TDA416 - Datastrukturer 14/15
Grupp 6
Tim Kerschbaumer
Joakim Berntsson

Laboration 1
Uppgift 2

a)
Metoderna undersöker fältets delintervall och returnerar den största summan som dessa delintervall kan ge.
De börjar med att undersöka index 0 till 0, sedan 0 till 1 ... sedan 0 till n
Sedan index 1 till 1, 1 till 2 ... sedan 1 till n
Detta resulterar i att alla möjliga delintervall undersöks.

b)
Handviftning
maxSubSum1:
Eftersom första metoden innehåller 3 loopar så kommer den ha en ungefärlig komplexitet på O(n^3)

maxSubSum2:
Andra metoden kommer i samma stil ha en ungefärlig O(n^2)

maxSubSum3:
Den tredje kommer ha en ungefärlig på O(n) vilket gör att den bör vara den minst komplexa av de tre

Matematisk analys
Elementära operationer: Tilldelningar, aritmetiska operationer och logiska operationer.
maxSubSum1:

$$T(n) = 2 + \sum_{i=0}^{n-1}(4 + \sum_{j=i}^{n-1}(9 + \sum_{k=i}^{j}(5))) = 2 + 4n + 9\sum_{i=0}^{n-1}(\sum_{j=i}^{n-1}(1) + 5\sum_{i=0}^{n-1}(\sum_{j=i}^{n-1}(j - i + 1)) =$$

$$= 2 + 4n + 9n(n-1+1) - 9n(n-1)/2 + 5\sum_{i=0}^{n-1}((n-i)(n+i-1)/2 - i(n-i) + (n-i)) =$$

$$= 2 + 8.5n + 4.5n^2 + 5\sum_{i=0}^{n-1}((1/2)(n^2 + n + i^2 - i - 2in)) =$$

$$= 2 + 8.5n + 4.5n^2 + 2.5(n^2 + n)n + (5/12)(n^2 - n)(2n - 1) - 2.5n(n-1)/2 - 5n^2(n-1)/2 =$$

$$= 2 + (10/6)n + 7n^2 + (5/6)n^3 \ \varepsilon \ O(n^3)$$

maxSubSum2

$$T(n) = 2 + \sum_{i=0}^{n-1}(5 + \sum_{j=i}^{n-1}(9)) = 2 + \tfrac{19n}{2} + \tfrac{9n^2}{2} \ \varepsilon \ O(n^2)$$

$$T(n) = 2 + 5\sum_{i=0}^{n-1}(1) + 9\sum_{i=0}^{n-1}(n-i) = 2 + 5n + 9n^2 - 9n(n-1)/2 = 4.5n^2 + 9.5n + 2$$

maxSubSum3

$$T(n) = 4 + \sum_{i=0}^{n-1}(9) = 4 + 9n \ \varepsilon \ O(n)$$

TDA416 - Datastrukturer 14/15
Grupp 6
Tim Kerschbaumer
Joakim Berntsson

Pedantisk analys
maxSubSum3

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

Detta blir:

T(n) = 1+1+1+1+1+n(1+2+1+1+1+1+1+1+1+1+1+1) = 5+13n ~ O(n)

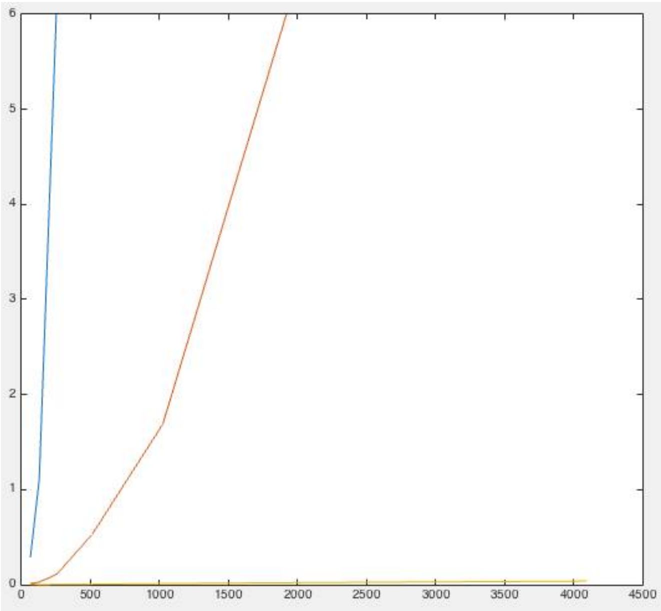
Den pedantiska ger 5 som konstant istället för 4, på grund av att jämförelsen i for-loopen utförs en sista gång för att bestämma att den inte ska fortsätta.

c)

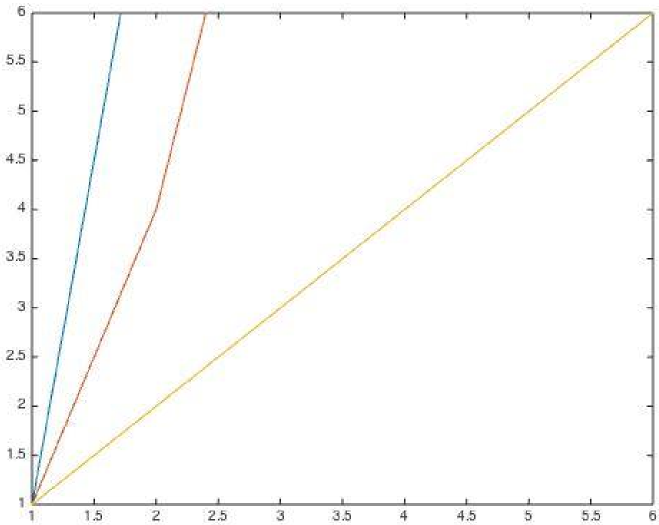
Size	64	128	256	512	1024	2048	4096

TDA416 - Datastrukturer 14/15
Grupp 6
Tim Kerschbaumer
Joakim Berntsson

algo #1	0.288	1.107	6.172	-	-	-	-
algo #2	0.012	0.027	0.112	0.530	1.690	6.619	-
algo #3	0.004	0.001	0.002	0.005	0.012	0.022	0.038
algo #1 O	64^3*2000	128^3*2000	256^3*2000	512^3*2000	1024^3*2000	2048^3*2000	4096^3*2000
algo #2 O	64^2*2000	128^2*2000	256^2*2000	512^2*2000	1024^2*2000	2048^2*2000	4096^2*2000
algo #3 O	64*2000	128*2000	256*2000	512*2000	1024*2000	2048*2000	4096*2000



De verkliga värdena från tabell (algo #1 - blå, algo #2 - orange, algo #3 - gul)



Exempel på en linjär, kvadratisk och kubistisk kurva
(algo #1 - blå, algo #2 - orange, algo #3 - gul)

Hypotes
Vi förväntar oss våra beräkningar skall växa på ett likartat sätt som de faktiska värdena.

TDA416 - Datastrukturer 14/15
Grupp 6
Tim Kerschbaumer
Joakim Berntsson

Resultat

Även om värdena inte är samma (på grund varierande klockfrekvenser) så är ökningarna snarlika. Där den tredje algoritmen är linjärt växande, den andra kvadratiskt och den första kubistiskt. Detta stödjer vår hypotes.