**Datastrukturer och algoritmer**
Lab 1
Johan Ericsson
Johan Lindkvist
Grupp 40

För varje version skrev vi ned koden och hur många operationer varje rad kommer köras. Därefter gjorde vi en handviftning främst genom att titta på hur många nästlade for-loopar som koden innehöll. Då version 1 innehöll 3 for-loopar blev vår handviftning O(n^3), version 2 blev på samma vis O(n^2) medans version 3 kom att tillhöra O(n).
Se bilder nedanför.

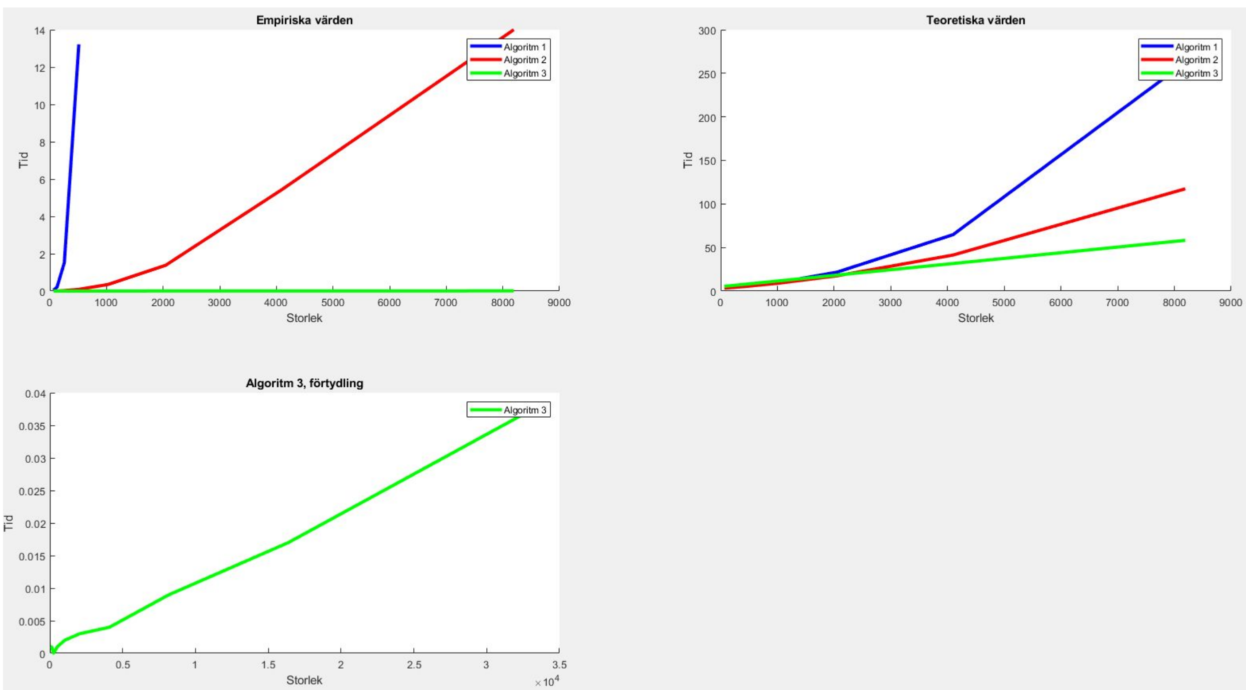
```
// Version 1
public static int maxSubSum1( int [ ] a ) {
        int maxSum = 0;                          (1 op)
        for( int i = 0;                          (1 op)
        i < a.length;                            (1 op every turn)
        i++ )                                    (2 op every turn)
                for( int j = i;                  (1 op every turn)
                j < a.length;                    (1 op every turn) every (n+1)/2 turn
                j++ ){                           (2 op every turn) every (n+1)/2 turn
                thisSum = 0;                     (1 op every turn) every (n+1)/2 turn
                        for( int k = i;          (1 op every turn) every (n+1)/2 turn
                        k <= j;             (1 op every turn) every (n+1)/2 turn*(n+2)/3
                        k++ ){              (2 op every turn) every (n+1)/2 turn*(n+2)/3
                        thisSum += a[k];    (2 op every turn) every (n+1)/2 turn*(n+2)/3
                        }
                        if( thisSum > maxSum )  (1 op every turn) every (n+1)/2 turn*(n+2)/3
                        {
                        maxSum = thisSum;   (1 op every turn) every (n+1)/2 turn*(n+2)/3
                        seqStart = i;       (1 op every turn) every (n+1)/2 turn*(n+2)/3
                        seqEnd = j;         (1 op every turn) every (n+1)/2 turn*(n+2)/3
                        }
        }
return maxSum;                    (1 op)
```

```
// Version 2
          public static int maxSubSum2( int [ ] a ) {
                    int maxSum = 0;                              (1 op)
                    for( int i = 0;                              (1 op)
                              i < a.length;                      (1 op every turn)
                              i++ ) {                            (2 op every turn)
                              int thisSum = 0;                   (1 op every turn)
                                        for( int j = i;          (1 op every turn)
                                        j < a.length;            (1 op every turn2) every turn
                                        j++ ) {  (4 op)          (2 op every turn2) every turn

                                             thisSum += a[ j ];             (2 op et2)et
                                             if( thisSum > maxSum ) {       (1 op et2)et
                                             maxSum = thisSum;              (1 op et2)et
                                                  seqStart = i;             (1 op et2)et
                                                  seqEnd   = j;             (1 op et2)et
                                                  }
                                             }
                                        }
                                        return maxSum;                                (1 op)
                              }


// Version 3
public static int maxSubSum3( int[] a ) {
          int maxSum = 0;                              (1 op)
          int thisSum = 0;                             (1 op)
          for( int i = 0;                              (1 op)
                    j = 0;                             (1 op)
                    j < a.length;                      (1 op every turn)
                    j++ ) {                            (2 op every turn)
                    thisSum += a[j];                   (2 op every turn)
                    if( thisSum > maxSum ) {           (1 op every turn)
                    maxSum = thisSum;                  (1 op every turn)
                    seqStart = i;                      (1 op every turn)
                    seqEnd = j; }                      (1 op every turn)
                    else if( thisSum < 0 ) {           (1 op every turn)
                    i = j + 1;                         (2 op every turn)
                    thisSum = 0;                       (1 op every turn)
                    }
          }
return maxSum;                                         (1 op)
}
```

# Empiriska värden

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_151.jdk/Contents/Home/bin/java ...
simple correctness test
#1 Max sum is 11; it goes from 0 to 6
#2 Max sum is 11; it goes from 0 to 6
#3 Max sum is 11; it goes from 0 to 6
>>>Time is seconds per 2000 calls to MaxSubSum
Size of Array ->         64        128        256        512       1024       2048       4096       8192      16384      32768
algo  #1             0,031s     0,210s     1,564s    13,215s          -          -          -          -          -          -
algo  #2             0,002s     0,005s     0,024s     0,092s     0,356s     1,370s     5,432s          -          -          -
algo  #3             0,001s     0,000s     0,000s     0,001s     0,001s     0,002s     0,005s     0,009s     0,017s     0,037s


Process finished with exit code 0
|
```

# Grafer



# Analyser

Version 1

## Pedantisk analys

$$
\begin{aligned}
T(n) = {} & 1 + \\
& 1 + \\
& 1 \cdot n + \\
& 2 \cdot n + \\
& 1 \cdot n + \\
& 1 \cdot n \cdot \frac{n+1}{2} + \\
& 2 \cdot n \cdot \frac{n+1}{2} + \\
& 1 \cdot n \cdot \frac{n+1}{2} + \\
& 1 \cdot n \cdot \frac{n+1}{2} + \\
& 1 \cdot n \cdot \left(\frac{n+1}{2}\right) \cdot \left(\frac{n+2}{3}\right) + \\
& 2 \cdot n \cdot \left(\frac{n+1}{2}\right) \cdot \left(\frac{n+2}{3}\right) + \\
& 2 \cdot n \cdot \left(\frac{n+1}{2}\right) \cdot \left(\frac{n+2}{3}\right) + \\
& 1 \cdot n \cdot \left(\frac{n+1}{2}\right) \cdot \left(\frac{n+2}{3}\right) + \\
& 1 \cdot n \cdot \left(\frac{n+1}{2}\right) \cdot \left(\frac{n+2}{3}\right) + \\
& 1 \cdot n \cdot \left(\frac{n+1}{2}\right) \cdot \left(\frac{n+2}{3}\right) + \\
& 1 \cdot n \cdot \left(\frac{n+1}{2}\right) \cdot \left(\frac{n+2}{3}\right) + \\
& 1 \\
= {} & 3 + 4n + 5n\frac{n+1}{2} + 9n\left(\frac{n+1}{2}\right)\left(\frac{n+2}{3}\right) \in O(n^3)
\end{aligned}
$$

## Handviftning

3 for-loopar: $O(n^3)$

## Matematisk korrekt analys

$$2 + \sum_{i=1}^{n} 4 \sum_{j=i}^{n} 20 \sum_{k=i}^{j} 5$$

$$= 2 + 400 \sum_{i=1}^{n} \sum_{j=i}^{n} \sum_{k=i}^{j}$$

Version 2

# Handviftning

2 for-loopar: $O(n^2)$

$$
\begin{aligned}
T(n) = {} & 1 + \\
& 1 + \\
& 1 \cdot n + \\
& 2 \cdot n + \\
& 1 \cdot n + \\
& 1 \cdot n + \\
& 1 \cdot \left(\frac{n+1}{2}\right) \cdot n + \\
& 2 \left(\frac{n+1}{2}\right) \cdot n + \\
& 2 \left(\frac{n+1}{2}\right) \cdot n + \\
& 1 \cdot \left(\frac{n+1}{2}\right) \cdot n + \\
& 1 \cdot \left(\frac{n+1}{2}\right) \cdot n + \\
& 1 \cdot \left(\frac{n+1}{2}\right) \cdot n + \\
& 1 \cdot \left(\frac{n+1}{2}\right) \cdot n + \\
& 1
\end{aligned}
$$

$= 3 + 5n + 9\left(\frac{n+1}{2}\right) \cdot n$ = an exact solution $\in O(n^2)$

## Matematisk korrekt uppskattning

$$3 + \sum_{i=1}^{n} 5 + \sum_{i=1}^{n} 9i = 3 + 5\sum_{i=1}^{n} 1 + 9\sum_{i=1}^{n} i = 3 + 5n + 9\frac{n(n+1)}{2} \in O(n^2)$$

$\boxed{\text{Version 3}}$

## Handviftning

En forloop = $O(n)$

## Pedantisk analys

$$T(n) = 1 + 1 + 1 + 1 + 1 \cdot n + 2 \cdot n + 2 \cdot n + 1 \cdot n + 1 \cdot n + 1 \cdot n + 1 \cdot n + 1 \cdot n + 2 \cdot n + 1 \cdot n + 1$$

$$= 5 + 13n \in O(n)$$

## Matematisk korrekt uppskattning

$$5 + \sum_{i=1}^{n} 13 = 5 + 13 \sum_{i=1}^{n} = 5 + 13n \in O(n)$$