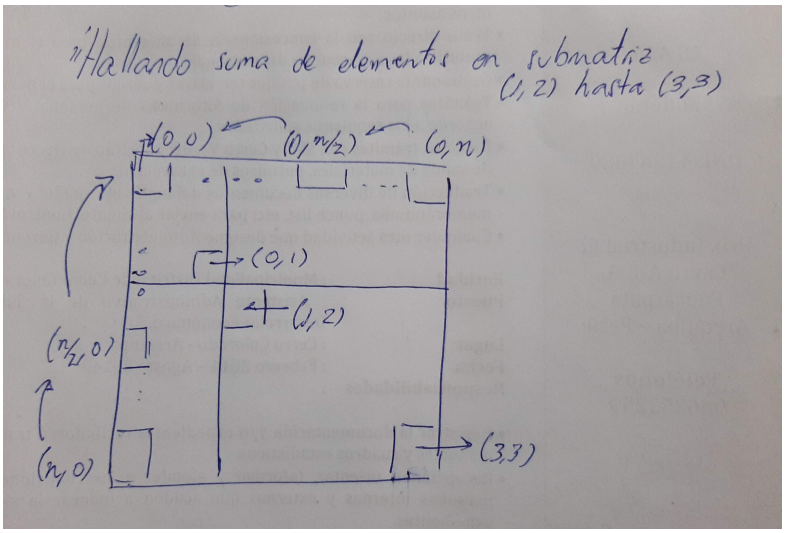
Jerson Zúñiga Coayla

20 de julio de 2020

1. DIVIDE Y VENCERÁS

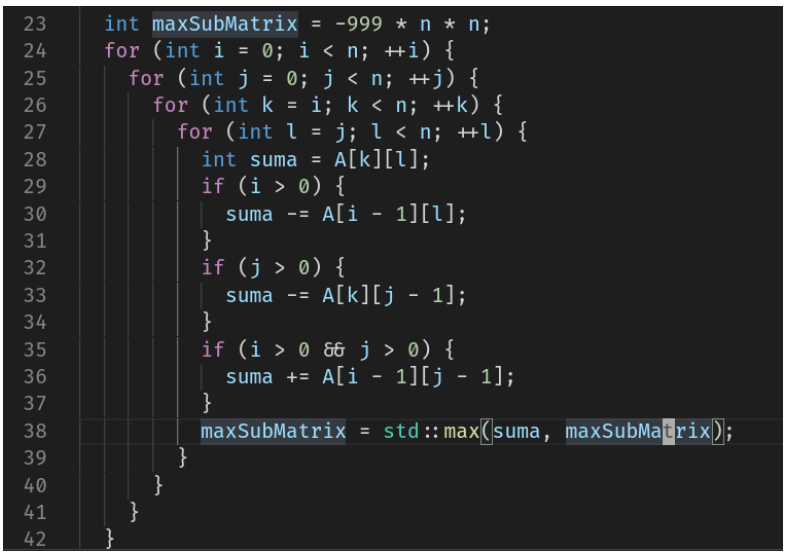
Para hallar la submatriz cuyos elementos dan una suma máxima, podemos hacer uso del paradigma de **divide y vencerás**, sin embargo en el proceso encontraremos que muchas **sumas acumuladas son vueltas a calcular**. Con programación dinámica estás sumas se guardan en una tabla facilitando la consulta, es decir en $O(1)$.

El algoritmo de divide y vencerás usa sus *"locales"* y tendría un **caso base** cuando se consulte sobre la submatriz en $(0,0)$ además de sus **condiciones de límite**, es decir cuando se alcanza la **fila 0** ó **la columna 0**. Tal como se muestra en la figura 1.



2. Programación dinámica

En ves de resolver las sumas acumuladas recursivamente para cada índice (i,j). Hallamos la suma acumulada de sus locales y las almacenamos en una tabla. De esta forma la consulta y la solución de la búsqueda nos tomaría $O(n^4)$. Ver figura 2



```
23  int maxSubMatrix = -999 * n * n;
24  for (int i = 0; i < n; ++i) {
25    for (int j = 0; j < n; ++j) {
26      for (int k = i; k < n; ++k) {
27        for (int l = j; l < n; ++l) {
28          int suma = A[k][l];
29          if (i > 0) {
30            suma -= A[i - 1][l];
31          }
32          if (j > 0) {
33            suma -= A[k][j - 1];
34          }
35          if (i > 0 && j > 0) {
36            suma += A[i - 1][j - 1];
37          }
38          maxSubMatrix = std::max(suma, maxSubMatrix);
39        }
40      }
41    }
42  }
```