國立臺北科技大學
National Taipei University of Technology

資料結構

Data Structure

DS_Assignment01_Group#4

Name & Student ID

吳瑋恩 107300132

劉千榮 110310138

吳翊軒 110310224

余秉勳 110310235

Q1. Ackermann's function

Ackerman's function $A(m, n)$ is defined as:

$$A(m,n) = \begin{cases} n + 1, & \text{if } m = 0. \\ A(m-1,1), & \text{if } n = 0. \\ A(m-1, A(m, n-1)), & \text{otherwise.} \end{cases}$$

This function is studied because it grows very quickly for some small values of m and n.

(a) Write a recursive version of this function.

(b) Using step count to evaluate the performance. Draw the graph with various m, n

(c) Measure the real performance time. Draw the graph with various m, n.

(d) What is the time complexity in big-oh notation?

Code

```
/*****************************************************************************


                    |   n + 1                               , m = 0
Ackermann(m, n) =   |   Ackermann(m - 1, 1)                  , n = 0
                    |   Ackermann(m - 1, Ackermann(m, n - 1))   , otherwise


*****************************************************************************/

#include <stdio.h>
#include <windows.h> //for timeing

int ackerman(int m, int n);

// initial the variable "count"
int count = 0;

int main()
{
    // input variable m & n
    int m, n;

    printf("Please Enter two numbers\n m & n : ");
    // read input from stdin
    scanf("%d %d", &m, &n);

    // set up the variable for timing
```

```
    LARGE_INTEGER t1, t2, tc;

    // check the tick frequency
    QueryPerformanceFrequency(&tc);

    // start timing
    QueryPerformanceCounter(&t1);

    // output the final answer
    printf("Ackermann(%d, %d) = %d\n", m, n, ackerman(m, n));

    // stop timing
    QueryPerformanceCounter(&t2);

    // calculate elapsed time by finding difference (end - begin) and
    // dividing the difference by CLOCKS_PER_SEC to convert to seconds
    double time = ((double)(t2.QuadPart - t1.QuadPart) / (double)tc.QuadPart) *
1000;

    // print the steps
    printf("%s%d\n", "Steps : ", count);

    // print the performance time
    printf("The elapsed time is %lf ms", time);
}

int ackerman(int m, int n)
{
    count++; // for if conditional
    if (m == 0)
    {
        count++; // for return

        // if m = 0, A(m, n) = n + 1
        return n + 1;
    }
    else if (n == 0)
    {
```

```
        count++; // for return

        // if n = 0, A(m, n) = A(m - 1, 1)
        return ackerman(m - 1, 1);
    }
    else
    {
        count++; // for return

        // if (m != 0 && n != 0), A(m ,n) = A(m - 1, A(m, n - 1))
        return ackerman(m - 1, ackerman(m, n - 1));
    }
}
```

Result

#Test 1 - Regular Data

```
PS C:\Users\chann\Desktop\Data Strcture\Assignment 01 - Performance Analysis> ./main.exe
Please Enter two numbers
 m & n : 0 1
Ackermann(0, 1) = 2
Steps : 2
The elapsed time is 0.0020 seconds
```

#Test 2 - Regular Data

```
PS C:\Users\chann\Desktop\Data Strcture\Assignment 01 - Performance Analysis> ./main.exe
Please Enter two numbers
 m & n : 3 1
Ackermann(3, 1) = 13
Steps : 212
The elapsed time is 0.0010 seconds
```

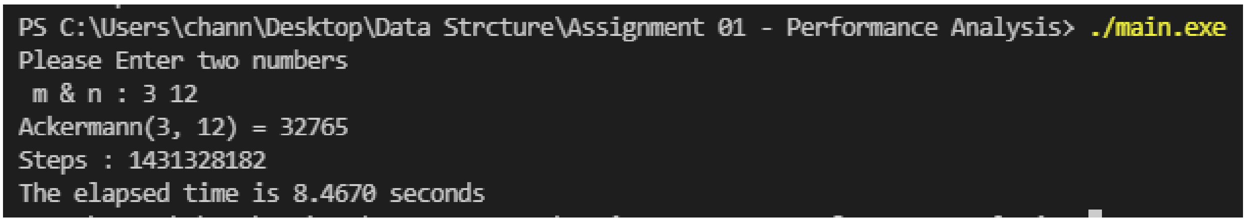
#Test 3 - Regular Data

```
PS C:\Users\chann\Desktop\Data Strcture\Assignment 01 - Performance Analysis> ./main.exe
Please Enter two numbers
 m & n : 2 0
Ackermann(2, 0) = 3
Steps : 10
The elapsed time is 0.0010 seconds
```

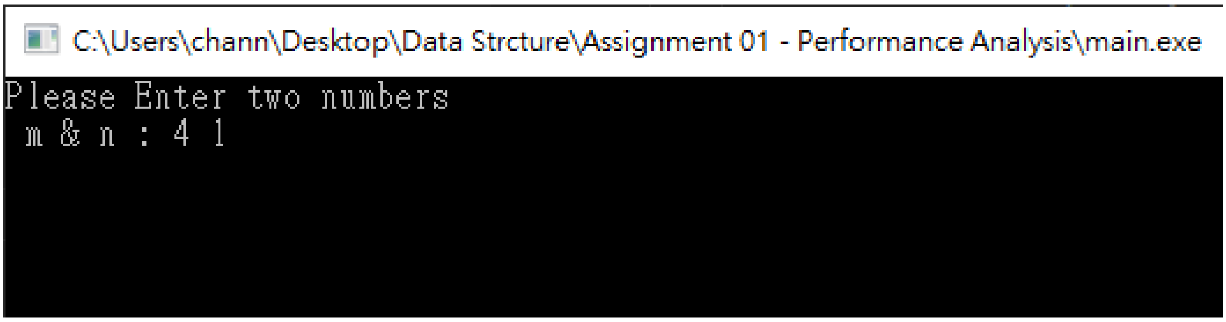
#Test 4 - large number of n

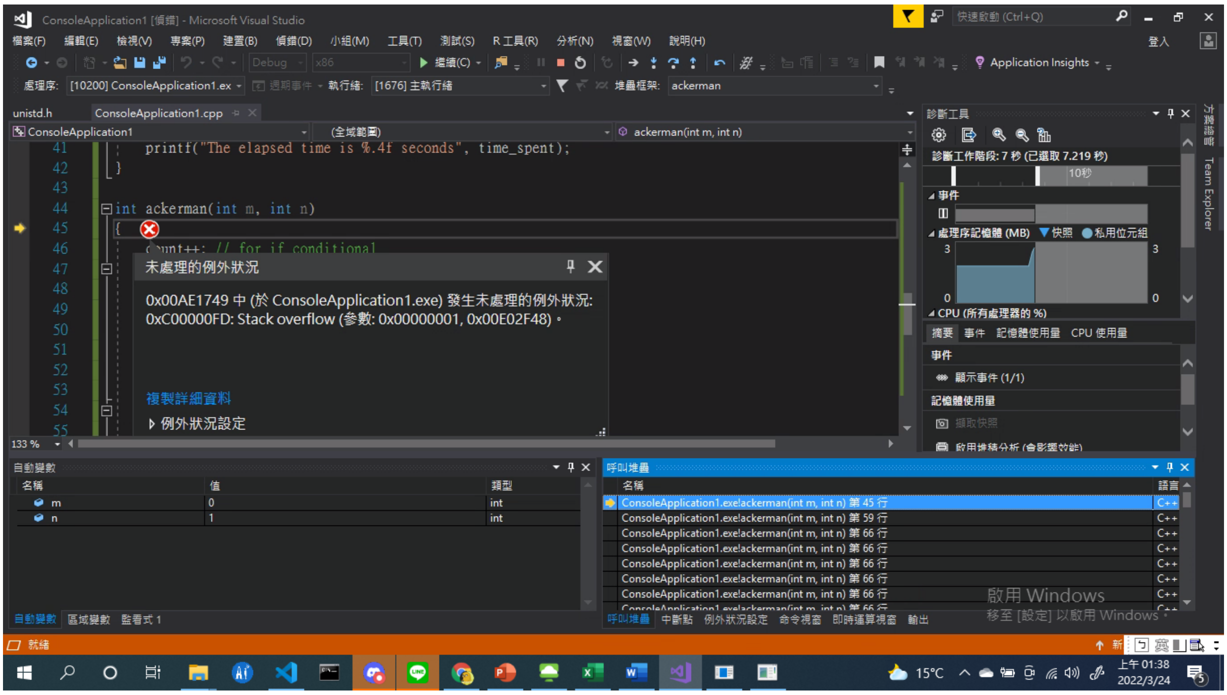
```
PS C:\Users\chann\Desktop\Data Strcture\Assignment 01 - Performance Analysis> ./main.exe
Please Enter two numbers
 m & n : 0 100000
Ackermann(0, 100000) = 100001
Steps : 2
The elapsed time is 0.0020 seconds
```

#Test 5 – longer execution time



#Test 6 (Test on Visual Studio) Stack overflow



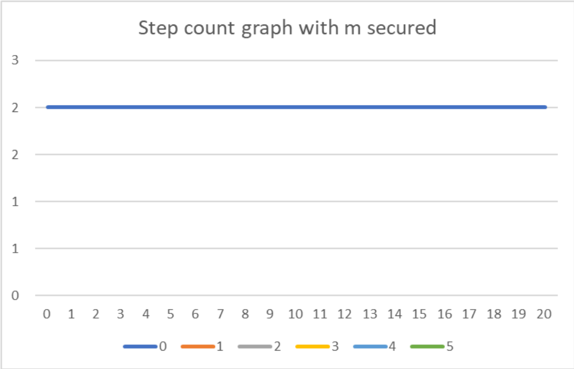


Discussion

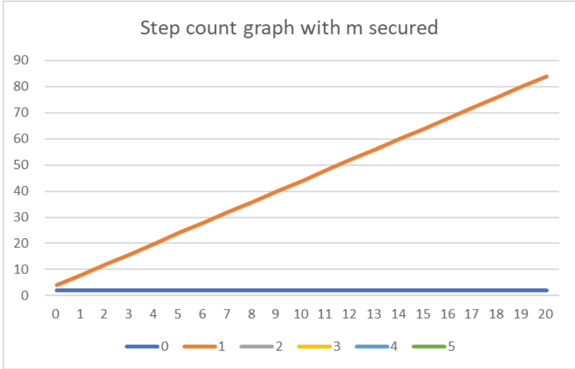
(a) Write a recursive version of this function. Test your code by using the following test cases.

ANS： 程式碼主體放置於 Code 區域，測試數據則為 Result 區域的前三項的 # Test

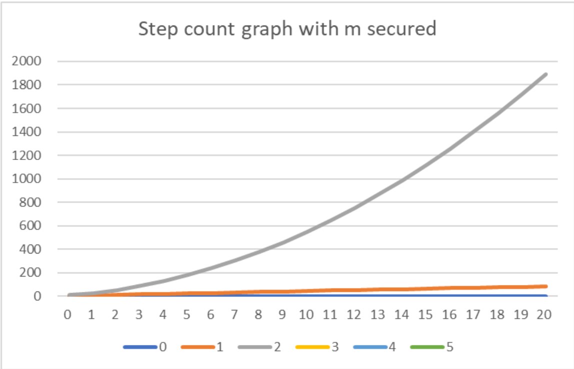
(b) Using step count to evaluate the performance. Draw the graph with various m, n.



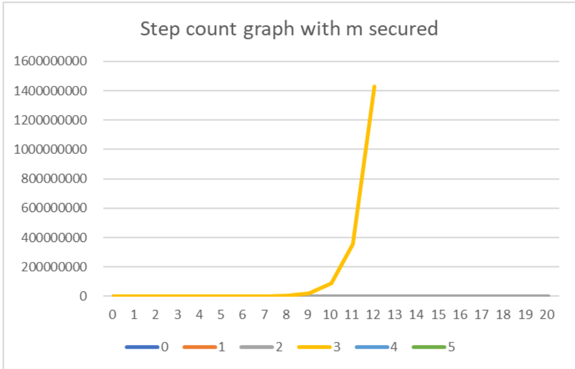
m = 0



m = 1



m = 2



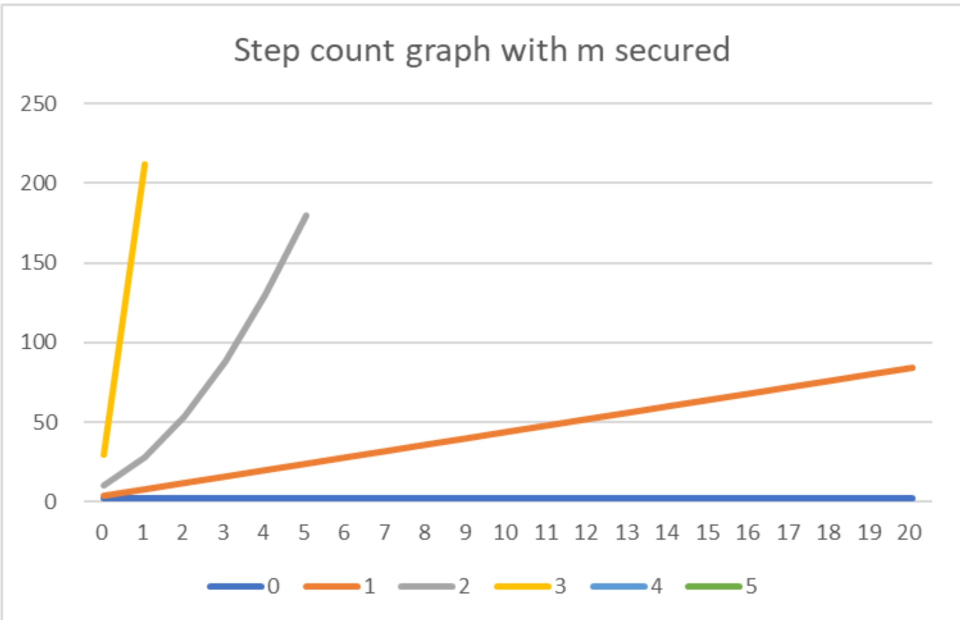
m = 3

The step count equation of different m

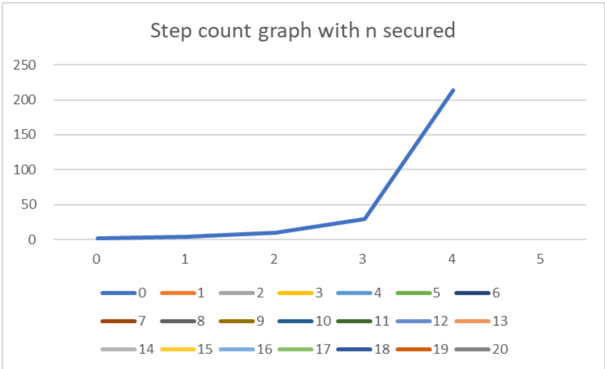
$$m = 0, \qquad step(0, n) = 2$$

$$m = 1, \qquad step(1, n) = 4n + 4$$

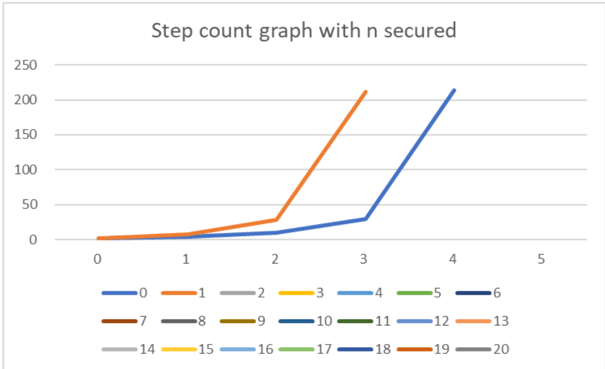
$$m = 2, \qquad step(2, n) = 4n^2 + 14n + 10$$



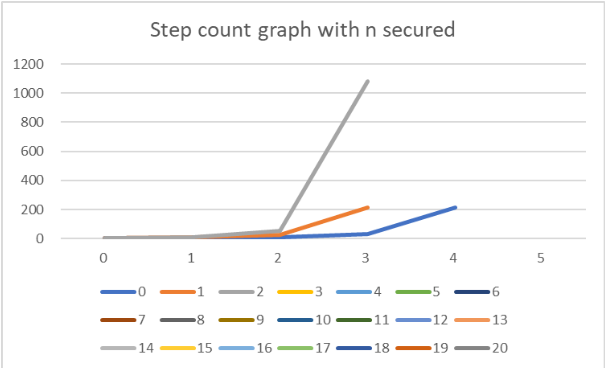
在 m < 2 前，步數成線性關係，如上方圖表所示。當 m≧2，步數即為非線性關係，當 m = 2 時，該數列為典型的二階等差數列。當 m 越來越大後，電腦所需的運算步數急遽增加，其遞增幅度如較大的折線圖所示。當計算至 A(3, 13) 及 A(4, 1) 時，皆因為堆疊溢位(Stack overflow)，導致無法成功跑出運算結果。不過該結果也代表著，當 m 越來越大，使用普通遞迴運算 Ackermann’s function 時，將佔用相當大的記憶體空間，最後導致程式無法順利執行。



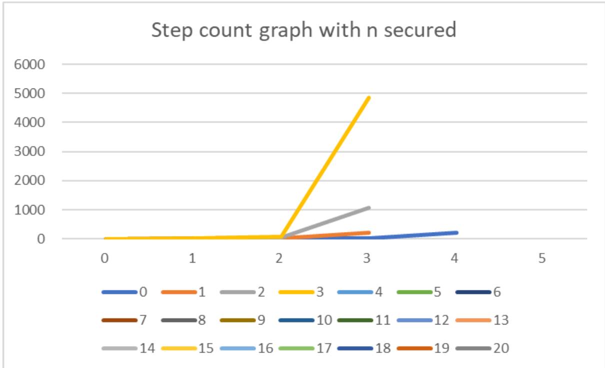
n = 0



n = 1



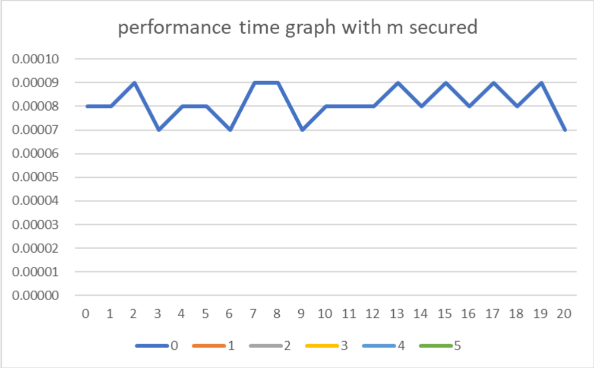
n = 2



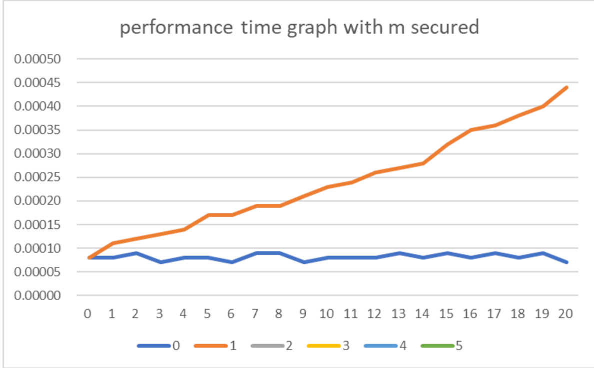
n = 3

在限制 n 不動下，光是在 A(5, 0)及 A(4, 1)就出現堆疊溢位(Stack overflow)的狀況，代表在逐漸增加 m 的情況下，電腦運算步數的上升幅度比逐漸增加 n 大非常多。在 n 為 0 及 1 的圖中，可以感覺到 A(0, n)的運算步數會趨近 A(1, n - 1)，其他在逐漸增大 m 下則無明顯規律。

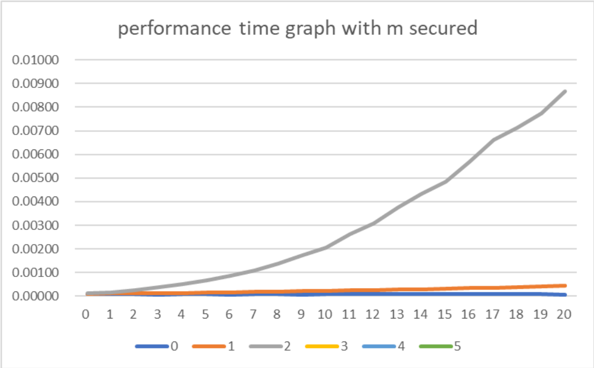
(c) Measure the real performance time. Draw the graph with various m, n.



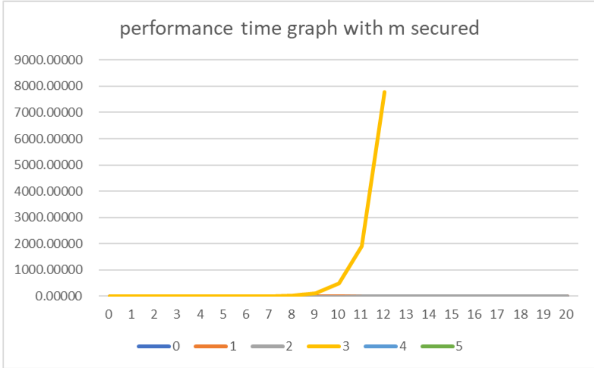
m = 0



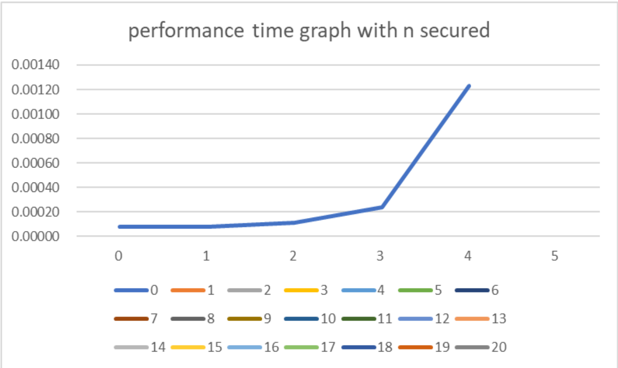
m = 1



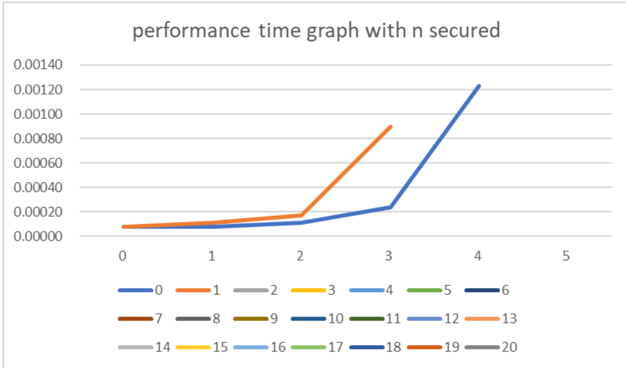
m = 2



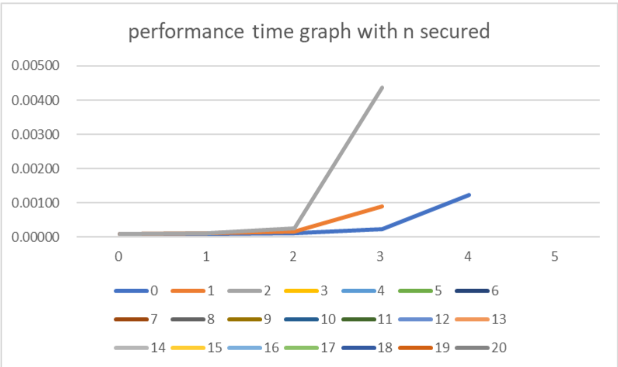
m = 3



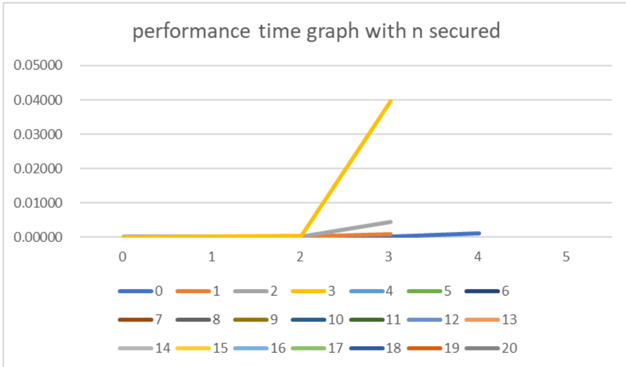
n = 0



n = 1



n = 2



n = 3

利用表格所做出的執行時間圖如上。當執行時間運算時，應將輸出等功能去除，僅留下執行該函式的必要執行項。透過 windows.h 的程式庫，可使計算時間的最小精度與所使用電腦系統相同。理論上，電腦處理每項指令的時間皆相同，故可以推測

$$執行步數 \propto 執行時間$$

而以上的趨勢圖也證實這個推測的方向大致是正確的。若與 b 小題的執行步數圖比對，可以發現是高度相似的。

(d) What is the time complexity in big-oh notation?

	0	1	2	3	4	…	n
0	1	2	3	4	5	…	n+1
1	2	3	4	5	6	…	n+2
2	3	5	7	9	11	…	2(n+3)-3
3	5	13	29	61	125	…	2^{n+3}-3

When $m = 0$

$T(0, n) = 2$

$$So, A(0, n) = O(1)$$

When $m = 1$

$$\begin{aligned}
T(1,n) &= T(0, A(1, n-1)) + T(1, n-1) \\
&= T(0, A(1, n-1)) + T(0, A(1, n-2)) + T(1, n-2) \\
&= T(0, A(1, n-1)) + T(0, A(1, n-2)) + T(0, A(1, n-3)) + T(1, n-3) \\
&\vdots \\
&= \underbrace{T(0, A(1, n-1)) + T(0, A(1, n-2)) + T(0, A(1, n-3)) + \cdots + T(0, A(1, 0))}_{n\ times} + T(1, 0) \\
&= \underbrace{T(0, A(1, n-1)) + T(0, A(1, n-2)) + T(0, A(1, n-3)) + \cdots + T(0, A(1, 0))}_{n\ times} + T(0, 1) + 2 \\
&= \underbrace{T(0, A(1, n-1)) + T(0, A(1, n-2)) + T(0, A(1, n-3)) + \cdots + T(0, A(1, 0))}_{n\ times} + 4 \\
&(We\ know\ A(1, n) = n + 2) \\
&= \underbrace{T(0, n+1) + T(0, n) + T(0, n-1) + \cdots + T(0, 3)}_{n\ times} + 4 \\
&(We\ know\ T(0, n) = 2) \\
&= \underbrace{2 + 2 + 2 + \cdots + 2}_{n\ times} + 4 \\
&= 2n + 4
\end{aligned}$$

$$So, A(1, n) = O(n)$$

When $m = 2$

$$\begin{aligned}
T(2,n) &= T(1,A(2,n-1)) + T(2,n-1) \\
&= T(1,A(2,n-1)) + T(1,A(2,n-2)) + T(2,n-2) \\
&= T(1,A(2,n-1)) + T(1,A(2,n-2)) + T(1,A(2,n-3)) + T(2,n-3) \\
&\vdots \\
&= \underbrace{T(1,A(2,n-1)) + T(1,A(2,n-2)) + T(1,A(2,n-3)) + \cdots + T(1,A(2,0))}_{n\ times} + T(2,0) \\
&= \underbrace{T(1,A(2,n-1)) + T(1,A(2,n-2)) + T(1,A(2,n-3)) + \cdots + T(1,A(2,0))}_{n\ times} + T(1,1) + 2 \\
&= \underbrace{T(1,A(2,n-1)) + T(1,A(2,n-2)) + T(1,A(2,n-3)) + \cdots + T(1,A(2,0))}_{n\ times} + 6 + 2 \quad (T(1,n) = 2n+4) \\
&(We\ know\ A(2,n) = 2n+3) \\
&= \underbrace{T(1,2n+1) + T(1,2n-1) + T(1,2n-3) + \cdots + T(1,3)}_{n\ times} + 8 \\
&(We\ know\ T(1,n) = 2n+4) \\
&= \underbrace{(4n+6) + (4n+2) + (4n-2) + \cdots + 10}_{n\ times} + 8 \\
&= 2n^2 + 8n + 8
\end{aligned}$$

$$So, A(2,n) = O(n^2)$$

When $m = 3$

$$\begin{aligned}
T(3,n) &= T(2,A(3,n-1)) + T(3,n-1) \\
&= T(2,A(3,n-1)) + T(2,A(3,n-2)) + T(3,n-2) \\
&= T(2,A(3,n-1)) + T(2,A(3,n-2)) + T(2,A(3,n-3)) + T(3,n-3) \\
&\vdots \\
&= \underbrace{T(2,A(3,n-1)) + T(2,A(3,n-2)) + T(2,A(3,n-3)) + \cdots + T(2,A(3,0))}_{n\ times} + T(3,0) \\
&= \underbrace{T(2,A(3,n-1)) + T(2,A(3,n-2)) + T(2,A(3,n-3)) + \cdots + T(2,A(3,0))}_{n\ times} + T(2,1) + 2 \\
&= \underbrace{T(2,A(3,n-1)) + T(2,A(3,n-2)) + T(2,A(3,n-3)) + \cdots + T(2,A(3,0))}_{n\ times} + 18 + 2 \quad (T(2,n) = 2n^2+3n+13) \\
&(We\ know\ A(3,n) = 2^{n+3}-3) \\
&= \underbrace{T(2,2^{n+2}-3) + T(2,2^{n+1}-3) + T(2,2^{n}-3) + \cdots + T(2,5)}_{n\ times} + 20 \\
&(We\ know\ T(2,n) = 2n^2+8n+8) \\
&= \underbrace{\left[2(2^{n+2}-3)^2 + 8(2^{n+2}-3) + 8\right] + \left[2(2^{n+1}-3)^2 + 8(2^{n+1}-3) + 8\right] + \left[2(2^{n}-3)^2 + 8(2^{n}-3) + 8\right] + \cdots + 98}_{n\ times} + 20
\end{aligned}$$

去除其他變化小的變數

$$
\begin{aligned}
\text{So, } A(3,n) &\cong O(2^{2n+5} + 2^{2n+5} + 2^{2n+5} + \cdots + 2^7) \\
&= O(\frac{2^7(1-4^n)}{1-4}) \\
&= O(\frac{2^7(2^{2n}-1)}{3}) \\
&= O(2^{14n})
\end{aligned}
$$

在 m 越來越大後，由於數字過於龐大，便只推至 m=3，而從這裡我們可以猜測，當 Ackermann's function 帶入極大的 m 及 n，bigO 大約會是 n^n 或 m^m，代表使用普通遞迴完成的 Ackermann's function 非常浪費效能，需要優化或透過其他手段才能跑出大的數據。

Appendix

Step Count 原始數據

	0	1	2	3	4	5	6	7	8	9
0	2	2	2	2	2	2	2	2	2	2
1	4	8	12	16	20	24	28	32	36	40
2	10	28	54	88	130	180	238	304	378	460
3	30	212	1082	4864	20614	84876	344466	1387928	5571998	22328740

10	11	12	13	14	15	16	17
2	2	2	2	2	2	2	2
44	48	52	56	60	64	68	72
550	648	754	868	990	1120	1258	1404
89396650	357750192	1431328182	-	-	-	-	-

performance time 原始數據

(column : times of n, row : times of m, time unit : ms)

	0	1	2	3	4	5	6
0	0.00008	0.00008	0.00009	0.00007	0.00008	0.00008	0.00007
1	0.00008	0.00011	0.00012	0.00013	0.00014	0.00017	0.00017
2	0.00011	0.00017	0.00025	0.00037	0.00050	0.00068	0.00086
3	0.00024	0.00090	0.00437	0.03955	0.13966	0.46358	2.13361

	7	8	9	10	11	12	13
0	0.00009	0.00009	0.00007	0.00008	0.00008	0.00008	0.00009
1	0.00019	0.00019	0.00021	0.00023	0.00024	0.00026	0.00027
2	0.00110	0.00138	0.00173	0.00205	0.00264	0.00308	0.00377
3	7.66701	29.41691	116.63010	476.72468	1904.12602	7789.8822	-

	14	15	16	17	18	19	20
0	0.00008	0.00009	0.00008	0.00009	0.00008	0.00009	0.00007
1	0.00028	0.00032	0.00035	0.00036	0.00038	0.00040	0.00044
2	0.00433	0.00485	0.00570	0.00662	0.00712	0.00773	0.00868
3	-	-	-	-	-	-	-

The LaTex code of m=1

```
\begin{alignat}{2}
T(1,n) & = \ T(0,A(1,n-1)) + T(1,n-1)\\
& = \ T(0,A(1,n-1)) + T(0,A(1,n-2)) + T(1,n-2) \\
& = \  T(0,A(1,n-1)) + T(0,A(1,n-2)) + T(0,A(1,n-3)) + T(1,n-3) \\
&  \vdots \\
& = \ \begin{matrix} \underbrace{ T(0,A(1,n-1)) + T(0,A(1,n-2)) + T(0,A(1,n-3)) +
\dots+ T(0,A(1,0))  } \\ n \ times\end{matrix} + T(1,0)\\
& = \ \begin{matrix} \underbrace{ T(0,A(1,n-1)) + T(0,A(1,n-2)) + T(0,A(1,n-3)) +
\dots+ T(0,A(1,0))  } \\ n \ times\end{matrix} + T(0,1)+2\\
& = \ \begin{matrix} \underbrace{ T(0,A(1,n-1)) + T(0,A(1,n-2)) + T(0,A(1,n-3)) +
\dots+ T(0,A(1,0))  } \\ n \ times\end{matrix} + 4\\

&(We\ know\ A(1,n)=n+2) \\
\\

& = \ \begin{matrix} \underbrace{ T(0,n+1) + T(0,n) + T(0,n-1) + \dots+ T(0,3)  } \\
n \ times\end{matrix} + 4\\
&(We\ know\ T(0,n)=2) \\
\\

& = \ \begin{matrix} \underbrace{ 2 + 2 + 2 + \dots+ 2  } \\ n \ times\end{matrix} +
4\\
& = \ 2n+4

\end{alignat}
```

The LaTex code of m=2

```
\begin{alignat}{2}
T(2,n) & = \ T(1,A(2,n-1)) + T(2,n-1)\\
& = \ T(1,A(2,n-1)) + T(1,A(2,n-2)) + T(2,n-2) \\
```

```
& = \ T(1,A(2,n-1)) + T(1,A(2,n-2)) + T(1,A(2,n-3)) + T(2,n-3) \\
&  \vdots \\
& = \ \begin{matrix} \underbrace{ T(1,A(2,n-1)) + T(1,A(2,n-2)) + T(1,A(2,n-3)) +
\dots+ T(1,A(2,0))  } \\ n \ times\end{matrix} + T(2,0)\\
& = \ \begin{matrix} \underbrace{ T(1,A(2,n-1)) + T(1,A(2,n-2)) + T(1,A(2,n-3)) +
\dots+ T(1,A(2,0))  } \\ n \ times\end{matrix} + T(1,1)+2\\
& = \ \begin{matrix} \underbrace{ T(1,A(2,n-1)) + T(1,A(2,n-2)) + T(1,A(2,n-3)) +
\dots+ T(1,A(2,0))  } \\ n \ times\end{matrix} + 6+2\ \ \ \ \ (T(1,n)=2n+4)\\

&(We\ know\ A(2,n)=2n+3) \\
\\

& = \ \begin{matrix} \underbrace{ T(1,2n+1) + T(1,2n-1) + T(1,2n-3) + \dots+
T(1,3)  } \\ n \ times\end{matrix} + 8\\

&(We\ know\ T(1,n)=2n+4) \\
\\

& = \ \begin{matrix} \underbrace{ (4n+6) + (4n+2) + (4n-2) + \dots+ 10  } \\ n \
times\end{matrix} + 8\\
& = \ 2n^2+8n+8
\end{alignat}\\
```

The LaTex code of m=3

```
\begin{alignat}{2}
T(3,n) & = \ T(2,A(3,n-1)) + T(3,n-1)\\
& = \ T(2,A(3,n-1)) + T(2,A(3,n-2)) + T(3,n-2) \\
& = \ T(2,A(3,n-1)) + T(2,A(3,n-2)) + T(2,A(3,n-3)) + T(3,n-3) \\
&  \vdots \\
& = \ \begin{matrix} \underbrace{ T(2,A(3,n-1)) + T(2,A(3,n-2)) + T(2,A(3,n-3)) +
\dots+ T(2,A(3,0))  } \\ n \ times\end{matrix} + T(3,0)\\
& = \ \begin{matrix} \underbrace{ T(2,A(3,n-1)) + T(2,A(3,n-2)) + T(2,A(3,n-3)) +
\dots+ T(2,A(3,0))  } \\ n \ times\end{matrix} + T(2,1)+2\\
& = \ \begin{matrix} \underbrace{ T(2,A(3,n-1)) + T(2,A(3,n-2)) + T(2,A(3,n-3)) +
\dots+ T(2,A(3,0))  } \\ n \ times\end{matrix} + 18+2\ \ \ \ \ (T(2,n)=2n^2+3n+13)\\

&(We\ know\ A(3,n)=2^{n+3}-3) \\
```

```
\\

& = \ \begin{matrix} \underbrace{ T(2,2^{n+2}-3) + T(2,2^{n+1}-3) + T(2,2^{n}-3) +
\dots+ T(2,5)  } \\ n \ times\end{matrix} + 20\\

&(We\ know\ T(2,n)=2n^2+8n+8) \\
\\

& = \ \begin{matrix} \underbrace{ \left[2(2^{n+2}-3)^2+8(2^{n+2}-3)+8\right] +
\left[2(2^{n+1}-3)^2+8(2^{n+1}-3)+8\right] + \left[2(2^{n}-3)^2+8(2^{n}-3)+8\right]
+ \dots+ 98 } \\ n \ times\end{matrix} + 20\\

\end{alignat}\\

\begin{alignat}{2}

\text{So, } A(3,n) & \cong  O(2^{2n+5} + 2^{2n+5} + 2^{2n+5} + \dots+2^7) \\
& = O(\frac{2^7(1-4^n)}{1-4}) \\
& = O(\frac{2^7(2^{2n}-1)}{3}) \\
& = O(2^{14n})

\end{alignat}\\
```