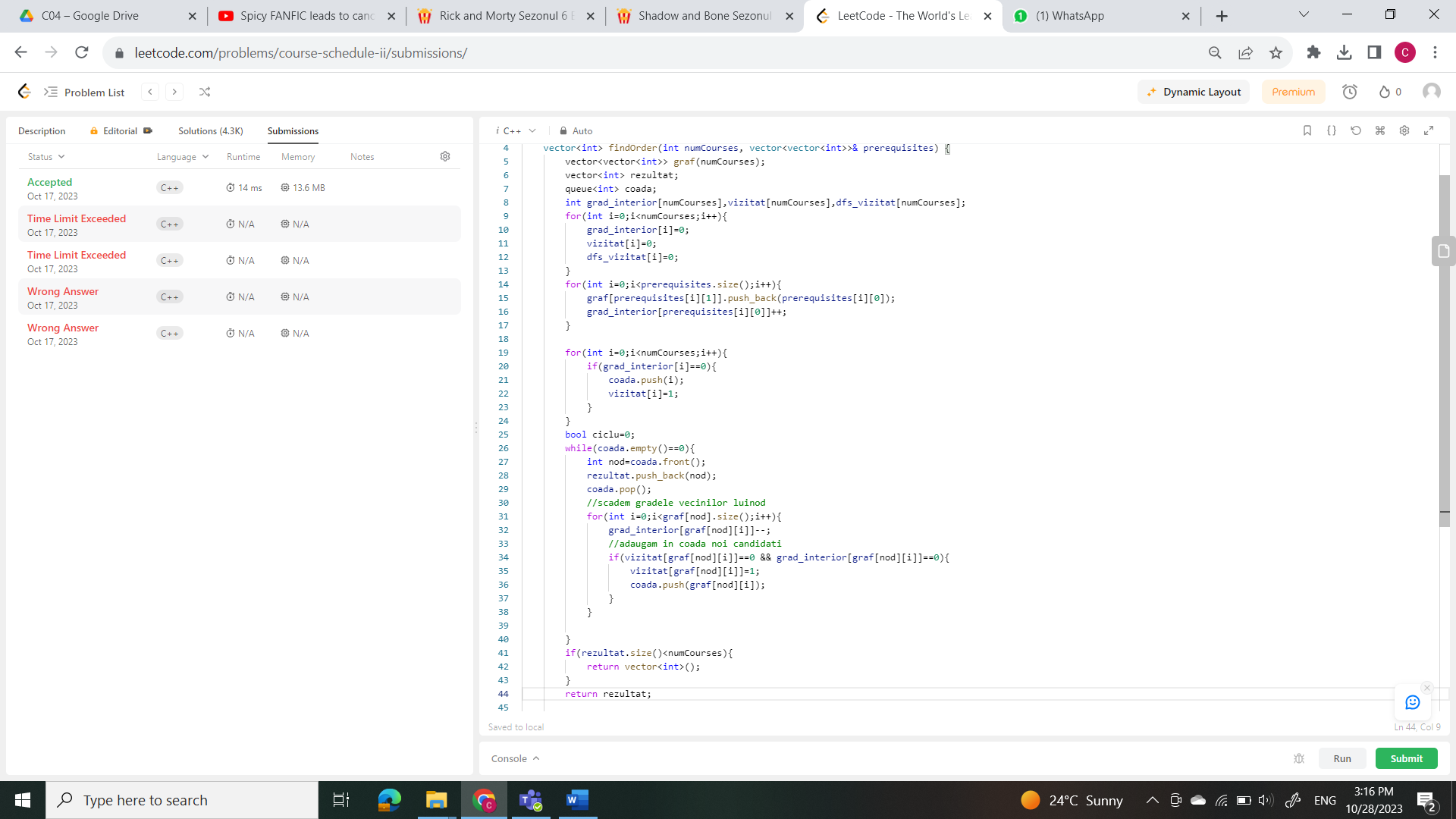
Link Rezolvare:

https://leetcode.com/problems/possible-bipartition/submissions/1079764884/



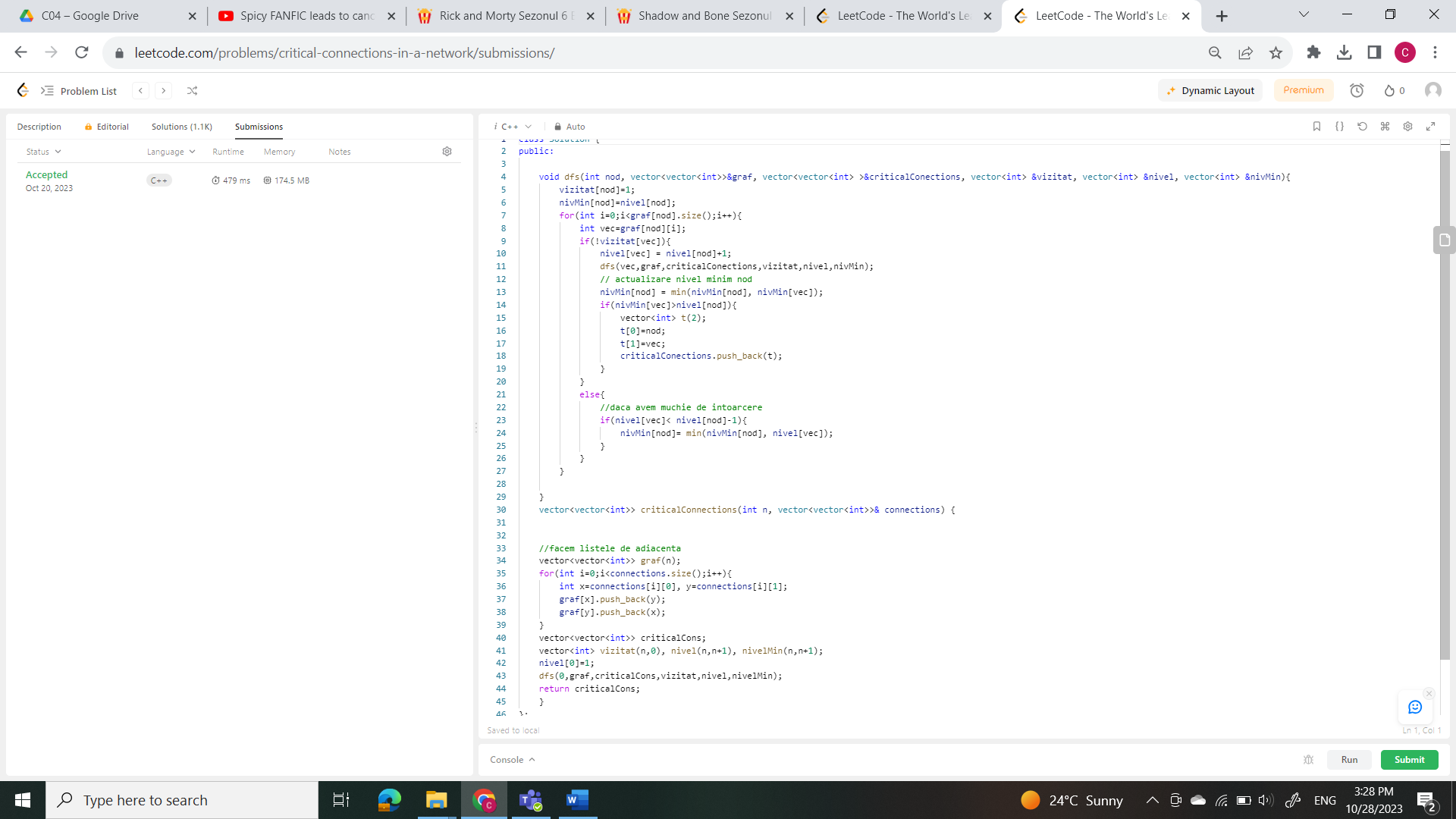
2.- <https://leetcode.com/problems/course-schedule-ii/>

Rezolvare: <https://leetcode.com/submissions/detail/1077672104/>



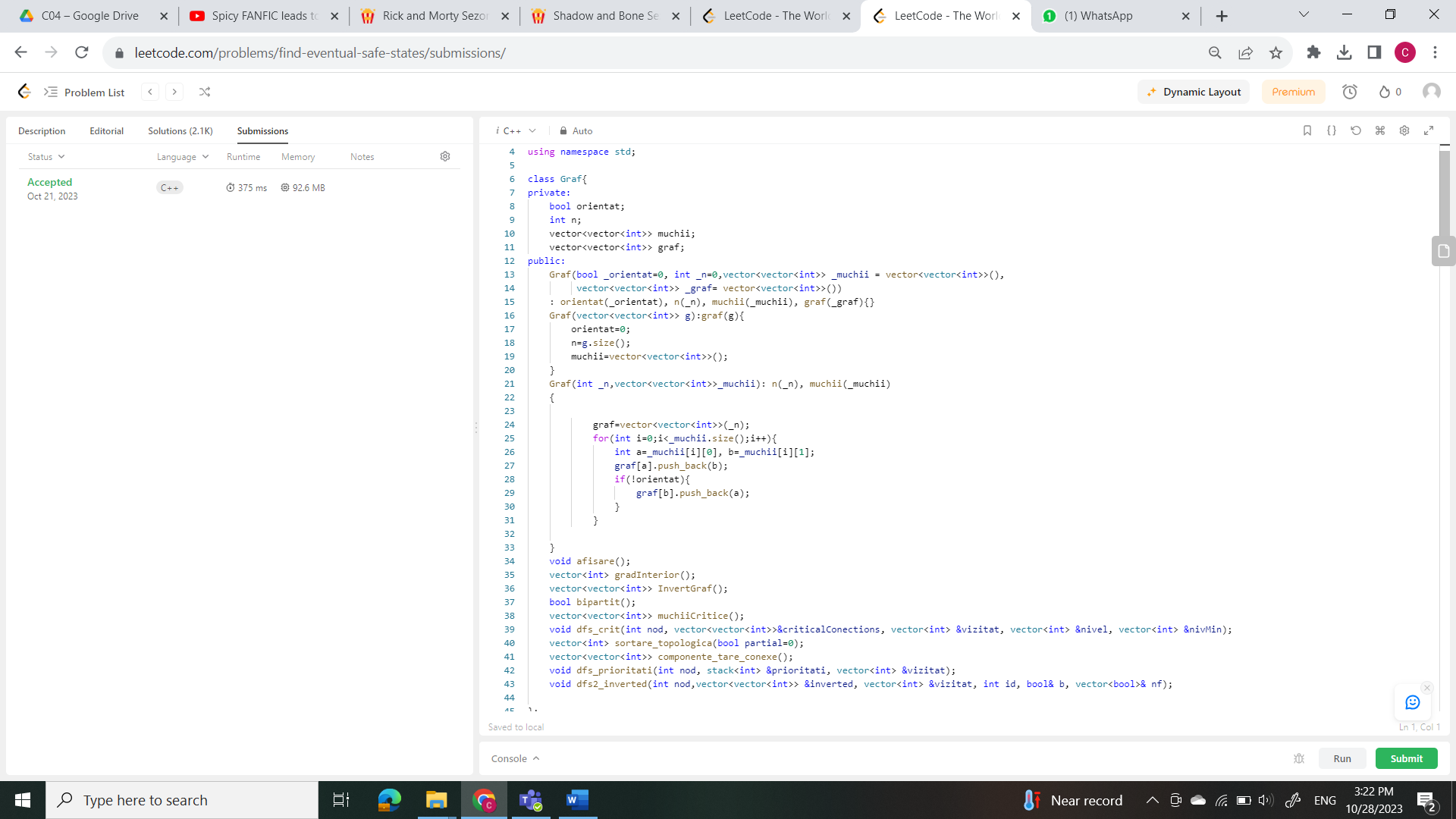
3.- <https://leetcode.com/problems/critical-connections-in-a-network>

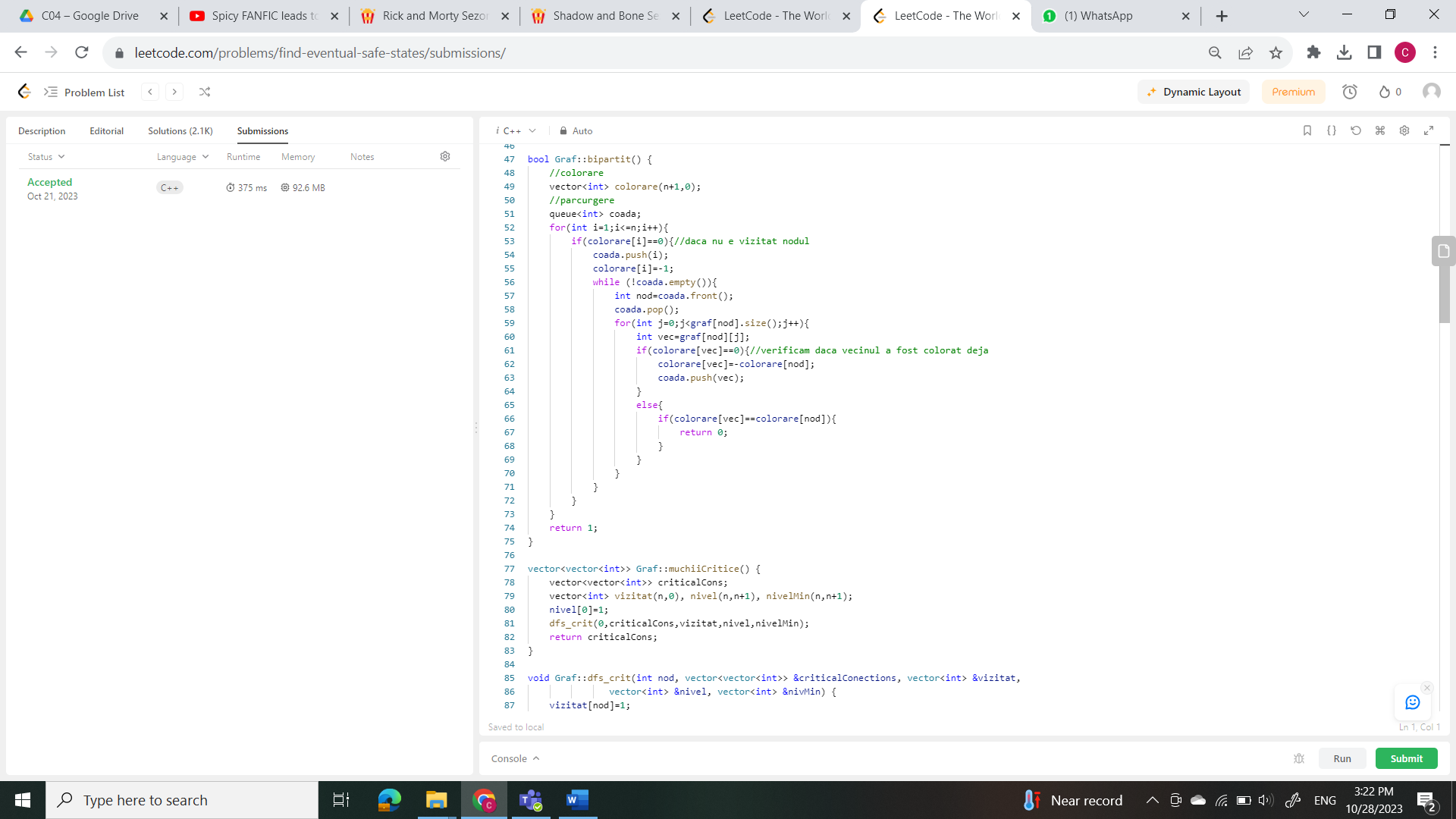
Rezolvare: <https://leetcode.com/submissions/detail/1079761164/>

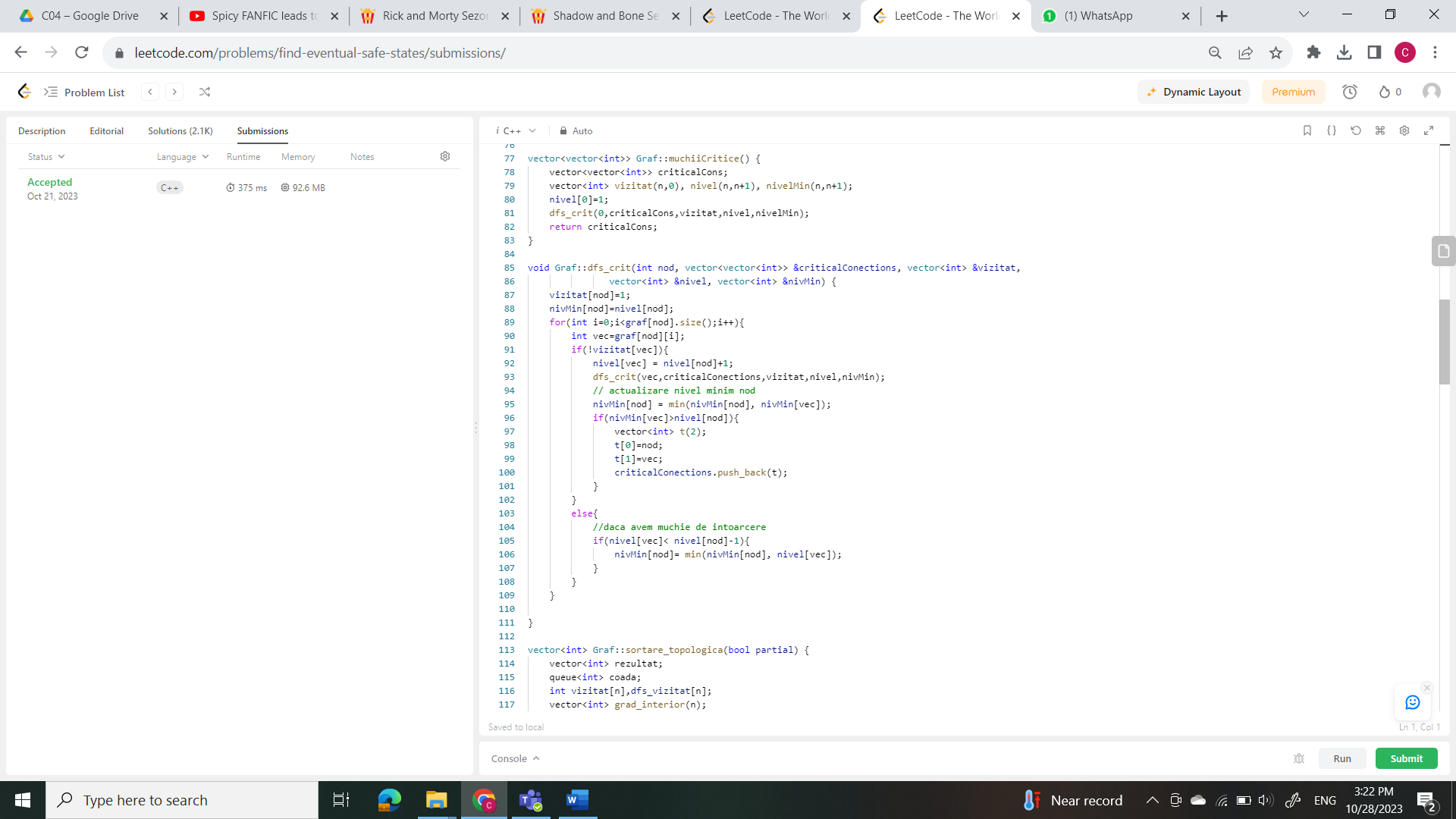
­

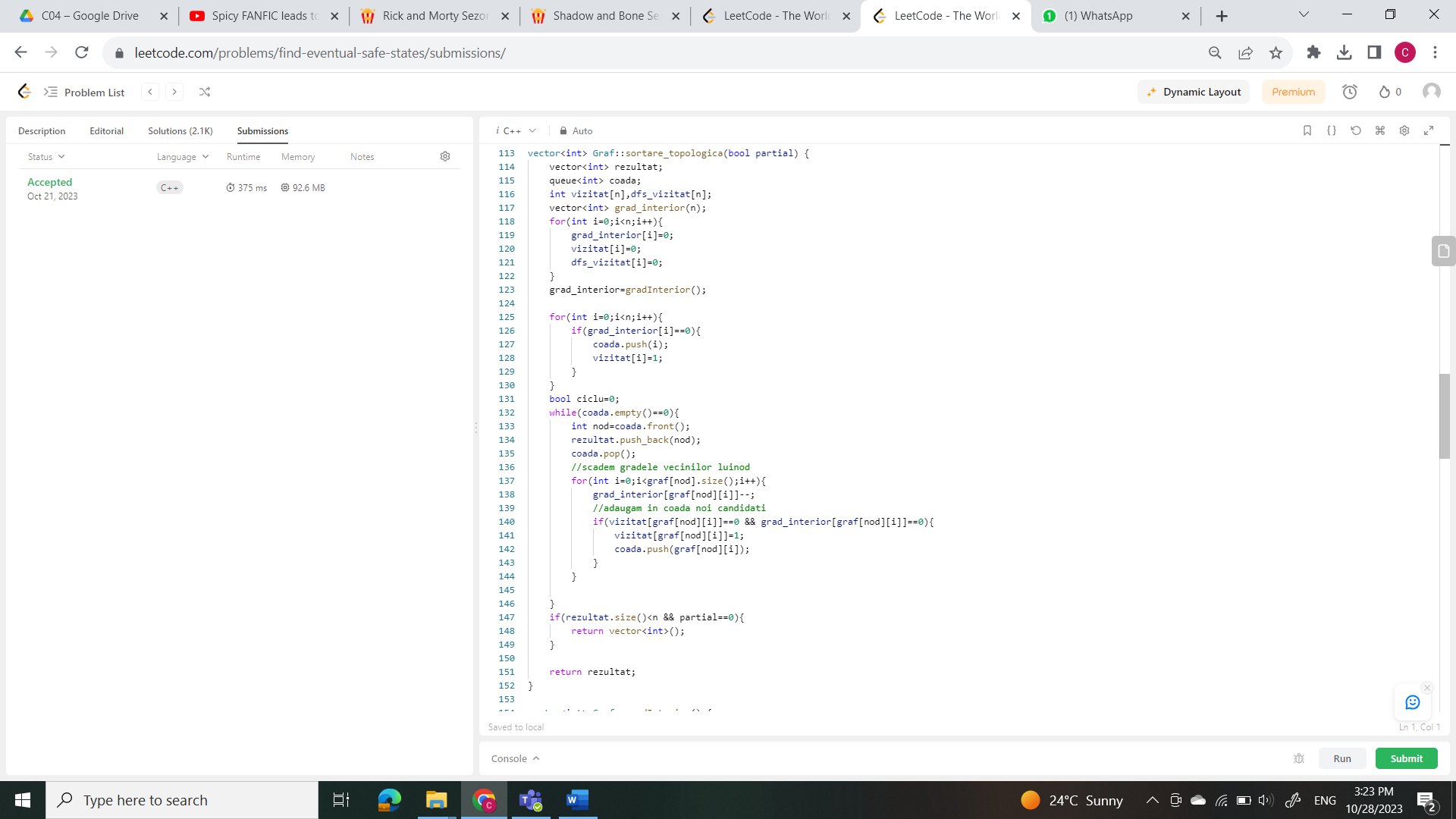
4.- <https://leetcode.com/problems/find-eventual-safe-states/>

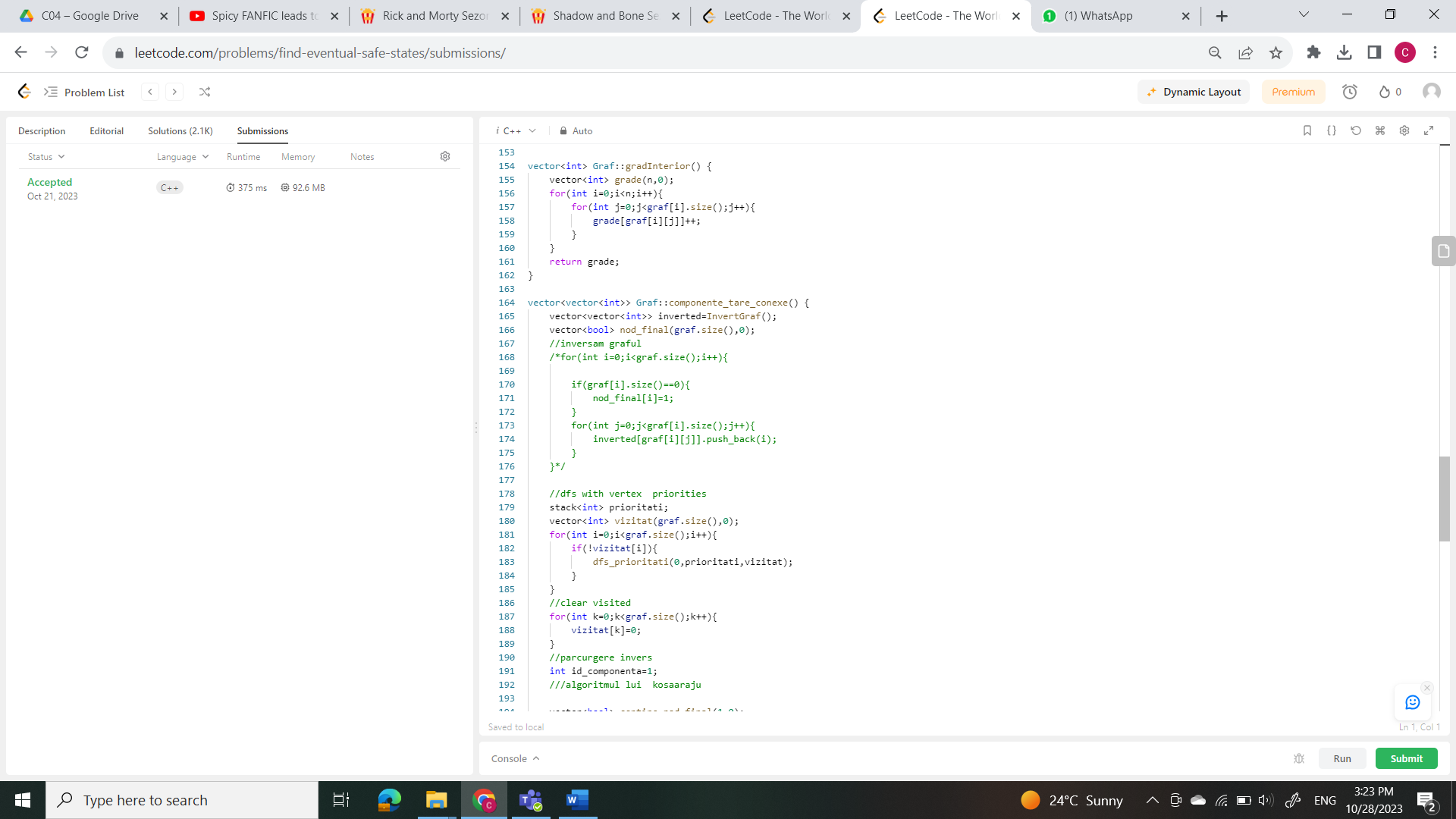
Rezolvare: <https://leetcode.com/submissions/detail/1080512980/>

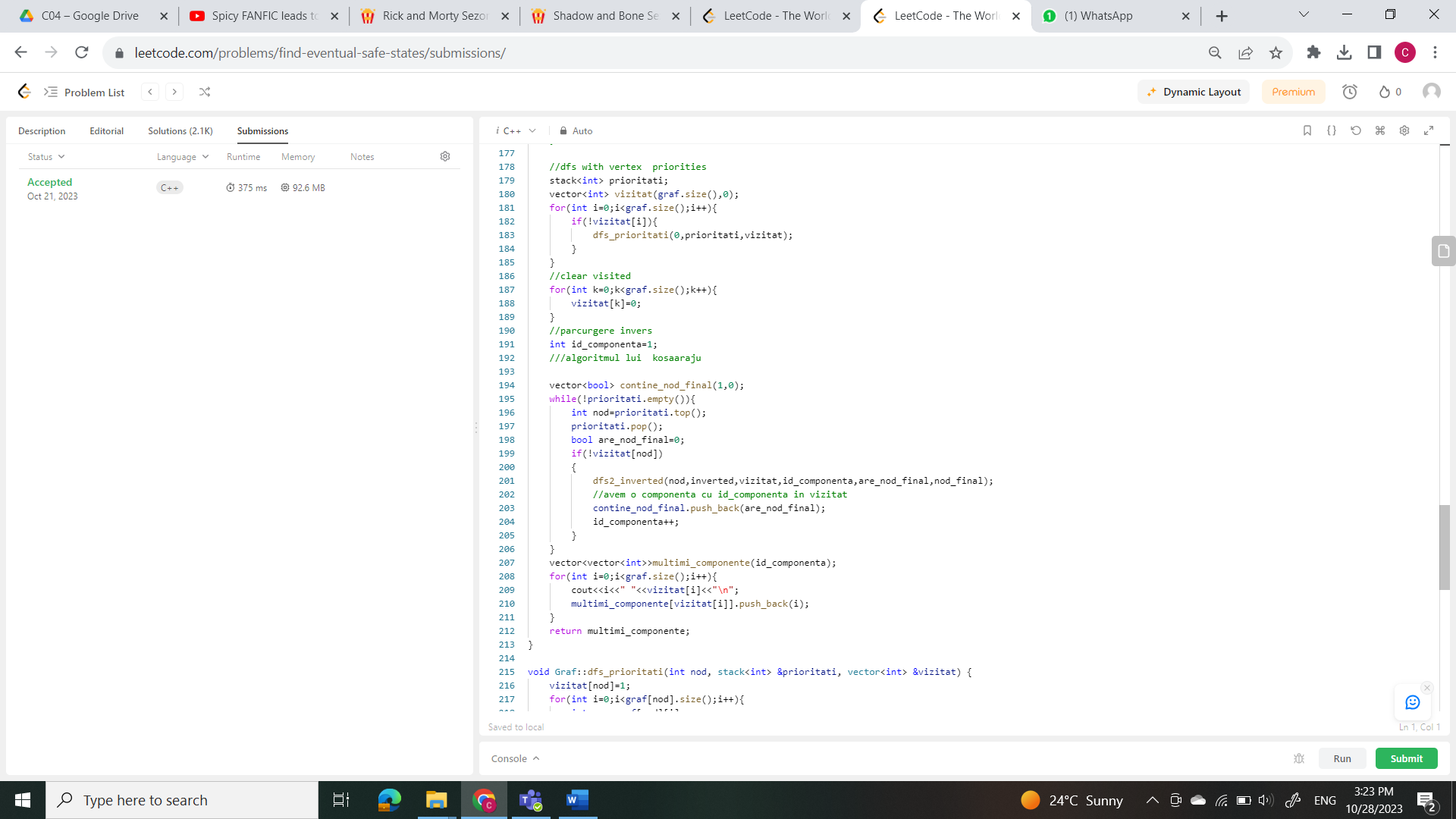


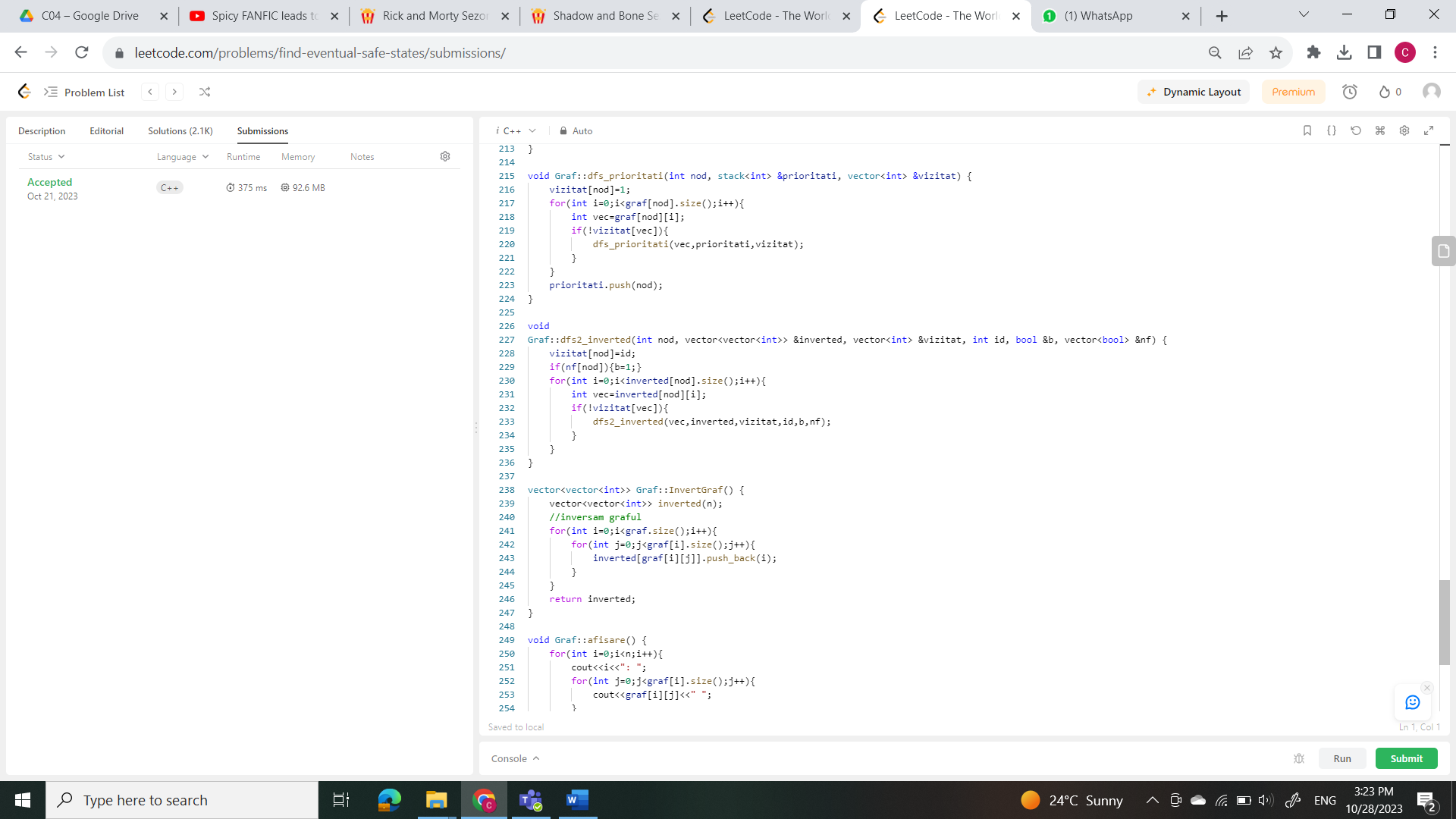


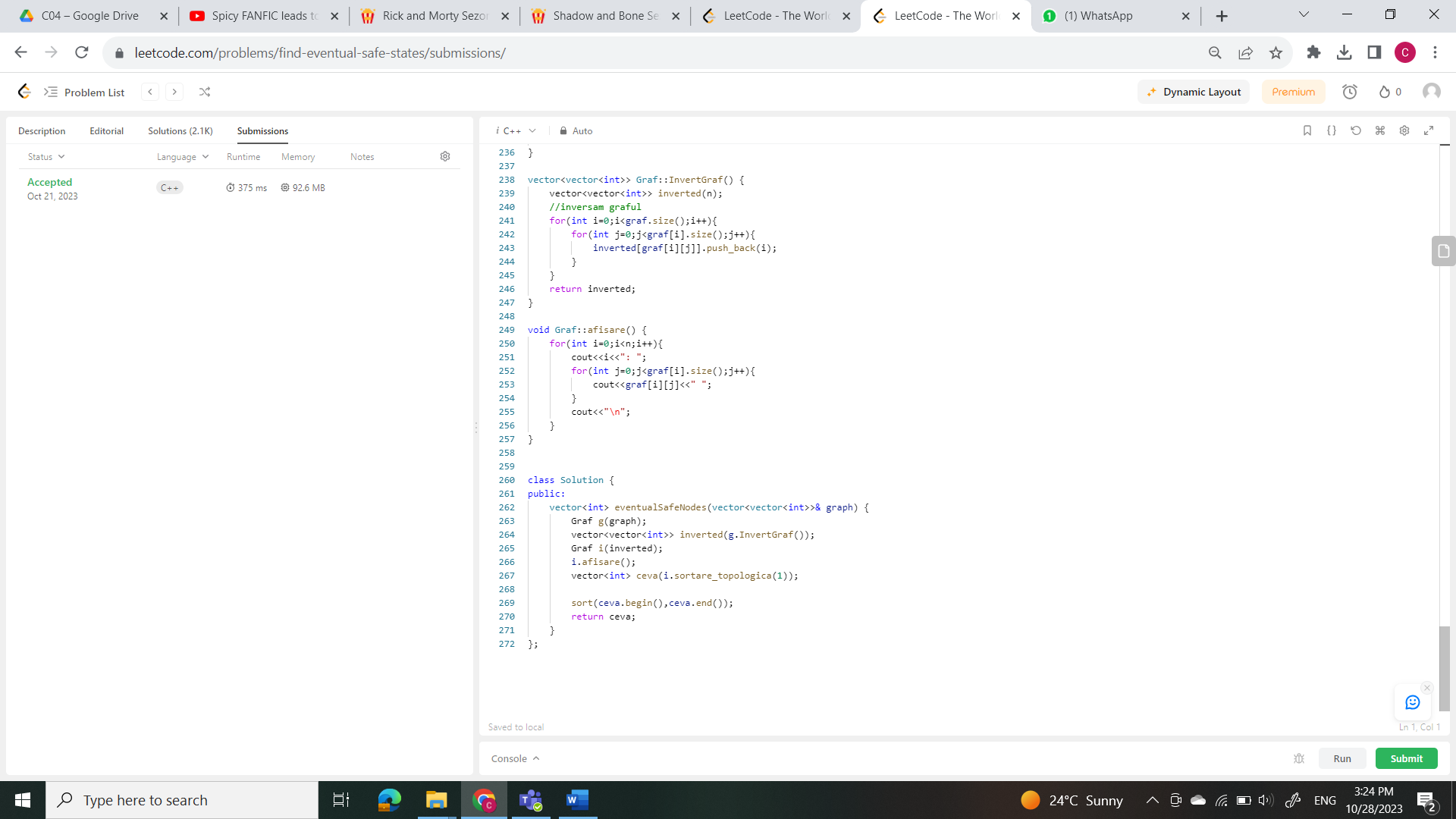






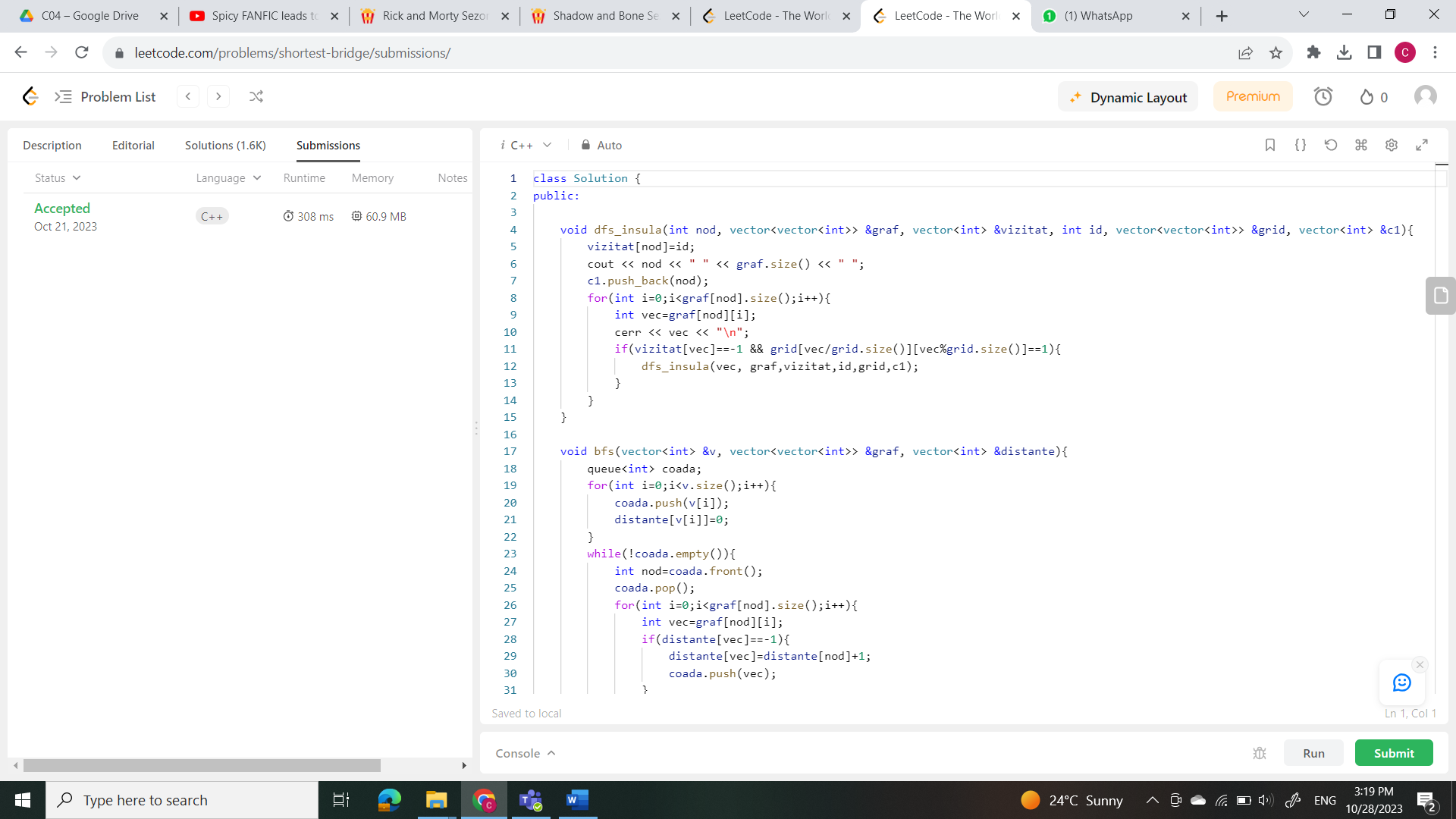


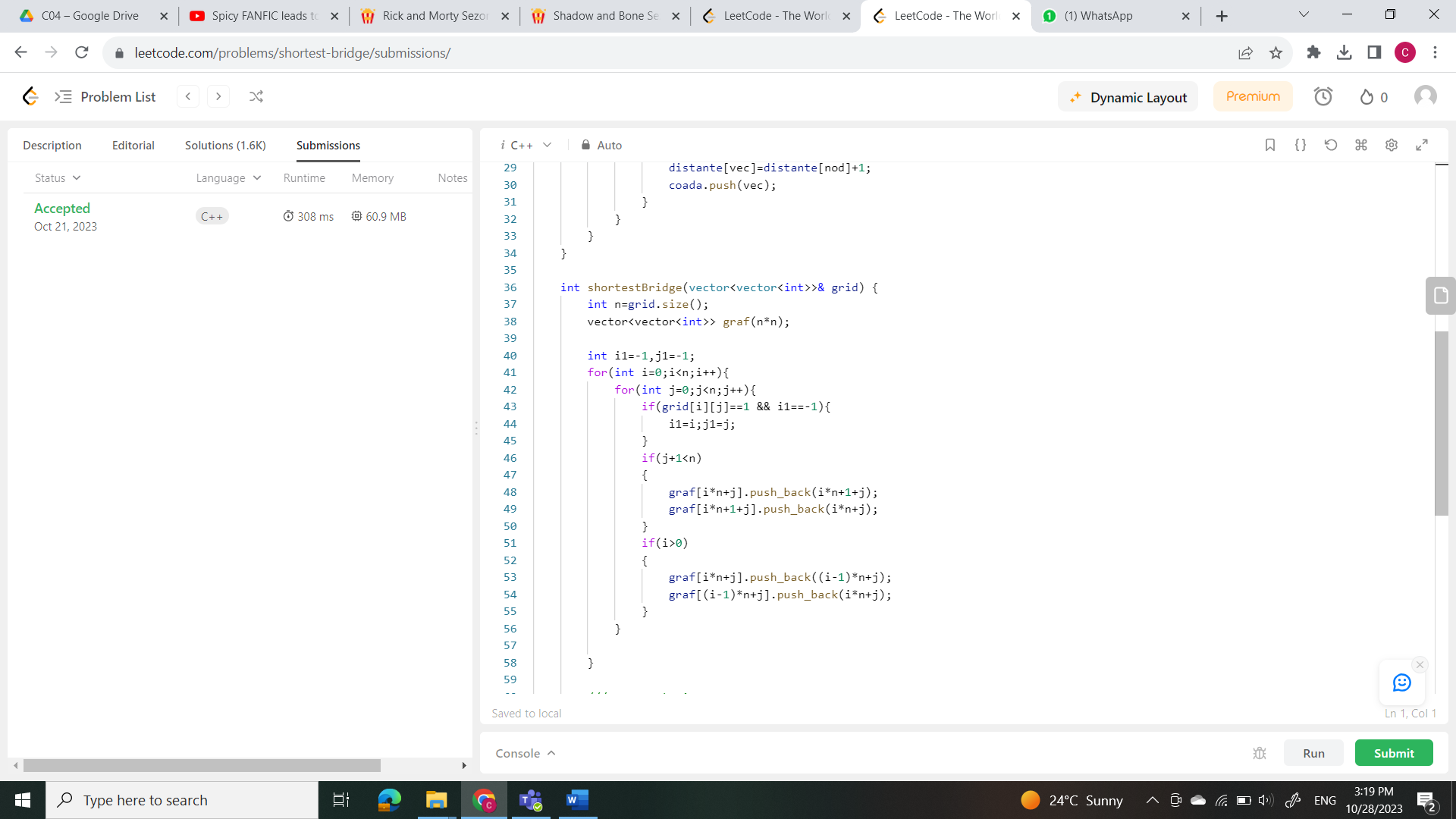


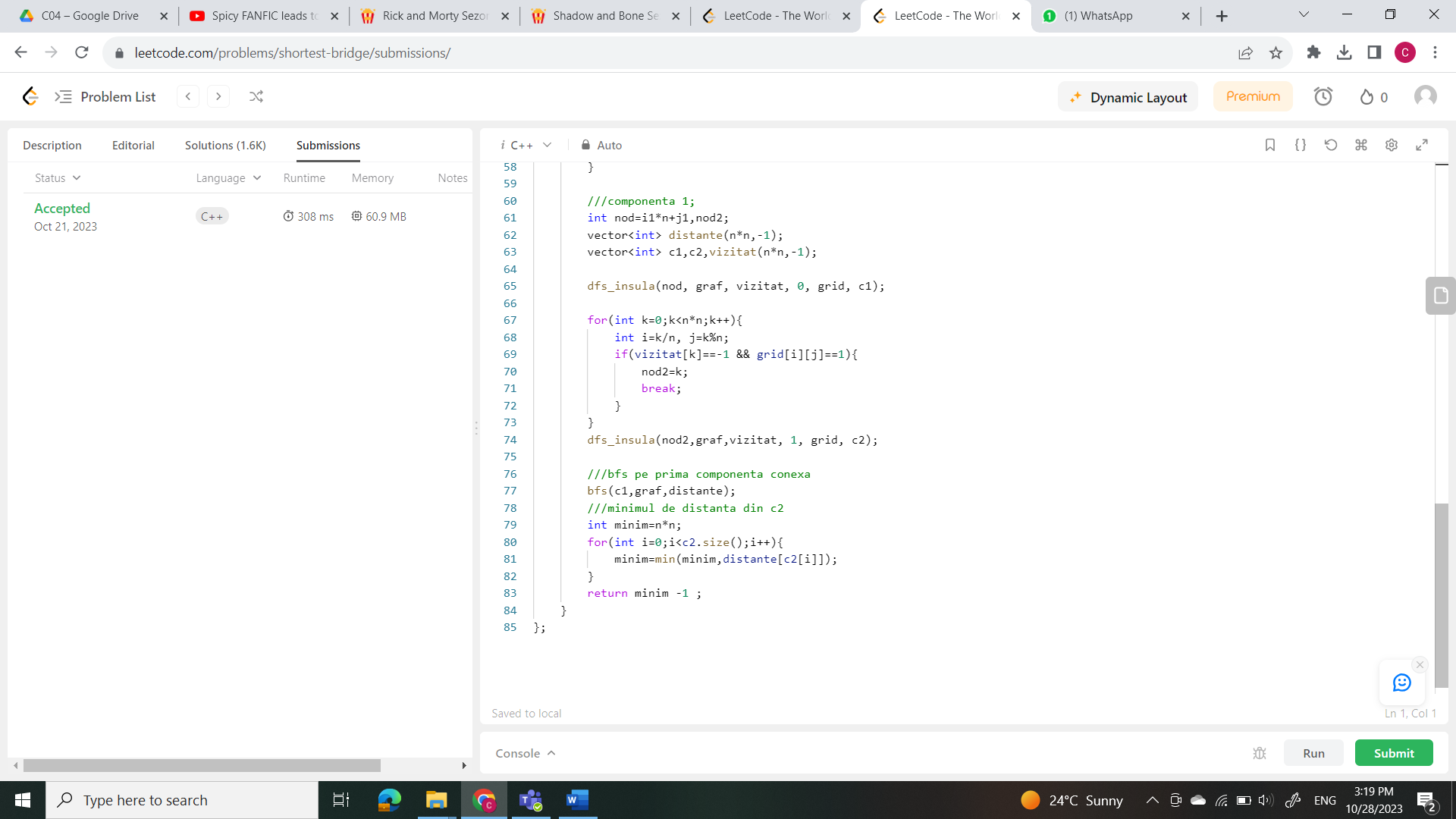


5.- <https://leetcode.com/problems/shortest-bridge/>

Rez: <https://leetcode.com/submissions/detail/1080574776/>

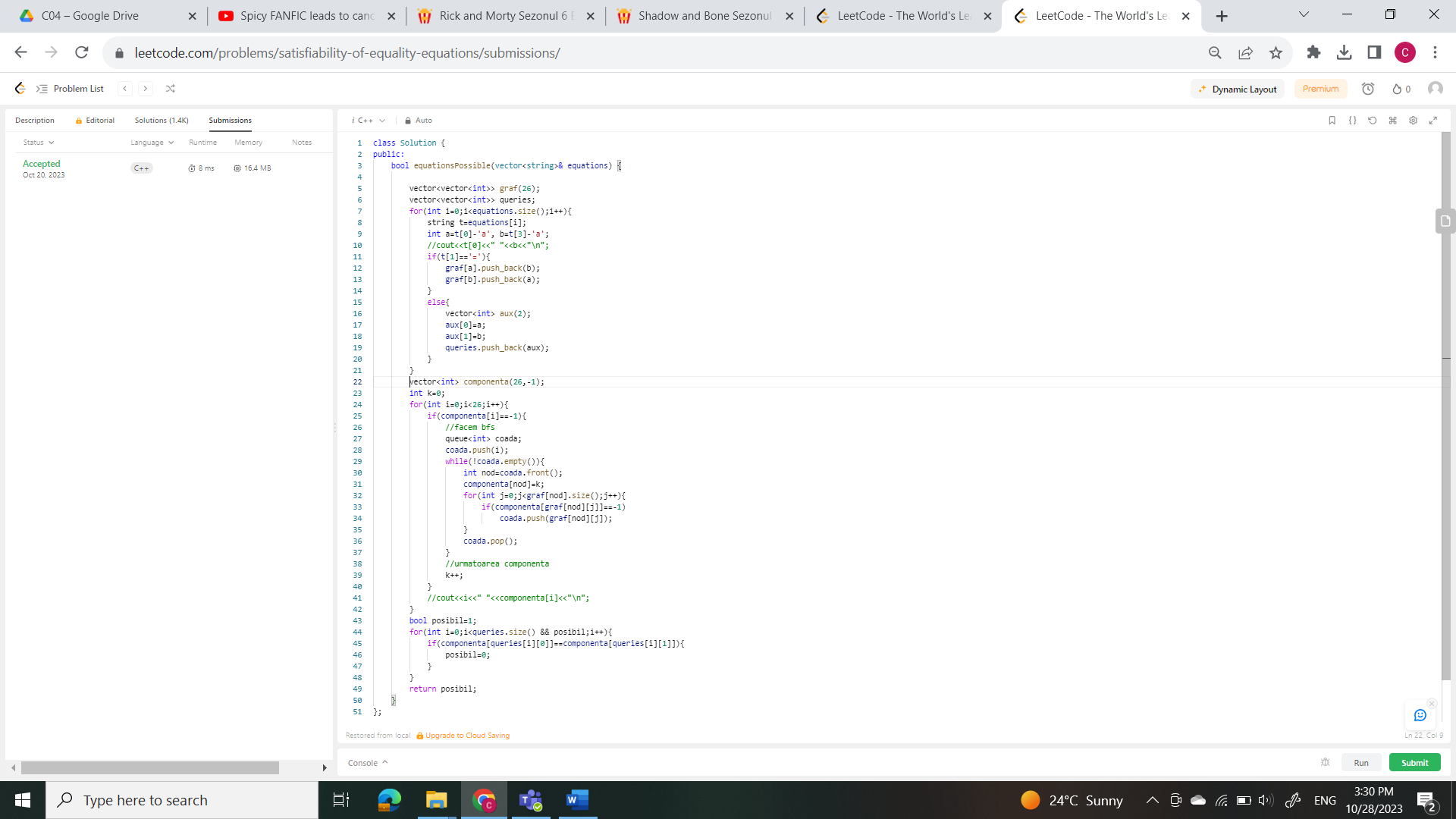






6.- <https://leetcode.com/problems/satisfiability-of-equality-equations/>

Rez: <https://leetcode.com/submissions/detail/1079791649/>



Grele

7.- <https://www.infoarena.ro/problema/padure>

Link solutie: <https://www.infoarena.ro/job_detail/3161575?action=view-source>

#include <iostream>

#include <bits/stdc++.h>

using namespace std;

vector<vector<int>> matrice(1000,vector<int>(1000));

int INF=1e9;

int vizitat[10000005];

int v1[4]={0,0,1,-1};

int v2[4]={1,-1,0,0};

int main()

{

freopen("padure.in","r",stdin);

freopen("padure.out","w",stdout);

int pl,pc,cl,cc,n,m;

cin>>n>>m>>pl>>pc>>cl>>cc;

vector<vector<pair<int,int>>> graf(n\*m);

pl--;pc--;cl--;cc--;

for(int i=0;i<n;i++){

for(int j=0;j<m;j++){

cin>>matrice[i][j];

}

}

int distante[n\*m+1];

for(int i=0;i<n;i++){

for(int j=0;j<m;j++){

distante[i\*m+j]=INF;

}

}

*//dijsktra*

distante[pl\*m+pc]=0;

*//sursa = pl,pc*

deque<int> coada;

coada.push\_back(pl\*m+pc);

while(!coada.empty()){

int nod=coada.front();

coada.pop\_front();

for(int k=0;k<4;k++){

if(nod/m+v1[k]<n && nod/m+v1[k]!=-1 && nod%m+v2[k]<m &&nod%m+v2[k]!=-1){

int vec= (nod/m+v1[k])\*m+nod%m+v2[k];

if(matrice[nod/m][nod%m]==matrice[vec/m][vec%m]){

if(distante[vec]>distante[nod]){

distante[vec]=distante[nod];

if(coada.empty()){

coada.push\_back(vec);

}

else{

if(distante[coada.front()]>=distante[vec]){

coada.push\_front(vec);

}

else{

coada.push\_back(vec);

}

}

}

}

else{

if(distante[vec]>distante[nod]+1){

distante[vec]=distante[nod]+1;

if(coada.empty()){

coada.push\_back(vec);

}

else{

if(distante[coada.front()]>=distante[vec]){

coada.push\_front(vec);

}

else{

coada.push\_back(vec);

}

}

}

}

}

}

}

cout<<distante[cl\*m+cc];

return 0;

}

8.- <https://www.infoarena.ro/problema/patrol2>

Link solutie: <https://www.infoarena.ro/job_detail/3160425?action=view-source>

#include <iostream>

#include<bits/stdc++.h>

using namespace std;

int Cmmmc(int a, int b){

int p=a\*b;

while(b!=0)

{

int r=a%b;

a=b;

b=r;

}

return p/a;

}

int main()

{

freopen("patrol2.in","r",stdin);

freopen("patrol2.out","w",stdout);

int n,m,k;

cin>>n>>m>>k;

vector<vector<int>> graf(n), politisti(k);

for(int i=0;i<m;i++){

int a,b;

cin>>a>>b;

graf[a].push\_back(b);

graf[b].push\_back(a);

}

*///construim matrice in care vedem daca la un timp %cmmmc(Li) un nod este liber*

int cmmmc=1;

for(int i=0;i<k;i++){

int l;

cin>>l;

politisti[i].push\_back(l);

cmmmc=Cmmmc(cmmmc,l);

for(int j=0;j<l;j++){

int x;

cin>>x;

politisti[i].push\_back(x);

}

}

*//cout<<cmmmc<<"\n";*

int nod\_ocupat[n][cmmmc];

for(int i=0;i<n;i++){

for(int j=0;j<cmmmc;j++){

nod\_ocupat[i][j]=0;

}

}

*/// actualizam matricea*

for(int pol=0;pol<k;pol++){

int l=politisti[pol][0];

for(int i=1;i<=l;i++){

for(int k=0;k<cmmmc/l;k++){

nod\_ocupat[politisti[pol][i]][l\*k+i-1]=1;

*//cout<<"nod\_ocupat: "<<politisti[pol][i]<<" "<<l\*k<<"\n";*

}

}

}

*/\*for(int i=0;i<n;i++){*

*for(int j=0;j<cmmmc;j++){*

*cout<<nod\_ocupat[i][j]<<" ";*

*}*

*cout<<"\n";*

*}\*/*

*/// facem bfs pentru a face a doua matrice, matricea in care vedem*

*/// timpul minim in care ajungi in celula respectiva*

int timp\_minim[n][cmmmc];

for(int i=0;i<n;i++){

for(int j=0;j<cmmmc;j++){

timp\_minim[i][j]=-1;

}

}

timp\_minim[0][0]=0;

queue<pair<int,int>> coada; *//nod timp*

if(!nod\_ocupat[0][0]){

coada.push({0,0});

}

while(!coada.empty()){

pair<int,int> nod\_timp=coada.front();

int nod=nod\_timp.first;

int timp=nod\_timp.second;

coada.pop();

if(!nod\_ocupat[nod][(timp+1)%cmmmc] && timp\_minim[nod][(timp+1)%cmmmc]==-1){

timp\_minim[nod][(timp+1)%cmmmc]=timp\_minim[nod][timp]+1;

coada.push(make\_pair(nod,(timp+1)%cmmmc));

}

for(int i=0;i<graf[nod].size();i++){

int vec=graf[nod][i];

if(!nod\_ocupat[vec][(timp+1)%cmmmc] && timp\_minim[vec][(timp+1)%cmmmc]==-1)

{

coada.push(make\_pair(vec,(timp+1)%cmmmc));

timp\_minim[vec][(timp+1)%cmmmc]=timp\_minim[nod][timp]+1;

}

}

}

int timp=100000000;

for(int i=0;i<cmmmc;i++){

if(timp\_minim[n-1][i]!=-1 && timp\_minim[n-1][i]<timp){

timp=timp\_minim[n-1][i];

}

}

if(timp!=100000000)

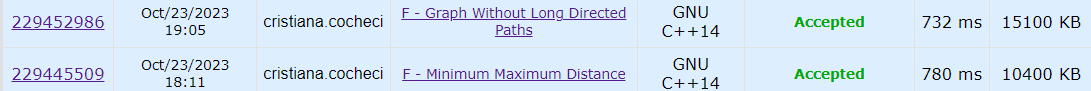
cout<<timp;

else

cout<<"-1";

return 0;

}



9.- <https://codeforces.com/contest/1881/problem/F>

https://codeforces.com/contest/1881/submission/229445509

#include <iostream>

#include<bits/stdc++.h>

using namespace std;

void bfsDistante(int nod, vector<vector<int>> &graf, vector<int> &distante){

queue<int> coada;

coada.push(nod);

distante[nod]=0;

while(!coada.empty()){

int nod=coada.front();

coada.pop();

for(int i=0;i<graf[nod].size();i++){

int vec=graf[nod][i];

if(distante[vec]==-1){

distante[vec]=distante[nod]+1;

coada.push(vec);

}

}

}

}

int main()

{

//freopen("intro.in","r",stdin);

//freopen("out.out","w",stdout);

int t;

cin>>t;

for(int q=0;q<t;q++){

int n,k;

cin>>n>>k;

vector<bool> rosii(n+1,0);

for(int i=0;i<k;i++){

int x;

cin>>x;

rosii[x]=1;

}

vector<vector<int>> graf(n+1);

for(int i=1;i<n;i++){

int x,y;

cin>>x>>y;

graf[y].push\_back(x);

graf[x].push\_back(y);

}

///facem diametrul doar cu nodurile rosii

vector<int> distante(n+1);

for(int i=1;i<=n;i++){

distante[i]=-1;

}

//bfs care returneaza cel mai departat nod de radacina aleasa

bfsDistante(1,graf,distante);

int max\_dist=-1;

int nod\_rosu\_indepartat1;

for(int i=1;i<=n;i++){

if(distante[i]>max\_dist && rosii[i]==1){

nod\_rosu\_indepartat1=i;

max\_dist=distante[i];

}

}

vector<int> distante1(n+1,-1),distante2(n+1,-1);

//bfs ca sa gasim celalalt cel mai indepartat nod

bfsDistante(nod\_rosu\_indepartat1,graf,distante1);

max\_dist=-1;

int nod\_rosu\_indepartat2;

for(int i=1;i<=n;i++){

if(distante1[i]>max\_dist && rosii[i]==1){

nod\_rosu\_indepartat2=i;

max\_dist=distante1[i];

}

}

//cout<<"\n"<<nod\_rosu\_indepartat1<<" "<<nod\_rosu\_indepartat2<<"\n";

//calculam distantele de la cele doua noduri extrema la fiecare nod

//bfsDistante(nod\_rosu\_departat1,graf,distante1);

bfsDistante(nod\_rosu\_indepartat2,graf,distante2);

int minim=1000000;

for(int i=1;i<=n;i++){

int dist\_max\_pentru\_i=max(distante1[i],distante2[i]);

if(minim>dist\_max\_pentru\_i){

minim=dist\_max\_pentru\_i;

}

}

cout<<minim<<"\n";

}

return 0;

}

10.- <https://codeforces.com/contest/1144/problem/F>

<https://codeforces.com/contest/1144/submission/229452986>

#include <iostream>

#include <bits/stdc++.h>

using namespace std;

vector<int> bipartit(vector<vector<int>> &graf) {

int n=graf.size()-1;

bool bipartit=true;

//colorare

vector<int> colorare(n+2,0);

//parcurgere

queue<int> coada;

for(int i=1;i<=n;i++){

if(colorare[i]==0){//daca nu e vizitat nodul

coada.push(i);

colorare[i]=-1;

while (!coada.empty() && bipartit){

int nod=coada.front();

coada.pop();

for(int j=0;j<graf[nod].size();j++){

int vec=graf[nod][j];

if(colorare[vec]==0){//verificam daca vecinul a fost colorat deja

colorare[vec]=-colorare[nod];

coada.push(vec);

}

else{

if(colorare[vec]==colorare[nod]){

bipartit=false;

break;

}

}

}

}

}

}

colorare[n+1]=(bipartit);

return colorare;

}

int main()

{

int n,m;

cin>>n>>m;

vector<vector<int>> graf(n+1), edges;

for(int i=0;i<m;i++){

vector<int> t(2);

cin>>t[0]>>t[1];

edges.push\_back(t);

graf[t[0]].push\_back(t[1]);

graf[t[1]].push\_back(t[0]);

}

vector<int> colorare=bipartit(graf);

if(colorare[n+1]==0){

cout<<"NO";

}

else{

cout<<"YES\n";

for(int i=0;i<m;i++){

if(colorare[edges[i][0]]==1){

cout<<0;

//rezultat.push\_back(0);

}

else{

cout<<1;

//rezultat.push\_back(1);

}

}

}

return 0;

}