Cálculo del Tiempo de Ejecución

Optimizando algoritmos

Problema: encontrar el valor de la suma de la sub-secuencia de suma máxima

Problema de la subsecuencia de suma máxima

Dada una secuencia de números enteros, algunos negativos:

$$a_1, a_2, a_3, \dots a_n$$

encontrar el valor máximo de la $\sum_{k=1}^{j} a_k$

Por convención, la suma es cero cuando todos los enteros son negativos.

Suma de la sub-secuencia de suma máxima Versión 1: O(n³)

```
public final class MaxSumTest
/* Cubic maximum contiguous subsequence sum algorithm.
 public static int maxSubSum1( int [ ] a )
   int maxSum = 0;
   for( int i = 0; i < a.length; i++)
      for( int j = i; j < a.length; j++)
           int thisSum = 0;
           for( int k = i; k \le j; k++)
               thisSum += a[k];
           if( thisSum > maxSum )
              maxSum = thisSum;
      return maxSum;
```

Suma de la sub-secuencia de suma máxima Versión 2: O(n²)

```
public final class MaxSumTest
 /* Quadratic maximum contiguous subsequence sum algorithm.
  public static int maxSubSum1( int [] a )
   int maxSum = 0;
   for(int i = 0; i < a.length; i++)
                                             int thisSum = 0;
      for( int j = i; j < a.length; j++)
            int thisSum = 0;
                                               thisSum += a[j];
          for( int k = i; k <= j; k++ )
             thisSum += al k 1:
         if( thisSum > maxSum )
              maxSum = thisSum;
      return maxSum;
```

Suma de la sub-secuencia de suma máxima Versión 2: O(n²)

```
public final class MaxSumTest
/* Quadratic maximum contiguous subsequence sum algorithm.
  public static int maxSubSum2( int [] a )
  int maxSum = 0;
  for( int i = 0; i < a.length; i++)
       int thisSum = 0;
       for( int j = i; j < a.length; j++)
          thisSum += a[j];
          if( thisSum > maxSum )
             maxSum = thisSum;
     return maxSum;
```

Suma de la sub-secuencia de suma máxima Versión 3: O(n*log n)

- /** Solución recursiva:
 - * Explicar la resolución detallada y graficamente */

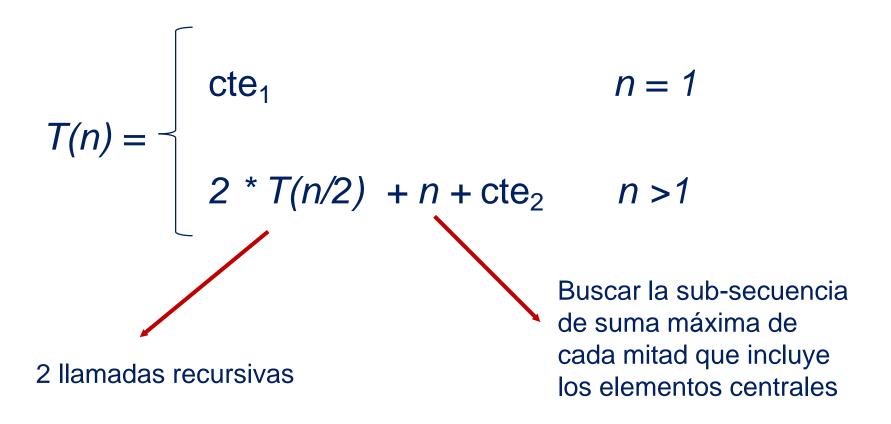
Suma de la sub-secuencia de suma máxima - Versión 3: O(n*log n)

```
private static int maxSumRec( int [] a, int left, int right)
       if( left == right ) // Base case
          if (a[left] > 0)
             return a[left];
          else
             return 0:
           int center = ( left + right ) / 2;
           int maxLeftSum = maxSumRec( a, left, center );
           int maxRightSum = maxSumRec( a, center + 1, right );
           int maxLeftBorderSum = 0, leftBorderSum = 0;
           for( int i = center; i >= left; i-- )
           leftBorderSum += a[ i ];
           if( leftBorderSum > maxLeftBorderSum )
                    maxLeftBorderSum = leftBorderSum:
           int maxRightBorderSum = 0, rightBorderSum = 0;
           for(int i = center + 1; i \le right; i++)
          rightBorderSum += a[ i ];
          if( rightBorderSum > maxRightBorderSum )
                    maxRightBorderSum = rightBorderSum;
            return max3( maxLeftSum, maxRightSum, maxLeftBorderSum + maxRightBorderSum );
```

Suma de la sub-secuencia de suma máxima Versión 3: O(n*log n)

```
/**
      * Driver for divide-and-conquer maximum contiguous * subsequence sum algorithm. */
    public static int maxSubSum3( int [] a ) {
       return maxSumRec( a, 0, a.length - 1);
           /* END */
                 * Return maximum of three integers.
    private static int max3( int a, int b, int c ) {
       return a > b ? a > c ? a : c : b > c ? b : c;
```

Versión 3: Función de Tiempo de Ejecución



Suma de la sub-secuencia de suma máxima Versión 4: O(n)

```
Linear-time maximum contiguous subsequence sum
    algorithm. */
public static int maxSubSum4( int [ ] a )
      int maxSum = 0, thisSum = 0;
      for( int j = 0; j < a.length; j++)
      thisSum += a[j];
      if( thisSum > maxSum )
        maxSum = thisSum;
        else if(thisSum < 0)
       thisSum = 0;
      return maxSum;
```