1. Tussen 2 punten ligt 1 lijn, tussen 3 liggen 3, etc…

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Punten | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Lijnen | 1 | 3 | 6 | 10 | 15 | 21 | 28 | 36 | 45 |

Elke keer als er een punt bij komt moet die lijnen naar alle punten die er al zijn hebben, terwijl de punten die er al waren nog steeds hetzelfde aantal lijnen heeft. Dus het aantal verbindingen is: het aantal verbindingen van het vorige aantal punten + het vorige aantal punten.

Vn = Vn-1 + n-1, V2 = 1

V = aantal verbindingen

n = aantal punten

1. Programma
2. T(n) = 3 \* T(n-1) + 2, T(1) = 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| n | 1 | 2 | 3 | 4 | 5 |
| Alg\_a(n) | 1 | 2 | 4 | 8 | 16 |
| Alg\_b(n) | 1 | 2 | 4 | 8 | 16 |

De efficientie van alg\_a is O(2n-1). Dit omdat voor elke n > 1 wordt het algoritme 2 keer extra uitgevoerd.

De efficientie van alg\_b is O(n). Dit omdat voor elke n > 1 wordt het algoritme slechts 1 keer extra uitgevoerd.

1. T(n) = T(n-1) + 3n

T(n) = 3n2

bitMultiply(x, y):

If(x is empty):

Return 0;

Else:

X’ = x >>;

Y’ = y <<;

Result = bitMultiply(x’, y’);

If(least significant bit of x == 1):

Result = bitAddition(result, y);

Return result;

1. T(n) = T(n-1) + 2n

T(n) = 2n2

Power(x, p):

If(p is empty):

Return 1;

Else:

P’ = p – 0001;

Result = power(x, p’);

Result \*= x;

Return result;

1. 1. Programma
   2. Programma
   3. Programma + Inductie

* + 1. Base Case:

N = 1

Linker kant:

Rechter kant:

* + 1. Induction Hypothesize:

voor bepaalde m

* + 1. Induction Step:

* + 1. Gegeven functie:

T(n) = n

O(n)

* + 1. Gemaakte functie:

T(n) = 1

O(1)

1. Programma