حل مسئله ۳۰ وزیر با الگوریتم رقابت استعماری ICA در متلب

صورت مسئله:

در یک صفحه شطرنجی ۳۰x۳۰، می خواهیم به نحوی ۳۰ وزیر را در صفحه قرار دهیم که در هر ستون تنها یک وزیر باشد و هیچ دو وزیری یکدیگر را گارد نکنند.

شرح کد:

این سورس کد شامل ۱۲ فایل می باشد که عبارتند از:

MyCost: تابعی که گارد بودن وزیرها را بررسی می کند و هزینه آن را محاسبه می کند.

```
function [z sol]=MyCost(s)

    n=numel(s);

    [~, X]=sort(s);
    Y=1:n;

    Hit=zeros(n,n); برخورد ها

    z=0;
    for i=1:n-1
        for j=i+1:n
            if abs(X(i)-X(j))==abs(Y(i)-Y(j)) بررسی برخورد
                Hit(i,j)=1;
                Hit(j,i)=1;
                z=z+1; تعداد برخورد
            end
        end
    end

                                        دادن مقادیر به مدل
    sol.X=X;
    sol.Y=Y;
    sol.Hit=Hit;
    sol.z=z;

end
```

CreateInitialEmpires(): تشکیل امپراطوری

```
function emp=CreateInitialEmpires()
```

استفاده از متغیرهای عمومی تعریف شده در SharedStting که در هر جای کد میشه به آن دسترسی داشت.

```
    global ProblemSettings;
    global ICASettings;

    CostFunction=ProblemSettings.CostFunction;
    nVar=ProblemSettings.nVar;
    VarSize=ProblemSettings.VarSize;
    VarMin=ProblemSettings.VarMin;
    VarMax=ProblemSettings.VarMax;

    nPop=ICASettings.nPop;
    nEmp=ICASettings.nEmp;
    nCol=nPop-nEmp;
    alpha=ICASettings.alpha;

    empty_country.Position=[];
    empty_country.Cost=[];
    empty_country.Sol=[];

    country=repmat(empty_country,nPop,1);
```

یک ماتریس country متشکل از یک بردار nPopx1 از empty_country کشور ایجاد می کند.

```
    For i=1:nPop

        country(i).Position=unifrnd(VarMin,VarMax,VarSize);
```

آرایه ای از اعداد تصادفی یکنواخت به اندازه VarSize و مینیمم و ماکزیمم مشخص شده.

```
        [country(i).Cost
country(i).Sol]=CostFunction(country(i).Position);
```

هزینه هر کشور

```
    end
```

```
    costs=[country.Cost];
    [~, SortOrder]=sort(costs);
                                        مرتب سازی بر اساس هزینه
    country=country(SortOrder);

    imp=country(1:nEmp);

    col=country(nEmp+1:end);


    empty_empire.Imp=[];
    empty_empire.Col=repmat(empty_country,0,1);
    empty_empire.nCol=0;
    empty_empire.TotalCost=[];

    emp=repmat(empty_empire,nEmp,1);

                                                انتصاب امپریالیست ها
    for k=1:nEmp
        emp(k).Imp=imp(k);
    end

                                                اختصاص مستعمرات
    P=exp(-alpha*[imp.Cost]/max([imp.Cost]));
    P=P/sum(P);
    for j=1:nCol

        k=RouletteWheelSelection(P); چرخ رولت

        emp(k).Col=[emp(k).Col
                    col(j)];

        emp(k).nCol=emp(k).nCol+1;
    end

    emp=UpdateTotalCost(emp);

end
```

AssimilateColonies(emp):سیاست جذب

```
function emp=AssimilateColonies(emp)

   استفاده از متغیرهای عمومی تعریف شده در SharedStting که در
                                هر جای برنامه میشه به آن دسترسی داشت
    global ProblemSettings;
```

```
    CostFunction=ProblemSettings.CostFunction;
    VarSize=ProblemSettings.VarSize;
    VarMin=ProblemSettings.VarMin;
    VarMax=ProblemSettings.VarMax;

    global ICASettings;
    beta=ICASettings.beta;

    nEmp=numel(emp);
    for k=1:nEmp
        for i=1:emp(k).nCol

            NewPos = emp(k).Col(i).Position +
beta*rand(VarSize).*(emp(k).Imp.Position-
emp(k).Col(i).Position);
            NewPos = max(NewPos,VarMin);
            NewPos = min(NewPos,VarMax);

            emp(k).Col(i).Position = NewPos;

            emp(k).Col(i).Cost =
CostFunction(emp(k).Col(i).Position);

        end
    end

end
```

انقلاب :DoRevolution(emp)

```
function emp=DoRevolution(emp)

    global ProblemSettings;
    CostFunction=ProblemSettings.CostFunction;
    nVar=ProblemSettings.nVar;
    VarSize=ProblemSettings.VarSize;
    VarMin=ProblemSettings.VarMin;
    VarMax=ProblemSettings.VarMax;

    global ICASettings;
    pRevolution=ICASettings.pRevolution;
    mu=ICASettings.mu;

    nmu=ceil(mu*nVar);
```

```
    sigma=0.1*(VarMax-VarMin);

    nEmp=numel(emp);
    for k=1:nEmp

        NewPos = emp(k).Imp.Position +
sigma*randn(VarSize);
        NewPos = max(NewPos,VarMin);
        NewPos = min(NewPos,VarMax);

        jj=randsample(nVar,nmu)';
        NewImp=emp(k).Imp;
        NewImp.Position(jj)=NewPos(jj);
        NewImp.Cost=CostFunction(NewImp.Position);
        if NewImp.Cost<emp(k).Imp.Cost
            emp(k).Imp = NewImp;
        end

        for i=1:emp(k).nCol
            if rand<=pRevolution

                NewPos = emp(k).Col(i).Position +
sigma*randn(VarSize);
                NewPos = max(NewPos,VarMin);
                NewPos = min(NewPos,VarMax);

                jj=randsample(nVar,nmu)';
                emp(k).Col(i).Position(jj) = NewPos(jj);

                emp(k).Col(i).Cost =
CostFunction(emp(k).Col(i).Position);

            end
        end
    end

end
```

رقابت درون امپراتوری :InterEmpireCompetition(emp)

```
function emp=InterEmpireCompetition(emp)

    if numel(emp)==1
        return;
    end
```

```
global ICASettings;
alpha=ICASettings.alpha;

TotalCost=[emp.TotalCost];

[~, WeakestEmpIndex]=max(TotalCost);
WeakestEmp=emp(WeakestEmpIndex);

P=exp(-alpha*TotalCost/max(TotalCost));
P(WeakestEmpIndex)=0;
P=P/sum(P);
if any(isnan(P))
    P(isnan(P))=0;
    if all(P==0)
        P(:)=1;
    end
    P=P/sum(P);
end

if WeakestEmp.nCol>0
    [~, WeakestColIndex]=max([WeakestEmp.Col.Cost]);
    WeakestCol=WeakestEmp.Col(WeakestColIndex);

    WinnerEmpIndex=RouletteWheelSelection(P);
    WinnerEmp=emp(WinnerEmpIndex);

    WinnerEmp.Col(end+1)=WeakestCol;
    WinnerEmp.nCol=WinnerEmp.nCol+1;
    emp(WinnerEmpIndex)=WinnerEmp;

    WeakestEmp.Col(WeakestColIndex)=[];
    WeakestEmp.nCol=WeakestEmp.nCol-1;
    emp(WeakestEmpIndex)=WeakestEmp;
end

if WeakestEmp.nCol==0

    WinnerEmpIndex2=RouletteWheelSelection(P);
    WinnerEmp2=emp(WinnerEmpIndex2);

    WinnerEmp2.Col(end+1)=WeakestEmp.Imp;
    WinnerEmp2.nCol=WinnerEmp2.nCol+1;
    emp(WinnerEmpIndex2)=WinnerEmp2;
```

```
        emp(WeakestEmpIndex)=[];
    end

end
```

UpdateTotalCost(emp): هزینه کل امپراتوری ها را به روز کنید

```
function emp=UpdateTotalCost(emp)

    global ICASettings;
    zeta=ICASettings.zeta;

    nEmp=numel(emp);

    for k=1:nEmp
        if emp(k).nCol>0

emp(k).TotalCost=emp(k).Imp.Cost+zeta*mean([emp(k).Col.Cost
]);
        else
            emp(k).TotalCost=emp(k).Imp.Cost;
        end
    end

end
```

PlotSolution(sol): کشیدن بهترین راه حل

```
function PlotSolution(sol)

    X=sol.X-0.5;
    Y=sol.Y-0.5;
    Hit=sol.Hit;
    z=sol.z;

    n=numel(X);

    for i=1:n-1
        for j=i+1:n
            if Hit(i,j)==1
               plot([X(i) X(j)],[Y(i)
Y(j)],'b:','LineWidth',2);
```

```
                hold on;
            end
        end
    end

    plot(X,Y,'ko','MarkerSize',12,'MarkerFaceColor','y');

    strTitle=[num2str(n) '-Queens: '];
    if z==0
        title([strTitle 'No Hits!']);
    elseif z==1
        title([strTitle 'Just one Hit']);
    else
        title([strTitle num2str(z) ' Hit(s)']);
    end

    grid on;
    axis square;

    set(gca,'XTick',0:n);
    set(gca,'YTick',0:n);

    xlim([0 n]);
    xlim([0 n]);

    hold off;

end
```

ICA: کد رقابت استعماری

```
clc;
clear;
close all;

تعریف مسئله

nQueen=30;

CostFunction=@(s) MyCost(s);        تابع هزینه

nVar=nQueen;        تعداد متغیرهای تصمیم گیری

VarSize=[1 nVar];  اندازه ماتریس متغیرهای تصمیم گیری
```

```
VarMin=0;          محدوده پایین متغیرها
VarMax=1;          محدوده بالا متغیرها

پارامترهای ICA

MaxIt=500;          حداکثر تعداد تکرارها

nPop=200;           میزان جمعیت

nEmp=10;            تعداد امپراتوری ها / امپریالیست ها

alpha=1;            ضریب انتخاب

beta=2;             ضریب جذب

pRevolution=0.4;    احتمال انقلاب

mu=0.05;            نرخ انقلاب

zeta=0.1;        کلونی ها ضریب متوسط هزینه را دارند

ShareSettings;

مقداردهی اولیه

امپراتوری ها را ابتدایی کنید
emp=CreateInitialEmpires();

آرایه برای نگه داشتن بهترین مقادیر هزینه
BestCost=zeros(MaxIt,1);

حلقه اصلی ICA

for it=1:MaxIt

                                                    جذب
    emp=AssimilateColonies(emp);

    انقلاب
    emp=DoRevolution(emp);
```

```
                                                رقابت درون امپراطوری
    emp=IntraEmpireCompetition(emp);

                                        بروزرسانی مجموع هزینه امپراطوری ها
    emp=UpdateTotalCost(emp);

                                                رقابت بین امپراطوری
    emp=InterEmpireCompetition(emp);

       به روز رسانی بهترین راه حل که تاکنون پیدا شده است
    imp=[emp.Imp];
    [~, BestImpIndex]=min([imp.Cost]);
    BestSol=imp(BestImpIndex);

                                            بروز رسانی کم ترین هزینه
    BestCost(it)=BestSol.Cost;

                                            نمایش اطلاعات در هر تکرار
    disp(['Iteration ' num2str(it) ': Best Cost = '
num2str(BestCost(it))]);

                                                رسم بهترین راه حل
    figure(1);
    PlotSolution(BestSol.Sol);

    if BestCost(it)==0
        break;
    end

end

BestCost=BestCost(1:it);

نتایج

figure;
plot(BestCost,'LineWidth',2);
xlabel('Iteration');
ylabel('Best Cost');
```

نتایج:

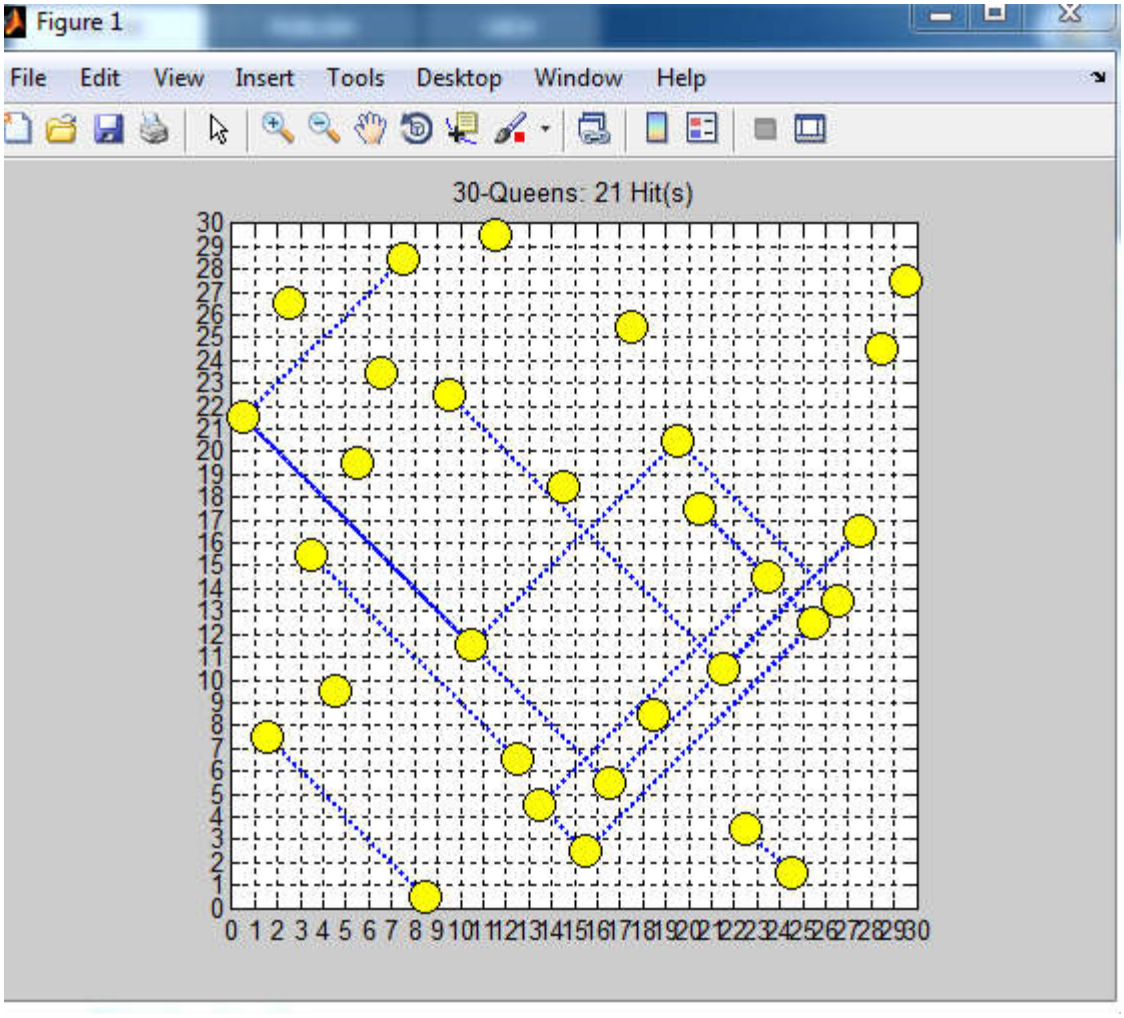
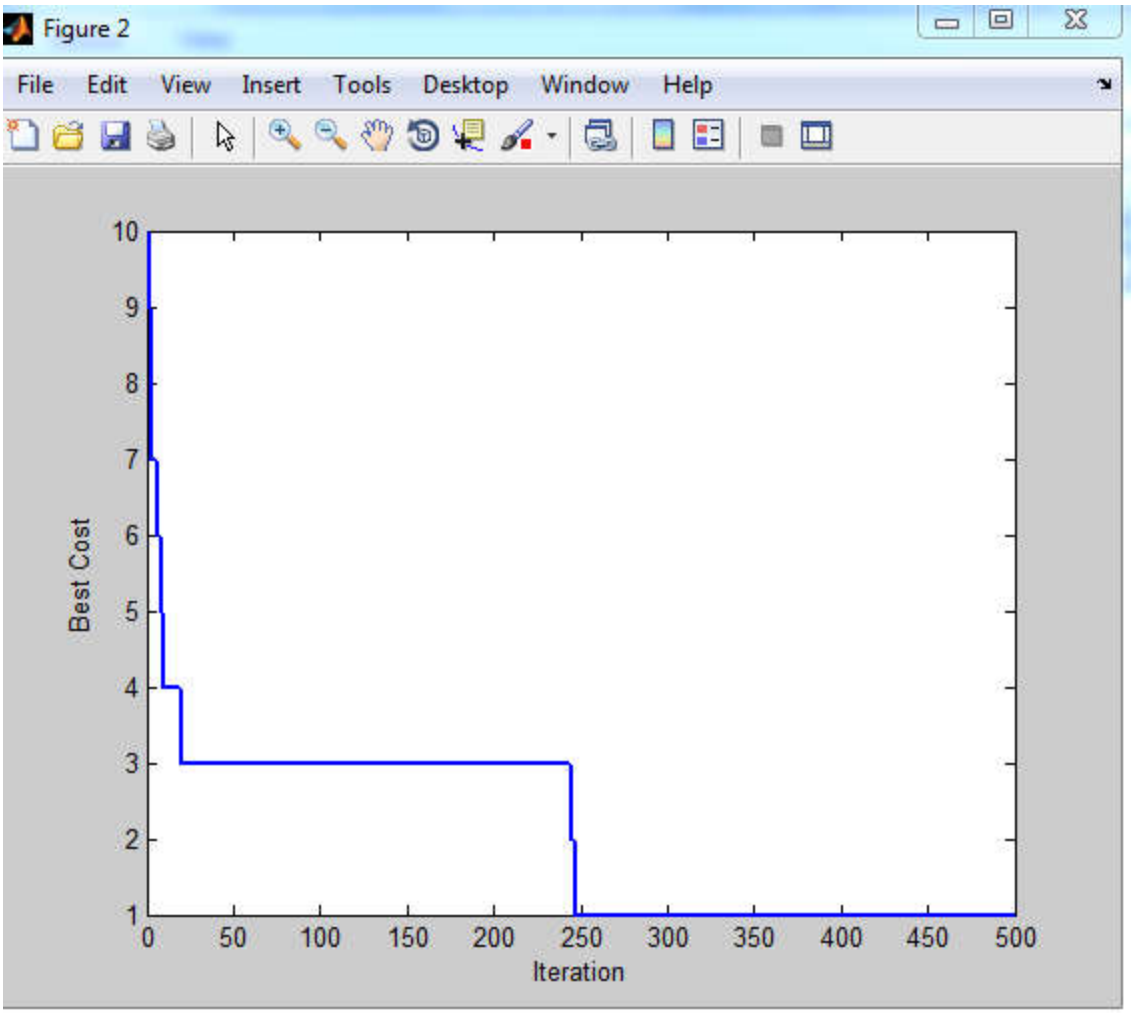


Figure 1
File Edit View Insert Tools Desktop Window Help
30-Queens: 21 Hit(s)




```
Iteration 1: Best Cost = 10

Iteration 2: Best Cost = 9

Iteration 3: Best Cost = 7

Iteration 4: Best Cost = 7

Iteration 5: Best Cost = 7

Iteration 6: Best Cost = 6

Iteration 7: Best Cost = 6

Iteration 8: Best Cost = 6

Iteration 9: Best Cost = 5
```

```
Iteration 10: Best Cost = 4
Iteration 11: Best Cost = 4
Iteration 12: Best Cost = 4
Iteration 13: Best Cost = 4
Iteration 14: Best Cost = 4
Iteration 15: Best Cost = 4
Iteration 16: Best Cost = 4
Iteration 17: Best Cost = 4
Iteration 18: Best Cost = 4
Iteration 19: Best Cost = 4
Iteration 20: Best Cost = 3
Iteration 21: Best Cost = 3
Iteration 22: Best Cost = 3
Iteration 23: Best Cost = 3
Iteration 24: Best Cost = 3
Iteration 25: Best Cost = 3
Iteration 26: Best Cost = 3
Iteration 27: Best Cost = 3
Iteration 28: Best Cost = 3
Iteration 29: Best Cost = 3
Iteration 30: Best Cost = 3
Iteration 31: Best Cost = 3
```

```
Iteration 32: Best Cost = 3
Iteration 33: Best Cost = 3
Iteration 34: Best Cost = 3
Iteration 35: Best Cost = 3
Iteration 36: Best Cost = 3
Iteration 37: Best Cost = 3
Iteration 38: Best Cost = 3
Iteration 39: Best Cost = 3
Iteration 40: Best Cost = 3
Iteration 41: Best Cost = 3
Iteration 42: Best Cost = 3
Iteration 43: Best Cost = 3
Iteration 44: Best Cost = 3
Iteration 45: Best Cost = 3
Iteration 46: Best Cost = 3
Iteration 47: Best Cost = 3
Iteration 48: Best Cost = 3
Iteration 49: Best Cost = 3
Iteration 50: Best Cost = 3
Iteration 51: Best Cost = 3
Iteration 52: Best Cost = 3
Iteration 53: Best Cost = 3
```

```
Iteration 54: Best Cost = 3
Iteration 55: Best Cost = 3
Iteration 56: Best Cost = 3
Iteration 57: Best Cost = 3
Iteration 58: Best Cost = 3
Iteration 59: Best Cost = 3
Iteration 60: Best Cost = 3
Iteration 61: Best Cost = 3
Iteration 62: Best Cost = 3
Iteration 63: Best Cost = 3
Iteration 64: Best Cost = 3
Iteration 65: Best Cost = 3
Iteration 66: Best Cost = 3
Iteration 67: Best Cost = 3
Iteration 68: Best Cost = 3
Iteration 69: Best Cost = 3
Iteration 70: Best Cost = 3
Iteration 71: Best Cost = 3
Iteration 72: Best Cost = 3
Iteration 73: Best Cost = 3
Iteration 74: Best Cost = 3
Iteration 75: Best Cost = 3
```

```
Iteration 76: Best Cost = 3
Iteration 77: Best Cost = 3
Iteration 78: Best Cost = 3
Iteration 79: Best Cost = 3
Iteration 80: Best Cost = 3
Iteration 81: Best Cost = 3
Iteration 82: Best Cost = 3
Iteration 83: Best Cost = 3
Iteration 84: Best Cost = 3
Iteration 85: Best Cost = 3
Iteration 86: Best Cost = 3
Iteration 87: Best Cost = 3
Iteration 88: Best Cost = 3
Iteration 89: Best Cost = 3
Iteration 90: Best Cost = 3
Iteration 91: Best Cost = 3
Iteration 92: Best Cost = 3
Iteration 93: Best Cost = 3
Iteration 94: Best Cost = 3
Iteration 95: Best Cost = 3
Iteration 96: Best Cost = 3
Iteration 97: Best Cost = 3
```

```
Iteration 98: Best Cost = 3
Iteration 99: Best Cost = 3
Iteration 100: Best Cost = 3
Iteration 101: Best Cost = 3
Iteration 102: Best Cost = 3
Iteration 103: Best Cost = 3
Iteration 104: Best Cost = 3
Iteration 105: Best Cost = 3
Iteration 106: Best Cost = 3
Iteration 107: Best Cost = 3
Iteration 108: Best Cost = 3
Iteration 109: Best Cost = 3
Iteration 110: Best Cost = 3
Iteration 111: Best Cost = 3
Iteration 112: Best Cost = 3
Iteration 113: Best Cost = 3
Iteration 114: Best Cost = 3
Iteration 115: Best Cost = 3
Iteration 116: Best Cost = 3
Iteration 117: Best Cost = 3
Iteration 118: Best Cost = 3
Iteration 119: Best Cost = 3
```

```
Iteration 120: Best Cost = 3
Iteration 121: Best Cost = 3
Iteration 122: Best Cost = 3
Iteration 123: Best Cost = 3
Iteration 124: Best Cost = 3
Iteration 125: Best Cost = 3
Iteration 126: Best Cost = 3
Iteration 127: Best Cost = 3
Iteration 128: Best Cost = 3
Iteration 129: Best Cost = 3
Iteration 130: Best Cost = 3
Iteration 131: Best Cost = 3
Iteration 132: Best Cost = 3
Iteration 133: Best Cost = 3
Iteration 134: Best Cost = 3
Iteration 135: Best Cost = 3
Iteration 136: Best Cost = 3
Iteration 137: Best Cost = 3
Iteration 138: Best Cost = 3
Iteration 139: Best Cost = 3
Iteration 140: Best Cost = 3
Iteration 141: Best Cost = 3
```

```
Iteration 142: Best Cost = 3
Iteration 143: Best Cost = 3
Iteration 144: Best Cost = 3
Iteration 145: Best Cost = 3
Iteration 146: Best Cost = 3
Iteration 147: Best Cost = 3
Iteration 148: Best Cost = 3
Iteration 149: Best Cost = 3
Iteration 150: Best Cost = 3
Iteration 151: Best Cost = 3
Iteration 152: Best Cost = 3
Iteration 153: Best Cost = 3
Iteration 154: Best Cost = 3
Iteration 155: Best Cost = 3
Iteration 156: Best Cost = 3
Iteration 157: Best Cost = 3
Iteration 158: Best Cost = 3
Iteration 159: Best Cost = 3
Iteration 160: Best Cost = 3
Iteration 161: Best Cost = 3
Iteration 162: Best Cost = 3
Iteration 163: Best Cost = 3
```

```
Iteration 164: Best Cost = 3
Iteration 165: Best Cost = 3
Iteration 166: Best Cost = 3
Iteration 167: Best Cost = 3
Iteration 168: Best Cost = 3
Iteration 169: Best Cost = 3
Iteration 170: Best Cost = 3
Iteration 171: Best Cost = 3
Iteration 172: Best Cost = 3
Iteration 173: Best Cost = 3
Iteration 174: Best Cost = 3
Iteration 175: Best Cost = 3
Iteration 176: Best Cost = 3
Iteration 177: Best Cost = 3
Iteration 178: Best Cost = 3
Iteration 179: Best Cost = 3
Iteration 180: Best Cost = 3
Iteration 181: Best Cost = 3
Iteration 182: Best Cost = 3
Iteration 183: Best Cost = 3
Iteration 184: Best Cost = 3
Iteration 185: Best Cost = 3
```

```
Iteration 186: Best Cost = 3
Iteration 187: Best Cost = 3
Iteration 188: Best Cost = 3
Iteration 189: Best Cost = 3
Iteration 190: Best Cost = 3
Iteration 191: Best Cost = 3
Iteration 192: Best Cost = 3
Iteration 193: Best Cost = 3
Iteration 194: Best Cost = 3
Iteration 195: Best Cost = 3
Iteration 196: Best Cost = 3
Iteration 197: Best Cost = 3
Iteration 198: Best Cost = 3
Iteration 199: Best Cost = 3
Iteration 200: Best Cost = 3
Iteration 201: Best Cost = 3
Iteration 202: Best Cost = 3
Iteration 203: Best Cost = 3
Iteration 204: Best Cost = 3
Iteration 205: Best Cost = 3
Iteration 206: Best Cost = 3
Iteration 207: Best Cost = 3
```

```
Iteration 208: Best Cost = 3
Iteration 209: Best Cost = 3
Iteration 210: Best Cost = 3
Iteration 211: Best Cost = 3
Iteration 212: Best Cost = 3
Iteration 213: Best Cost = 3
Iteration 214: Best Cost = 3
Iteration 215: Best Cost = 3
Iteration 216: Best Cost = 3
Iteration 217: Best Cost = 3
Iteration 218: Best Cost = 3
Iteration 219: Best Cost = 3
Iteration 220: Best Cost = 3
Iteration 221: Best Cost = 3
Iteration 222: Best Cost = 3
Iteration 223: Best Cost = 3
Iteration 224: Best Cost = 3
Iteration 225: Best Cost = 3
Iteration 226: Best Cost = 3
Iteration 227: Best Cost = 3
Iteration 228: Best Cost = 3
Iteration 229: Best Cost = 3
```

```
Iteration 230: Best Cost = 3
Iteration 231: Best Cost = 3
Iteration 232: Best Cost = 3
Iteration 233: Best Cost = 3
Iteration 234: Best Cost = 3
Iteration 235: Best Cost = 3
Iteration 236: Best Cost = 3
Iteration 237: Best Cost = 3
Iteration 238: Best Cost = 3
Iteration 239: Best Cost = 3
Iteration 240: Best Cost = 3
Iteration 241: Best Cost = 3
Iteration 242: Best Cost = 3
Iteration 243: Best Cost = 3
Iteration 244: Best Cost = 3
Iteration 245: Best Cost = 2
Iteration 246: Best Cost = 2
Iteration 247: Best Cost = 1
Iteration 248: Best Cost = 1
Iteration 249: Best Cost = 1
Iteration 250: Best Cost = 1
Iteration 251: Best Cost = 1
```

```
Iteration 252: Best Cost = 1
Iteration 253: Best Cost = 1
Iteration 254: Best Cost = 1
Iteration 255: Best Cost = 1
Iteration 256: Best Cost = 1
Iteration 257: Best Cost = 1
Iteration 258: Best Cost = 1
Iteration 259: Best Cost = 1
Iteration 260: Best Cost = 1
Iteration 261: Best Cost = 1
Iteration 262: Best Cost = 1
Iteration 263: Best Cost = 1
Iteration 264: Best Cost = 1
Iteration 265: Best Cost = 1
Iteration 266: Best Cost = 1
Iteration 267: Best Cost = 1
Iteration 268: Best Cost = 1
Iteration 269: Best Cost = 1
Iteration 270: Best Cost = 1
Iteration 271: Best Cost = 1
Iteration 272: Best Cost = 1
Iteration 273: Best Cost = 1
```

```
Iteration 274: Best Cost = 1
Iteration 275: Best Cost = 1
Iteration 276: Best Cost = 1
Iteration 277: Best Cost = 1
Iteration 278: Best Cost = 1
Iteration 279: Best Cost = 1
Iteration 280: Best Cost = 1
Iteration 281: Best Cost = 1
Iteration 282: Best Cost = 1
Iteration 283: Best Cost = 1
Iteration 284: Best Cost = 1
Iteration 285: Best Cost = 1
Iteration 286: Best Cost = 1
Iteration 287: Best Cost = 1
Iteration 288: Best Cost = 1
Iteration 289: Best Cost = 1
Iteration 290: Best Cost = 1
Iteration 291: Best Cost = 1
Iteration 292: Best Cost = 1
Iteration 293: Best Cost = 1
Iteration 294: Best Cost = 1
Iteration 295: Best Cost = 1
```

```
Iteration 296: Best Cost = 1
Iteration 297: Best Cost = 1
Iteration 298: Best Cost = 1
Iteration 299: Best Cost = 1
Iteration 300: Best Cost = 1
Iteration 301: Best Cost = 1
Iteration 302: Best Cost = 1
Iteration 303: Best Cost = 1
Iteration 304: Best Cost = 1
Iteration 305: Best Cost = 1
Iteration 306: Best Cost = 1
Iteration 307: Best Cost = 1
Iteration 308: Best Cost = 1
Iteration 309: Best Cost = 1
Iteration 310: Best Cost = 1
Iteration 311: Best Cost = 1
Iteration 312: Best Cost = 1
Iteration 313: Best Cost = 1
Iteration 314: Best Cost = 1
Iteration 315: Best Cost = 1
Iteration 316: Best Cost = 1
Iteration 317: Best Cost = 1
```

```
Iteration 318: Best Cost = 1
Iteration 319: Best Cost = 1
Iteration 320: Best Cost = 1
Iteration 321: Best Cost = 1
Iteration 322: Best Cost = 1
Iteration 323: Best Cost = 1
Iteration 324: Best Cost = 1
Iteration 325: Best Cost = 1
Iteration 326: Best Cost = 1
Iteration 327: Best Cost = 1
Iteration 328: Best Cost = 1
Iteration 329: Best Cost = 1
Iteration 330: Best Cost = 1
Iteration 331: Best Cost = 1
Iteration 332: Best Cost = 1
Iteration 333: Best Cost = 1
Iteration 334: Best Cost = 1
Iteration 335: Best Cost = 1
Iteration 336: Best Cost = 1
Iteration 337: Best Cost = 1
Iteration 338: Best Cost = 1
Iteration 339: Best Cost = 1
```

```
Iteration 340: Best Cost = 1
Iteration 341: Best Cost = 1
Iteration 342: Best Cost = 1
Iteration 343: Best Cost = 1
Iteration 344: Best Cost = 1
Iteration 345: Best Cost = 1
Iteration 346: Best Cost = 1
Iteration 347: Best Cost = 1
Iteration 348: Best Cost = 1
Iteration 349: Best Cost = 1
Iteration 350: Best Cost = 1
Iteration 351: Best Cost = 1
Iteration 352: Best Cost = 1
Iteration 353: Best Cost = 1
Iteration 354: Best Cost = 1
Iteration 355: Best Cost = 1
Iteration 356: Best Cost = 1
Iteration 357: Best Cost = 1
Iteration 358: Best Cost = 1
Iteration 359: Best Cost = 1
Iteration 360: Best Cost = 1
Iteration 361: Best Cost = 1
```

```
Iteration 362: Best Cost = 1
Iteration 363: Best Cost = 1
Iteration 364: Best Cost = 1
Iteration 365: Best Cost = 1
Iteration 366: Best Cost = 1
Iteration 367: Best Cost = 1
Iteration 368: Best Cost = 1
Iteration 369: Best Cost = 1
Iteration 370: Best Cost = 1
Iteration 371: Best Cost = 1
Iteration 372: Best Cost = 1
Iteration 373: Best Cost = 1
Iteration 374: Best Cost = 1
Iteration 375: Best Cost = 1
Iteration 376: Best Cost = 1
Iteration 377: Best Cost = 1
Iteration 378: Best Cost = 1
Iteration 379: Best Cost = 1
Iteration 380: Best Cost = 1
Iteration 381: Best Cost = 1
Iteration 382: Best Cost = 1
Iteration 383: Best Cost = 1
```

```
Iteration 384: Best Cost = 1
Iteration 385: Best Cost = 1
Iteration 386: Best Cost = 1
Iteration 387: Best Cost = 1
Iteration 388: Best Cost = 1
Iteration 389: Best Cost = 1
Iteration 390: Best Cost = 1
Iteration 391: Best Cost = 1
Iteration 392: Best Cost = 1
Iteration 393: Best Cost = 1
Iteration 394: Best Cost = 1
Iteration 395: Best Cost = 1
Iteration 396: Best Cost = 1
Iteration 397: Best Cost = 1
Iteration 398: Best Cost = 1
Iteration 399: Best Cost = 1
Iteration 400: Best Cost = 1
Iteration 401: Best Cost = 1
Iteration 402: Best Cost = 1
Iteration 403: Best Cost = 1
Iteration 404: Best Cost = 1
Iteration 405: Best Cost = 1
```

```
Iteration 406: Best Cost = 1
Iteration 407: Best Cost = 1
Iteration 408: Best Cost = 1
Iteration 409: Best Cost = 1
Iteration 410: Best Cost = 1
Iteration 411: Best Cost = 1
Iteration 412: Best Cost = 1
Iteration 413: Best Cost = 1
Iteration 414: Best Cost = 1
Iteration 415: Best Cost = 1
Iteration 416: Best Cost = 1
Iteration 417: Best Cost = 1
Iteration 418: Best Cost = 1
Iteration 419: Best Cost = 1
Iteration 420: Best Cost = 1
Iteration 421: Best Cost = 1
Iteration 422: Best Cost = 1
Iteration 423: Best Cost = 1
Iteration 424: Best Cost = 1
Iteration 425: Best Cost = 1
Iteration 426: Best Cost = 1
Iteration 427: Best Cost = 1
```

```
Iteration 428: Best Cost = 1
Iteration 429: Best Cost = 1
Iteration 430: Best Cost = 1
Iteration 431: Best Cost = 1
Iteration 432: Best Cost = 1
Iteration 433: Best Cost = 1
Iteration 434: Best Cost = 1
Iteration 435: Best Cost = 1
Iteration 436: Best Cost = 1
Iteration 437: Best Cost = 1
Iteration 438: Best Cost = 1
Iteration 439: Best Cost = 1
Iteration 440: Best Cost = 1
Iteration 441: Best Cost = 1
Iteration 442: Best Cost = 1
Iteration 443: Best Cost = 1
Iteration 444: Best Cost = 1
Iteration 445: Best Cost = 1
Iteration 446: Best Cost = 1
Iteration 447: Best Cost = 1
Iteration 448: Best Cost = 1
Iteration 449: Best Cost = 1
```

```
Iteration 450: Best Cost = 1
Iteration 451: Best Cost = 1
Iteration 452: Best Cost = 1
Iteration 453: Best Cost = 1
Iteration 454: Best Cost = 1
Iteration 455: Best Cost = 1
Iteration 456: Best Cost = 1
Iteration 457: Best Cost = 1
Iteration 458: Best Cost = 1
Iteration 459: Best Cost = 1
Iteration 460: Best Cost = 1
Iteration 461: Best Cost = 1
Iteration 462: Best Cost = 1
Iteration 463: Best Cost = 1
Iteration 464: Best Cost = 1
Iteration 465: Best Cost = 1
Iteration 466: Best Cost = 1
Iteration 467: Best Cost = 1
Iteration 468: Best Cost = 1
Iteration 469: Best Cost = 1
Iteration 470: Best Cost = 1
Iteration 471: Best Cost = 1
```

```
Iteration 472: Best Cost = 1
Iteration 473: Best Cost = 1
Iteration 474: Best Cost = 1
Iteration 475: Best Cost = 1
Iteration 476: Best Cost = 1
Iteration 477: Best Cost = 1
Iteration 478: Best Cost = 1
Iteration 479: Best Cost = 1
Iteration 480: Best Cost = 1
Iteration 481: Best Cost = 1
Iteration 482: Best Cost = 1
Iteration 483: Best Cost = 1
Iteration 484: Best Cost = 1
Iteration 485: Best Cost = 1
Iteration 486: Best Cost = 1
Iteration 487: Best Cost = 1
Iteration 488: Best Cost = 1
Iteration 489: Best Cost = 1
Iteration 490: Best Cost = 1
Iteration 491: Best Cost = 1
Iteration 492: Best Cost = 1
Iteration 493: Best Cost = 1
```

```
Iteration 494: Best Cost = 1

Iteration 495: Best Cost = 1

Iteration 496: Best Cost = 1

Iteration 497: Best Cost = 1

Iteration 498: Best Cost = 1

Iteration 499: Best Cost = 1

Iteration 500: Best Cost = 1

>>

>> BestSol.Sol


ans =


      X: [9 25 16 23 14 17 13 2 19 5 22 11 26 27 24 4
28 21 15 6 20 1 10 7 29 18 3 30 8 12]

      Y: [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
19 20 21 22 23 24 25 26 27 28 29 30]

    Hit: [30x30 double]

      z: 21


>>
```