Practica 08

Chullunquía Rosas, Sharon Rossely

Universidad Nacional de San Agustín de Arequipa
20 de julio de 2020

1. Enunciado

Dada la siguiente matriz de tamaño nxn (0<n<128)

4
-2 0 8 -1
1 -4 1 -4
2 -6 2 9
0 -2 -7 0

(Los valores dentro de la matriz están entre 0<=valor absoluto de n<1000)

Encontrar la submatriz cuya suma es máxima.
En este caso, es:
8 -1
1 -4
2 9
Sumando 15

1. Indicar cómo sería un algoritmo divide y conquista
2. Resolver el ejercicio utilizando Programación Dinámica (SEGÚN LAS INDICACIONES EN CLASE).

PS: probar con el archivo adjunto prueba.txt

1. Indicar cómo sería un algoritmo divide y conquista

Para hallar la suma máxima de una submatriz de la matriz nxn,lo que se haria es calculando la suma acumulada de todas las submatrices que contiene la matriz; para eso necesitamos hacer uso del algoritmo divide y conquista donde una suma acumulada puede dividirse en subsumas acumuladas y asi poder minimizar el problema.
El algoritmo de divide y conquista se aplica en la funcion sumAcum que calcula la suma acumulada hasta el índice i, j (donde i es fila y j es columna) y también se aplica en la función de sumaMatriz que calcula la suma acumulada a partir de limites inferiores y superiores.
En la función de la suma acumulada lo que se hace es tener un caso base donde la suma acumulada en la posición [0,0] es el mismo número que esta en esa posición.Luego se presentan tres casos ,el primer caso se da cuando i=0 y j!=0; para este primer caso se halla la suma entre el número que se encuentra en la posicion A[i,j] de la matriz original y la suma acumulada de A[i,j-1] obteniendo así la suma acumulada de A[i,j], en el segundo caso es cuando i!=0 y j=0,este caso es parecido al anterior solo que al sentido inverso,donde la suma acumulada se halla sumando el

número que se encuentra en la posición A[i,j] de la matriz original mas la suma acumulada de A[i-1,j];en el tercer caso es cuando i y j toman cualquier valor diferente de cero, donde se calcula la suma acumulada sumando el valor que esta en la posicion A[i,j] de la matriz original más la suma acumulada de la posicion una fila arriba A[i1,j], más la otra suma acumulada que es una columna más a la izquierdaA[i,j-1], restando la suma acumulada de la submatriz que se repite en la diagonal A[i-1,j-1]. En cuanto a la función sumaMatriz, es la función principal que hace uso de la funcion suma acumulada donde existen 4 casos,esta funcion calcula la suma de las submatrices; el primer caso esta dado cuando los limites inferiores son igual a [0,0] en ese caso simplemente se calcula la suma acumulada de los limites superiores A[fSup,cSup],siendo este la suma de la submatriz; en el segundo caso es cuando la columna inferior es 0 para lo cual hacemos una resta entre la suma acumulada de los limites superiores menos la suma acumulada de la posicion una fila menos a la de la fila superior (A[fInf-1,cSup]);en el tercer caso es cuando la fila inferior es 0,este caso es parecido al segundo pero a la inversa calculandose la suma de la submatriz a partir de la suma acumulada de A[fSup,cSup] menos la suma acumulada de A[fSup,cInf-1],para el último caso es cuando los limites inferiores son diferentes de cero es decir es una submatriz que se encuentra en el medio de la matriz y no en los bordes como los anteriores casos, la suma de esta submatriz se calcula con la regla presentada en clase, z - p - q + r, donde esta representado por suma acumulada de A[fSup,cSup] - suma acumulada de A[fInf-1,cSup] - suma acumulada de A [fSup,cInf-1] + suma acumulada de A[fInf-1,cInf-1].

Código

```
1   //Nombre : Sharon Chullunqu a Rosas
2
3   //Codigo hecho con Divide y Conquista
4
5   #include <iostream>
6
7   using namespace std;
8
9   int sumAcum(int **A,int i,int j){
10      //Caso base
11      if (i==0 && i==j){
12          return A[i][j];
13      }
14      //Primer caso
15      if (i==0 && j!=0){
16          return(sumAcum(A,i,j-1)+A[i][j]);
17      }
18      //Segundo caso
19      if (i!=0 && j==0){
20          return(sumAcum(A,i-1,j)+A[i][j]);
21      }
22      //Tercer caso
23      if (i!=0 && j !=0){
24          //A[i,j] = sumatoria de los elementos hasta el indice i,j
25          //         = Orig[i,j] + A[i,j-1] + A[i-1,j] - A[i-1,j-1]
26          return (A[i][j] + sumAcum(A,i,j-1)+ sumAcum(A,i-1,j)-
                sumAcum(A,i-1,j-1));
```

```
27          }
28      }
29
30      /*
31
32      Donde :
33
34      fInf : fila inferior
35      cInf : columna inferior
36      fSup : fila superior
37      cSup : columna superior
38
39      */
40
41      int sumaMatriz(int **A,int fInf,int cInf,int fSup ,int  cSup){
42          int resultado;
43          //Primer caso
44          if(fInf==0 && fInf==cInf){
45              resultado = sumAcum(A,fSup,cSup);
46              return resultado;
47          }
48          //Segundo Caso
49          if (cInf == 0 && fInf != 0){
50              resultado = sumAcum(A,fSup,cSup) - sumAcum(A,fInf-1,cSup);
51              return resultado;
52          }
53          //Tercer Caso
54          if (cInf != 0 && fInf ==0){
55              resultado = sumAcum(A,fSup,cSup) - sumAcum(A,fSup,cInf-1);
56              return resultado;
57          }
58          //Cuarto Caso
59          if (cInf != 0 & fInf != 0){
60              //resultado = z - p - q + r
61              resultado = sumAcum(A,fSup,cSup) - sumAcum(A,fInf-1,cSup) -
                    sumAcum(A,fSup,cInf-1) + sumAcum(A,fInf-1,cInf-1);
62              return resultado;
63          }
64      }
65
66      int suma_maxima(int **A,int n){
67          int resultado;
68          int sumMax = -999*(n**2); //el valor mas pequenio que puede
                tomar
69          for(int fInf = 0; fInf < n; fInf++){
70              for(int cInf = 0; cInf < n; cInf++){
71                  for(int fSup = fInf; fSup < n; fSup++){
72                      for(int cSup = cInf; cSup < n; cSup++){
```

```
73                          resultado = sumaMatriz(A, fInf, cInf, fSup,
                                cSup);
74                          if( sumMax < resultado)
75                              sumMax = resultado;
76                      }
77                  }
78              }
79          }
80          return sumMax;
81      }
82
83
84      int main (){
85          int n;
86          bool flag = true;
87          cin>>n;
88          if (0<n && n<128){
89
90              int **A = new int* [n];
91              for (int i = 0; i < n; i++){
92                  A[i] = new int[n];
93              }
94
95              for(int i=0;i<n;i++){
96                  for(int j=0;j<n;j++){
97                      cin>>A[i][j];
98                      //Los valores dentro de la matriz est n entre 0<=|
                            n|<1000
99                      if(A[i][j] >= 1000 || A[i][j] <= -1000){
100                         flag = false;
101                         break;
102                     }
103                 }
104                 if(flag == false)
105                     break;
106             }
107
108             if(flag == true){
109                 int resultado;
110                 resultado = suma_maxima(A,n);
111                 cout<<resultado<<endl;
112             }
113             else{
114                 cout<<"Valores fuera del rango establecido 0<=|n|<1000"
                        <<endl;
115             }
116
117             for(int i = 0; i < n; i++){
118                 delete[] A[i];
```

```
119             }
120             delete[] A;
121         }
122         return 0;
123     }
124 
125     /*
126     Para compilar :
127 
128     g++ suma_max_matriz.cpp -o suma_max_matriz.out
129     ./suma_max_matriz.out < prueba.txt > resultado.txt
130     */
```

2. Resolver el ejercicio utilizando Programación Dinámica (SEGÚN LAS INDICACIONES EN CLASE).

Código

```
1     //Nombre : Sharon Chullunqu a Rosas
2 
3     //Codigo hecho con programaci n dinamica
4 
5     #include <iostream>
6     #include <cmath>
7 
8     using namespace std;
9 
10    /*
11 
12    Donde :
13 
14    fInf : fila inferior
15    cInf : columna inferior
16    fSup : fila superior
17    cSup : columna superior
18 
19    */
20 
21    int suma_maxima(int **A,int n){
22        int Subsuma_resultado;
23        int sumMax = -999*pow(n,2); //el valor mas pequenio que puede
              tomar
24        for(int fInf = 0; fInf < n; fInf++){
25            for(int cInf = 0; cInf < n; cInf++){
26                for(int fSup = fInf; fSup < n; fSup++){
27                    for(int cSup = cInf; cSup < n; cSup++){
28                        Subsuma_resultado = A[fSup][cSup];// O(1)
29                        if(fInf > 0)
```

```
30                              Subsuma_resultado -= A[fInf-1][cSup];
31                          if(cInf > 0)
32                              Subsuma_resultado -= A[fSup][cInf-1];
33                          if(fInf > 0 && cInf > 0)
34                              Subsuma_resultado += A[fInf-1][cInf-1];
35                          sumMax = max(sumMax, Subsuma_resultado);
36                      }
37                  }
38              }
39          }
40          return sumMax;
41      }
42
43
44      int main (){
45          int n;
46          bool flag = true;
47          cin>>n;
48          if (0<n && n<128){
49
50              int **A = new int* [n];
51              for (int i = 0; i < n; i++){
52                  A[i] = new int[n];
53              }
54
55              for(int i=0;i<n;i++){
56                  for(int j=0;j<n;j++){
57                      cin>>A[i][j];
58                      //Los valores dentro de la matriz est n entre 0<=|
                            n|<1000
59                      if(A[i][j] >= 1000 || A[i][j] <= -1000){
60                          flag = false;
61                          break;
62                      }
63                  }
64                  if(flag == false)
65                      break;
66              }
67
68              if(flag == true){
69                  int resultado;
70                  //Hallando matriz acumulativa  sobre la matriz original
71                  for (int i = 0; i < n; ++i) {
72                      for (int j = 0; j < n; ++j) {
73                          if (i > 0)
74                              A[i][j] += A[i - 1][j];
75                          if (j > 0)
76                              A[i][j] += A[i][j - 1];
77                          if (i > 0 && j > 0)
```

```
78                              A[i][j] -= A[i - 1][j - 1];
79                          }
80                      }
81                      resultado = suma_maxima(A,n);
82                      cout<<resultado<<endl;
83                  }
84                  else
85                      cout<<"Valores fuera del rango establecido 0<=|n|<1000"
                            <<endl;
86
87                  for(int i = 0; i < n; i++){
88                      delete[] A[i];
89                  }
90                  delete[] A;
91              }
92              return 0;
93          }
94
95          /*
96          Para compilar :
97
98          g++ suma_max_matriz_pd.cpp -o suma_max_matriz_pd.out
99          ./suma_max_matriz_pd.out < prueba1.txt > resultado1.txt
100         */
```

Bibliografía

[1] Código del ejercicio con Programación Dinámica en GitHub : `https://github.com/sharon1160/ADA/blob/master/Practica08/suma_max_matriz_pd.cpp`

[2] Código del ejercicio con Divide y Conquista en GitHub : `https://github.com/sharon1160/ADA/blob/master/Practica08/suma_max_matriz.cpp`