Metoda DEI în cazul problemelor ce necesită descompunerea în 4 subprobleme

Se dă o bucată de tablă de formă dreptunghiulară cu lungimea l şi înălţimea h, având pe suprafaţa ei n găuri de coordonate numere întregi. Se cere să se decupeze din ea o bucată de arie maximă care nu prezintă găuri. Sunt permise numai tăieturi orizontale şi verticale.

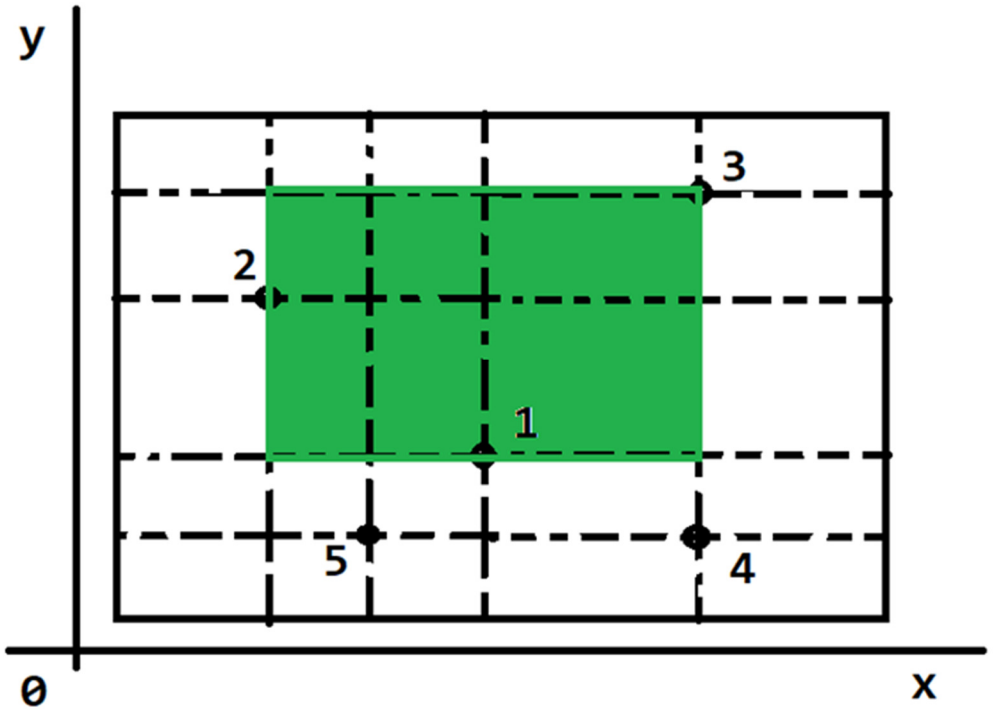

Fig. 1

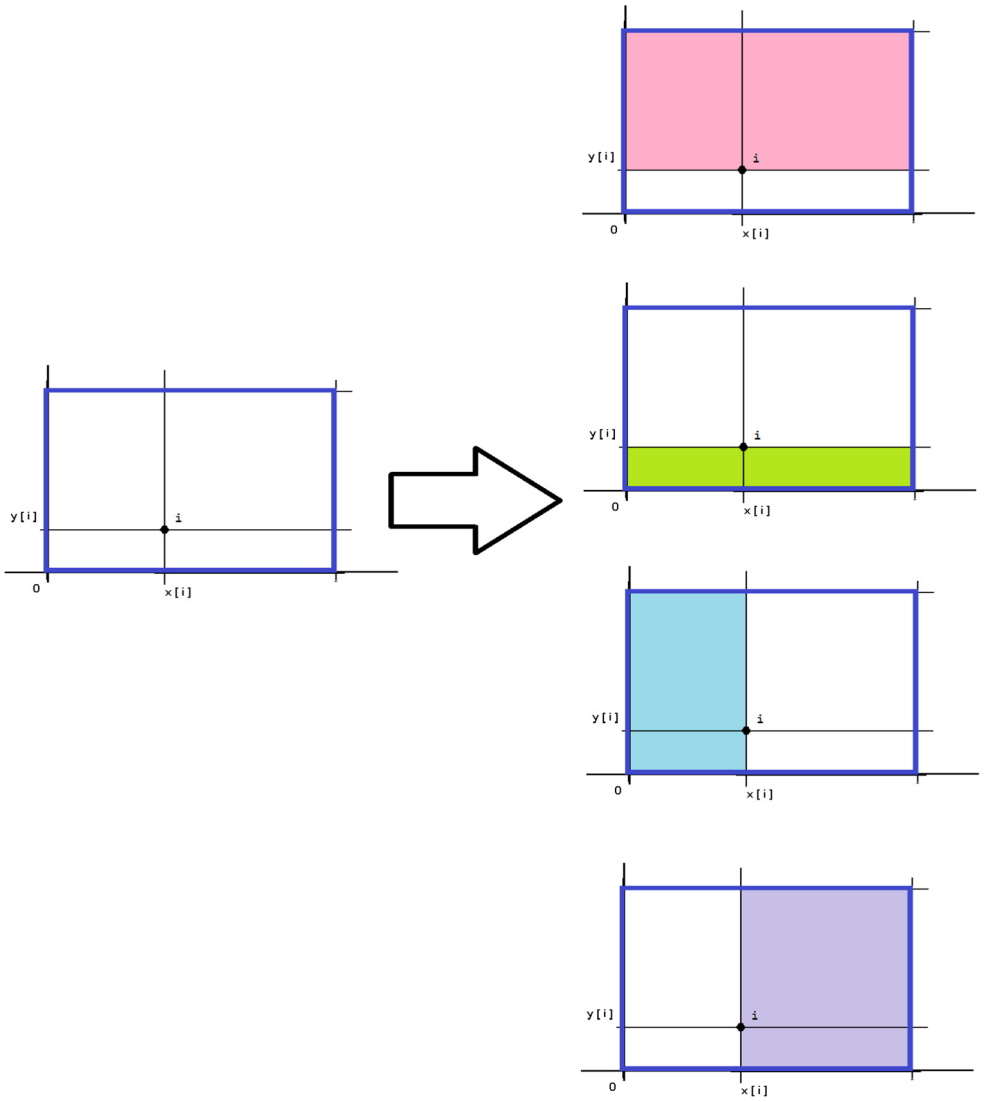

Fig. 2

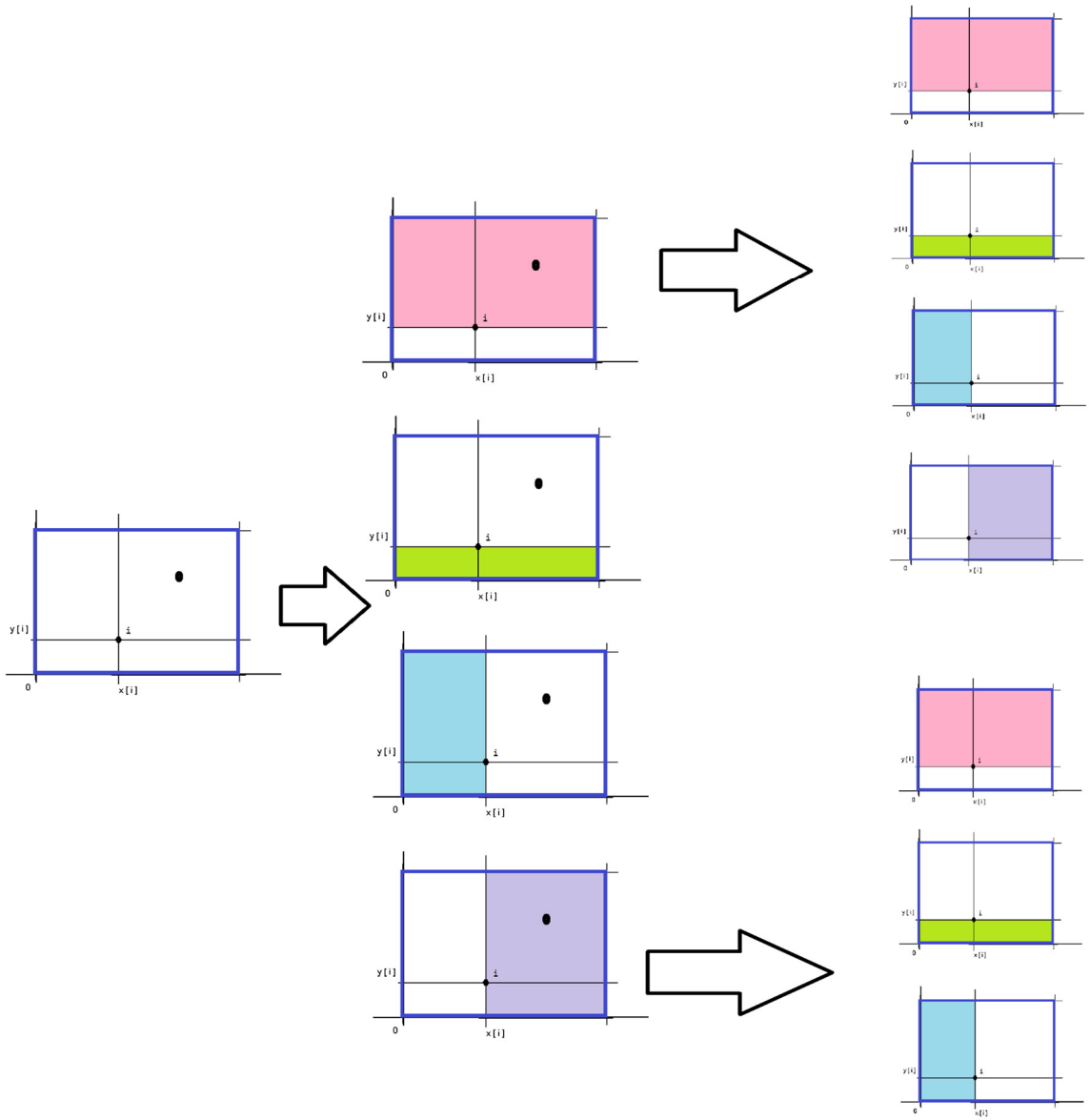

Fig. 3

```
#include<iostream>
using namespace std;
int xv[20], yv[20];
int L,H,n,xf,yf,lf,hf;
void citire()
{   int i;
      cout<<"numar gauri=\n"; cin>>n;
      cout<<" Introduceti coordonatele gaurilor (numere intregi)\n";
      for(i=1; i<=n; i++)
          { cout<<"x,y="; cin>>xv[i]>>yv[i]; }
      cout<<"lungimea tablei="; cin>>L;
      cout<<"latimea  tablei="; cin>>H;
}
void arie(int x,int y,int L,int H,int& xf,int& yf,int& lf,int& hf)
{  int gasit=0,i=1;
   while(i<=n && !gasit)
      if(xv[i]>x && xv[i]<x+l && yv[i]>y && yv[i]<y+h)
         gasit=1;
      else i++;
   if(gasit)
   {
     arie(x,y,xv[i]-x,h,xf,yf,lf,hf);
      arie(xv[i],y,l+x-xv[i],h,xf,yf,lf,hf);
      arie(x,y,l,yv[i]-y,xf,yf,lf,hf);
        arie(x,yv[i],l,h+y-yv[i],xf,yf,lf,hf);
   }
   else  // bucata nu are gauri
      if(L*H>lf*hf) {  xf=x; yf=y; lf=L; hf=H; }
}
void main()
{
   citire();
   lf=0; hf=0;
   arie(0,0,L,H,xf,yf,lf,hf);
   cout<<"\n bucata de arie maxima :  ";
   cout<<"x=”<<xf<<, y=”<<yf<<”, l=”<<lf<<”, h=”<<hf;
}
```

Fișă de lucru:
#1972
#3165

#1972. Gigel are o grădina sub forma unei matrice binare de ordin N, unde 0 reprezintă teren liber, 1 reprezintă pomi. El dorește să construiască un hambar pe dreptunghiul de arie maximă format doar din 0. Ajutați-l pe Gigel să găsească dreptunghiul de arie maximă format doar din 0. Fișierul de intrare hambar.in conține pe prima linie numerele N și M, reprezentând dimensinunea matricei, respectiv numărul de pomi, iar pe următoarele M linii se vor găsi perechi numere x și y, separate printr-un spațiu, reprezentând indicele liniei, respectiv al coloanei pe care se află un pom. Fișierul de ieșire hambar.out va conține pe prima linie numărul S, reprezentând aria maximă a unei suprafețe dreptunghiulare ce nu conține 1.
Precizări: 1 ≤ N, M ≤ 1000 , Nu se vor afla 2 sau mai mulți pomi în același loc.
Exemplu
`hambar.in`

```
5 5
1 3
2 1
2 5
5 1
5 5
```

`hambar.out`

```
12
```

```
#include <iostream>
#include <fstream>
using namespace std;
struct pom
{
  short int l;
  short int c;
} v[1001];
short int n,m;
int amax=0,k=1;

short int cauta(short int l1, short int c1, short int l2, short int c2, short int k)
{
  while(k<=m)
  {
   if(v[k].l>=l1 and v[k].l<=l2 and v[k].c>=c1 and v[k].c<=c2)
        return k;
   else k++;
  }
  return 0;
}

void hambar(short int l1, short int c1, short int l2, short int c2, short int p)
{
   int a=(l2-l1+1)*(c2-c1+1);
   if(amax<a)
   {
     p=cauta( l1, c1, l2, c2, p);
     if(p==0) amax=max(amax, a);
     else
     {
       hambar(l1, c1, l2, v[p].c-1,p+1);
       hambar(l1, v[p].c+1, l2, c2,p+1);
       hambar(l1, c1, v[p].l-1, c2,p+1);
       hambar(v[p].l+1, c1, l2, c2,p+1);
     }
  }
}
```

```
int main()
{
    ifstream f("hambar.in");
    ofstream g("hambar.out");
    f>>n>>m;
    for(int i=1;i<=m;i++)
        f>>v[i].l>>v[i].c;
    hambar(1,1,n,n,1);
    g<<amax;
    f.close();
    g.close();
    return 0;
}
```

#3165 Se consideră o matrice cu `m` linii și `n` coloane, numere naturale. Folosind metoda Divide et Impera, determinați suma numerelor pare din matrice. Programul citește de la tastatură numerele `m` și `n`, iar apoi cele `m * n` elemente ale matricei. Programul va afișa pe ecran numărul `S`, reprezentând suma numerelor pare din matrice. Precizări : `1 ≤ m, n ≤ 100,` numerele din matrice vor fi mai mici decât `1.000.000 .`

Exemplu

Intrare

```
3 5
1 2 3 4 5
2 4 6 1 7
1 3 4 5 9
```

Ieșire 22

```
#include <iostream>
using namespace std;
int a[101][101], m, n;
long long deiSum(int l1,int c1,int l2,int c2)
{
 if (l2<l1 || c2<c1) return 0;
 if (l1==l2 && c1==c2)
  { if (a[l1][c1]%2==0)  return a[l1][c1];
    else  return 0;}
 else
 { int lm=(l1+l2)/2, cm=(c1+c2)/2;
   return deiSum(l1,c1,lm,cm)+deiSum(l1,cm+1,lm,c2)+
        deiSum(lm+1,c1,l2,cm)+deiSum(lm+1,cm+1,l2,c2);  }  }
```

```
int main()
{
    cin>>m>>n;
    for (int i=1;i<=m;i++)
        for (int j=1;j<=n;j++)
            cin>>a[i][j];
    cout<<sum(1,1,m,n);
    return 0;
}
```