Taller de Programación

www.inf.ucv.cl/ wpalma/taller

Dr. Wenceslao Palma
wenceslao.palma@ucv.cl

Maximum Sum

Given a $n \times n$ array of positive and negative integers, find the sub-rectangle with the largest sum. The sum of a rectangle is the sum of all the elements in that rectangle. In this problem the sub-rectangle with the largest sum is referred to as the maximal sub-rectangle. A sub-rectangle is any contiguous sub-array of size 1×1 or greater located within the whole array.

La solución naive es $O(n^6)$.

```
for(i=0;i<n;i++) for(j=0;j<n;j++)
   for(k=i;k<n;k++) for(l=j;l<n;l++){
      subRectangulo = 0;
      for(a=i;a<=k;a++) for(b=j;b<=l;b++)
         subrectangulo += A[a][b];
      maxSubrectangulo = max(maxSubrectangulo,subRectangulo);
   }
```

$$\begin{bmatrix} 0 & -2 & -7 & 0 \\ 9 & 2 & -6 & 2 \\ -4 & 1 & -4 & 1 \\ -1 & 8 & 0 & -2 \end{bmatrix} \Longrightarrow \begin{bmatrix} 9 & 2 \\ -4 & 1 \\ -1 & 8 \end{bmatrix} = 15$$

Idea

Transformar el arreglo original A en un arreglo donde $A[i][j]$ contiene la suma de todos los valores ubicados dentro del rectángulo que va desde $(0,0)$ hasta (i,j). Dicho proceso toma $O(n^2)$.

$$\begin{bmatrix} 0 & -2 & -7 & 0 \\ 9 & 2 & -6 & 2 \\ -4 & 1 & -4 & 1 \\ -1 & 8 & 0 & -2 \end{bmatrix} \Longrightarrow \begin{bmatrix} 0 & -2 & -9 & -9 \\ 9 & 9 & -4 & -2 \\ 5 & 6 & -11 & -8 \\ 4 & 13 & -4 & -3 \end{bmatrix}$$

La suma del subrectángulo que va desde $(2,2)$ hasta $(3,3)$ se puede calcular como: $A[3][3] - A[1][3] - A[3][1] + A[1][1] = -3 - (-2) - 13 + 9 = -5$.

En general cualquier subrectángulo que va desde (i,j) hasta (k,l) se puede calcular en $O(1)$ y el problema puede ser resuelto en $O(n^4)$:-)

Programación dinámica

```
for(i=0;i<n;i++) for(j=0;j<n;j++)
   for(k=i;k<n;k++) for(l=j;l<n;l++){
      subRectangulo = A[k][l];
      if (i>0) subRectangulo -= A[i-1][l]; // O(1) :-)
      if (j>0) subRectangulo -= A[k][j-1]; // O(1) :-)
      if (i>0 && j>0) subRectangulo += A[i-1][j-1];
      maxSubrectangulo = max(maxSubrectangulo,subRectangulo);
   }
```

The Longest Increasing Subsequence Problem (LIS)

Given a string $X =< x_1, x_2, ..., x_n >$ of n characters drawn from a totally ordered alphabet, an Increasing Subsequence (IS) of X is a subsequence $Z =< z_1, z_2, ..., z_k >$ of X (that is, $Z =< x_{i_1}, x_{i_2}, ..., x_{i_k} >$, with $1 \leq i_1 < i_2 ... < i_k \leq n$) with $z_i < z_{i+1}$ for $1 \leq i < k$. The goal is to design an algorithm that returns a Longest Increasing Subsequence (LIS) of the input string X. For instance, the longest increasing subsequence of $5, 2, 8, 6, 3, 6, 9, 7$ is $2, 3, 6, 9$.

Un enfoque basado en fuerza bruta debe generar todos los subconjuntos posibles de X lo cual es $O(2^n)$. Necesitamos un enfoque más eficiente para calcular $LIS(X)$.

Programación dinámica