

## 演算法作業二

解決的問題:

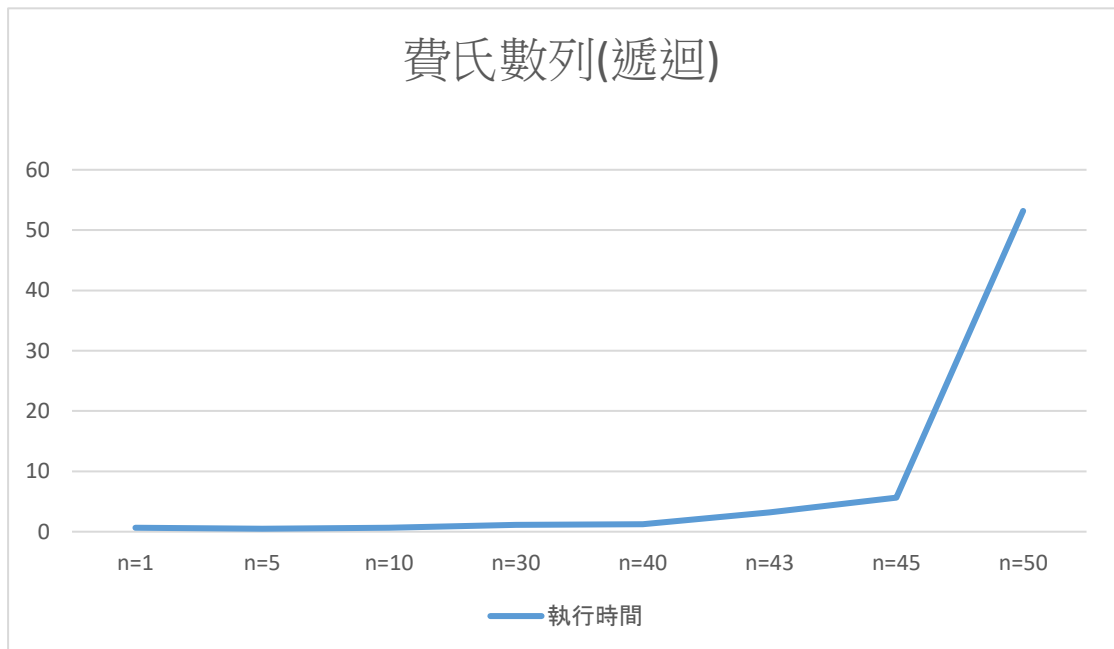
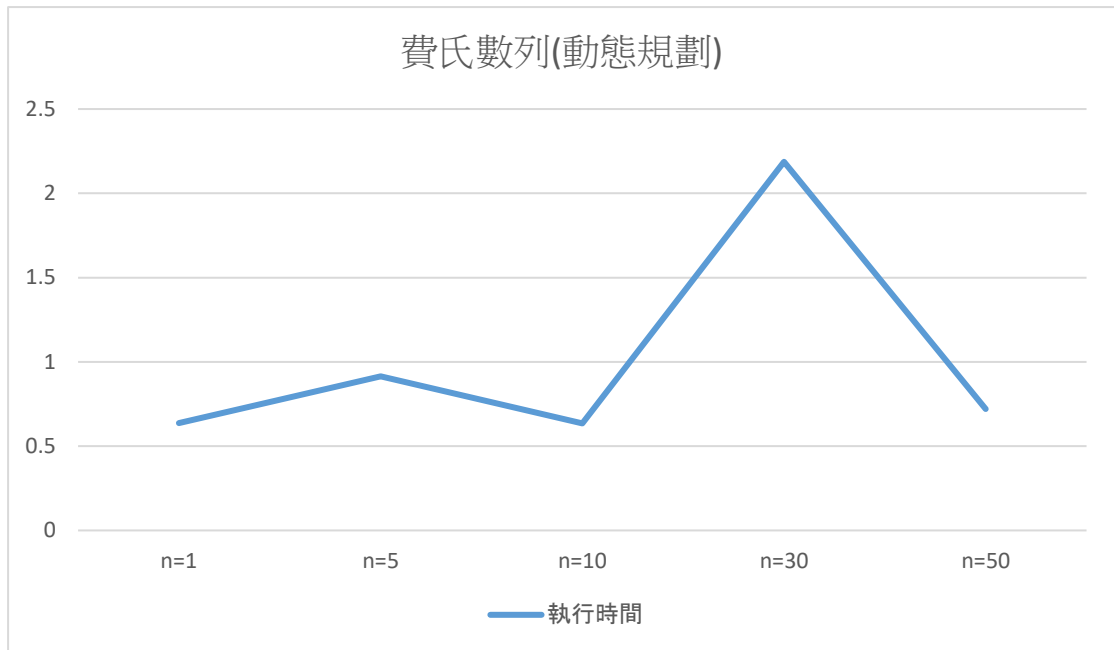
### 1. 費波納契數列

(1) 動態規劃:

```
#include<stdio.h>
int main()
{
    long long a[100];
    int n;
    int i=2;
    a[0]=0;
    a[1]=1;
    a[2]=1;
    scanf("%d",&n);
    if(n==1)
        printf("1");
    else{
        while(1)
        {
            a[i]=a[i-1]+a[i-2];
            if(i>=n)
                break;
            else
                i++;
        }
        printf("%llu\n", a[i]);
    }
    return 0;
}
```

(2) 遞迴:

```
#include<stdio.h>
int n = 0;
int m = 0;
long long fun(int p)
{
    if (p == 1 || p == 2)
    {
        return 1;
    }
    else
    {
        return fun(p - 1) + fun(p - 2);
    }
}
int main()
{
    scanf("%d", &n);
    m = fun(n);
    printf("%llu",m);
    return 0;
}
```



可以透過實驗數據得知

動態規劃執行時間不會隨著  $n$  大小成長，由於每一個計算過程都已經儲存在空間中不會需要耗費多餘時間計算，但是遞迴就不同，當  $n \geq 47$  以上遞迴就有機會比動態規劃多執行超過 10 秒

```
C:\C\演算法作業2費氏遞迴.exe
47
2971215073
Process returned 0 (0x0)   execution time : 12.746 s
Press any key to continue.
```

```
C:\C\演算法作業2費氏動態規劃.exe
47
2971215073
Process returned 0 (0x0)   execution time : 0.981 s
Press any key to continue.
```

2. 樓梯有  $n$  階，每次可以走 1 階或是 3 階，請問  $n$  階樓梯總共有幾種走法？

### (1) 動態規劃

```
#include<stdio.h>
int F[1005][1005];

int f(int x, int y)
{
    if(x==0) return 2;
    if(y==0) return 3;

    if(F[x][y]!=-1) return F[x][y];

    F[x][y]=f(x-1, y)+f(x, y-1);
    return F[x][y];
}

int main()
{
    for(int i=0;i<1003;i++){
        for(int j=0;j<1003;j++){
            F[i][j]=-1;
        }
    }

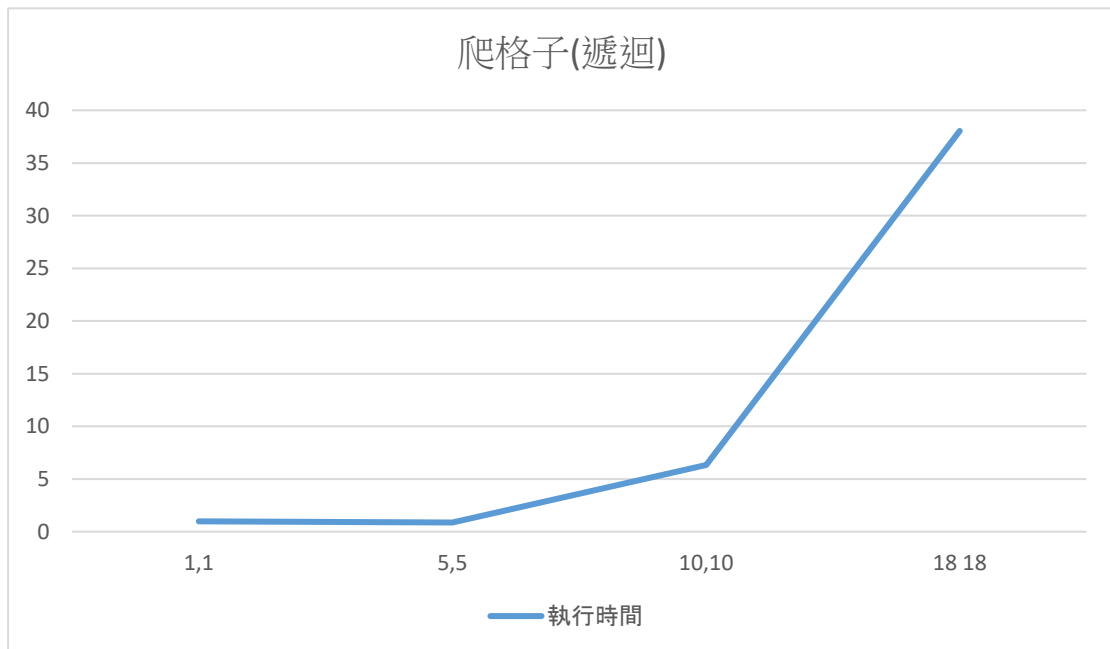
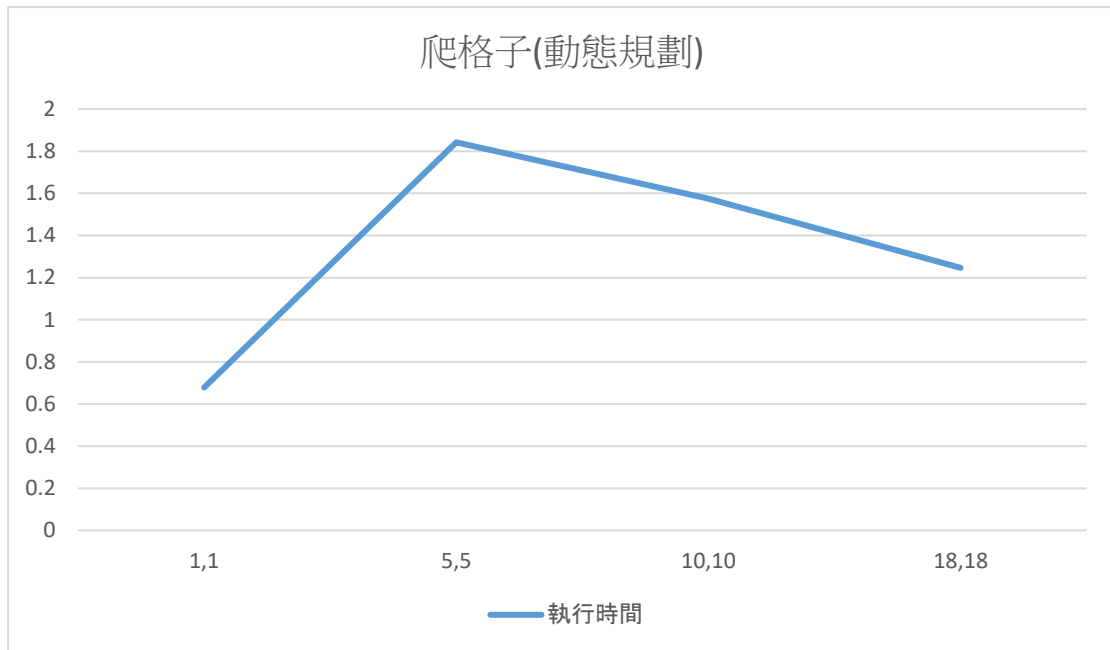
    int x, y, m;
    scanf("%d%d", &x, &y);
    m=f(x, y);
    printf("%d", m);

    return 0;
}
```

### (2) 遞迴

```
#include<stdio.h>
int x, y;
int m;
int f(int x, int y)
{
    if(x==0) return 2;
    if(y==0) return 3;
    return f(x-1, y)+f(x, y-1);
}

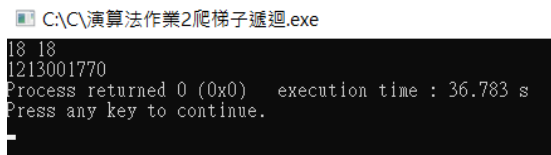
int main()
{
    scanf("%d%d", &x, &y);
    m = f(x, y);
    printf("%d", m);
    return 0;
}
```



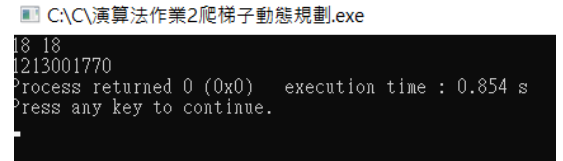
可以透過實驗數據得知

動態規劃執行時間不會隨著  $n$  大小成長，由於每一個計算過程都已經儲存在空間中不會需要耗費多餘時間計算，但是遞迴就不同，當  $x,y$  超過 17 以上就有機會比動態規劃多執行超過 10 秒

18,18 則是一定保守估計 30 秒以上



```
C:\演算法作業2爬梯子遞迴.exe
18 18
1213001770
Process returned 0 (0x0) execution time : 36.783 s
Press any key to continue.
```



```
C:\演算法作業2爬梯子動態規劃.exe
18 18
1213001770
Process returned 0 (0x0) execution time : 0.854 s
Press any key to continue.
```

總結:

本篇以費氏數列與爬格子方法數為例，重點對動態規劃演算法和遞迴演算法做分析比較，通過一系列的實驗資料表明，動態規劃演算法與遞迴演算法相比降低的時間複雜度和空間複雜度，從而提高工作效率，節省時間。本篇實驗結果與預期實驗結果一致。