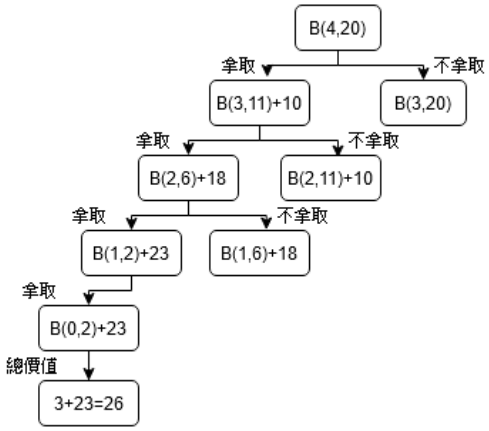
一、動態規劃解 01 背包問題：

假設：

序號(k)	重量(w)	價值(v)
0	2	3
1	3	4
2	4	5
3	5	8
4	9	10

$$B(k,w)=\begin{cases} B(k-1,w) & \text{當第k件物品過重} \\ max\begin{cases}B(k-1,w-w_k)+v_k & \text{拿取物品} \\ B(k-1,w) & \text{不拿取物品}\end{cases}\end{cases}$$



公式及動態規劃遞迴關係圖

1. 部分程式碼註解

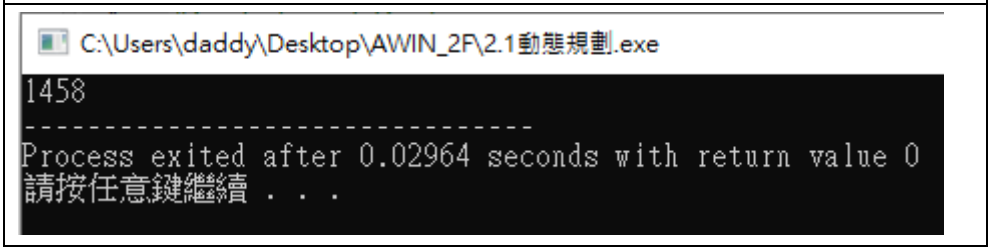
```
for(i=0;i<=n;i++)
{
    for(w=0;w<=W;w++)
    {
        if(i==0||w==0)
            K[i][w]=0; //當袋子容量為0時
        else if(wt[i-1]<=w)
            K[i][w] = max(val[i-1] + K[i-1][w-wt[i-1]], K[i-1][w]); //普遍情況
        else
            K[i][w]=K[i-1][w]; //過重時拿出最後一樣物品
    }
}

return K[n][W];
```

將狀況分為（背包容量為 0）、 普遍情況 以及 過重時

2. 執行方法

將物品價值、重量、背包重量輸入後便會算出最佳解。

```
int val[]={135,139,149,150,156,163,173,184,192,201,210,214,221,229,240};
int wt[]= { 70, 73, 77, 80 ,82, 87, 90, 94, 98,106,110,113,115,118,120};
int W=750;
```



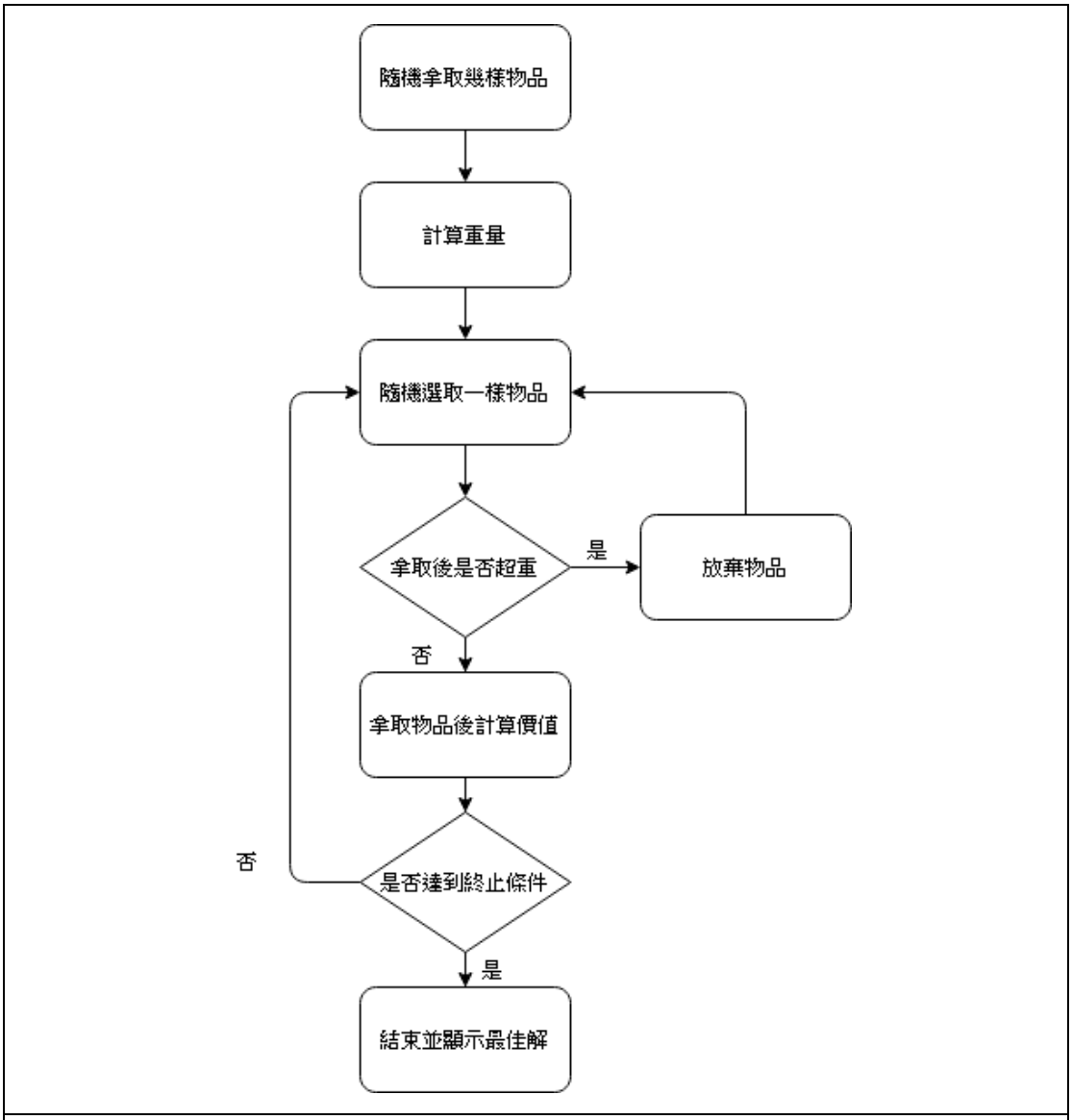
3. 遇到的困難

起初一開始看到題目的時候也是不知道需要什麼，連題目要求都有點不清楚，後來四處尋找相關資料後才知道原來是經典的題型 01 背包問題，這時候才開始有了方向。但一開始到處看了許多文字解說、範例程式都不是很了解為什麼會這樣設計程式，但後來發現了一部解說的影片後，就很快的看懂了。不過很可惜的是試了許久卻無法將結果的組合方式顯示出來，因考慮時間關係，故先將作業交出後並繼續嘗試將其印出。

4. 參考資料

https://www.bilibili.com/video/av36136952/

二、爬山演算法解 01 背包問題



流程圖

1. 部分程式碼註解

```
srand( time(NULL) );
printf("第 1次隨機拿取的物品有：");
for(int i=0;i<15;i++)
{
    random[i]=rand() % (1 - 0 + 1) + 0;
}
for(int j=0;j<15;j++)
{
    printf("%d,",random[j]);
    if(random[j]==1)
    {
        p+=val[j];
        w+=wt[j];
    }
}
printf("\b  第 1\t次拿取的價值與重量為 ： %d,%d\n",p,w);
```

先隨機拿取幾樣物品，1 為拿取，0 為不拿

```
if(w>maxw)
{
    int r= rand() % (15 - 0 + 1) + 0;
    p=0;w=0;
    if(random[r]==1)
    {
        printf("\n放棄拿取第 %3d 樣物品：",r);
        random[r]=0;
    }
    else
        continue;
    for(int j=0;j<15;j++)
    {
        printf("%d,",random[j]);
        if(random[j]==1)
        {
            p+=val[j];
            w+=wt[j];
        }
    }
    printf("\b  第 %3d 次拿取的價值、重量與目前最佳價值為 ： %4d,%4d",times+1,p,w);
}
```

第一次拿取的物品若超重，則隨機放棄一樣物品

```
else if(w<=maxw)
{
    int r= rand() % (15 - 0 + 1) + 0;
    p=0;w=0;
    for(int k=0;k<15;k++)
    {
        n[k]=random[k];
    }
    if(random[r]==0)
    {
        printf("\n選擇拿取第 %3d 樣物品：",r);
        random[r]=1;
    }
    else
        continue;
    for(int j=0;j<15;j++)
    {
        printf("%d,",random[j]);
        if(random[j]==1)
        {
            p+=val[j];
            w+=wt[j];
        }
    }
    printf("\b  第 %3d 次拿取的價值、重量與目前最佳價值為 ： %4d,%4d",times+1,p,w);
    if(p>bestval && w<=maxw)
    {
        for(int k=0;k<15;k++)
        {
            best[k]=n[k];
        }
        besttimes=times;
        bestval=p;
        bestw=w;
    }
}
printf(",%4d\n",bestval);
```

下一次拿取的物品若沒超重，則計算價值，並記錄下來

2. 執行方法

將物品價值、重量、背包重量、猜測次數輸入後便會算出最佳解。

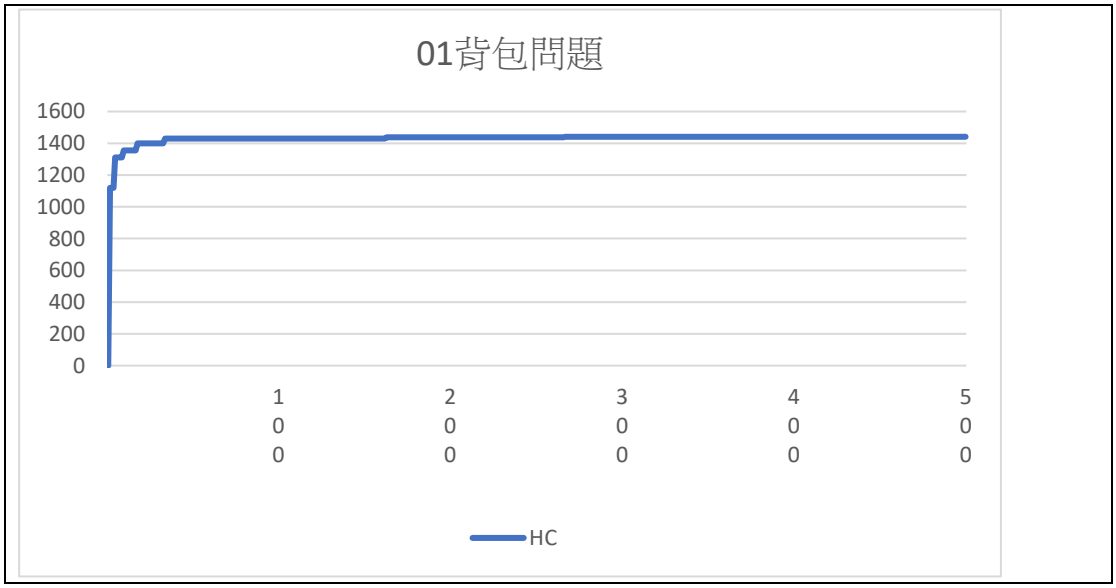
```
int val[]={135,139,149,150,156,163,173,184,192,201,210,214,221,229,240};
int wt[]= { 70, 73, 77, 80 ,82, 87, 90, 94, 98,106,110,113,115,118,120};
int maxw=750;
int guess_times=500;
```

```
C:\Users\daddy\Desktop\AWIN_2F\2.2爬山演算法.exe
放棄拿取第  12 樣物品：1,0,1,0,1,0,0,0,1,1,1,1,0,0,0  第 489 次拿取的價值、重量與目前最佳價值為 : 1257, 656,1430
選擇拿取第   3 樣物品：1,0,1,1,1,0,0,0,1,1,1,1,0,0,0  第 490 次拿取的價值、重量與目前最佳價值為 : 1407, 736,1430
選擇拿取第   5 樣物品：1,0,1,1,1,1,0,0,1,1,1,1,0,0,0  第 491 次拿取的價值、重量與目前最佳價值為 : 1570, 823,1430
放棄拿取第   9 樣物品：1,0,1,1,1,1,0,0,1,0,1,1,0,0,0  第 492 次拿取的價值、重量與目前最佳價值為 : 1369, 717,1430
選擇拿取第   7 樣物品：1,0,1,1,1,1,0,1,1,0,1,1,0,0,0  第 493 次拿取的價值、重量與目前最佳價值為 : 1553, 811,1430
放棄拿取第   0 樣物品：0,0,1,1,1,1,0,1,1,0,1,1,0,0,0  第 494 次拿取的價值、重量與目前最佳價值為 : 1418, 741,1430
選擇拿取第   0 樣物品：1,0,1,1,1,1,0,1,1,0,1,1,0,0,0  第 495 次拿取的價值、重量與目前最佳價值為 : 1553, 811,1430
放棄拿取第   4 樣物品：1,0,1,1,0,1,0,1,1,0,1,1,0,0,0  第 496 次拿取的價值、重量與目前最佳價值為 : 1397, 729,1430
選擇拿取第   1 樣物品：1,1,1,1,0,1,0,1,1,0,1,1,0,0,0  第 497 次拿取的價值、重量與目前最佳價值為 : 1536, 802,1430
選擇拿取第   4 樣物品：1,1,1,1,1,1,0,1,1,0,1,1,0,0,0  第 498 次拿取的價值、重量與目前最佳價值為 : 1692, 884,1430
放棄拿取第  11 樣物品：1,1,1,1,1,1,0,1,1,0,1,0,0,0,0  第 499 次拿取的價值、重量與目前最佳價值為 : 1478, 771,1430
放棄拿取第   1 樣物品：1,0,1,1,1,1,0,1,1,0,1,0,0,0,0  第 500 次拿取的價值、重量與目前最佳價值為 : 1339, 698,1430

最佳為第463次的：1430 ,743 ,拿取物品為：0 0 1 1 0 0 1 1 1 0 0 1 0 1 0
--------------------------------
Process exited after 8.861 seconds with return value 0
請按任意鍵繼續 . . .
```

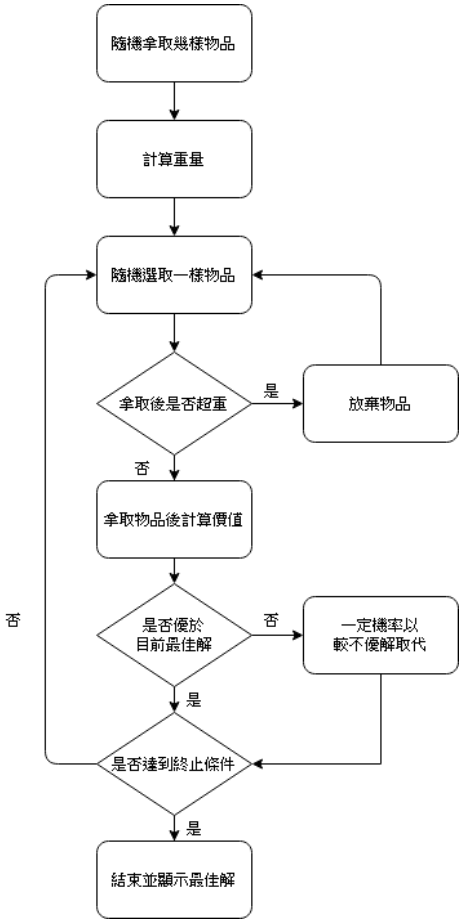
3. 遇到的困難

寫完第一題後就開始尋找有關爬山演算法解此題的例子，但是找來找去似乎都是其他的啟發式演算法，幾乎沒看到有使用爬山演算法的例子，於是只好嘗試著用自己所理解的爬山演算法來嘗試從頭寫寫看，但寫出來的答案在猜測次數少的時候猜對的機率很低，不確定是不是我對於此演算法的理解有錯誤。

三、爬山演算法解 01 背包問題 過程圖



四、模擬退火演算法解 01 背包問題



流程圖

1. 部分程式碼註解

```
dif1 = calprofit(a) - bestmaxp;
if (dif1 > 0)
{
    for (j = 0 ; j < M; j++)
    {
        preseq[j] = a[j];
        bestseq[j] = a[j];
    }
   premaxp = calprofit(a);
   bestmaxp = premaxp;
}
```

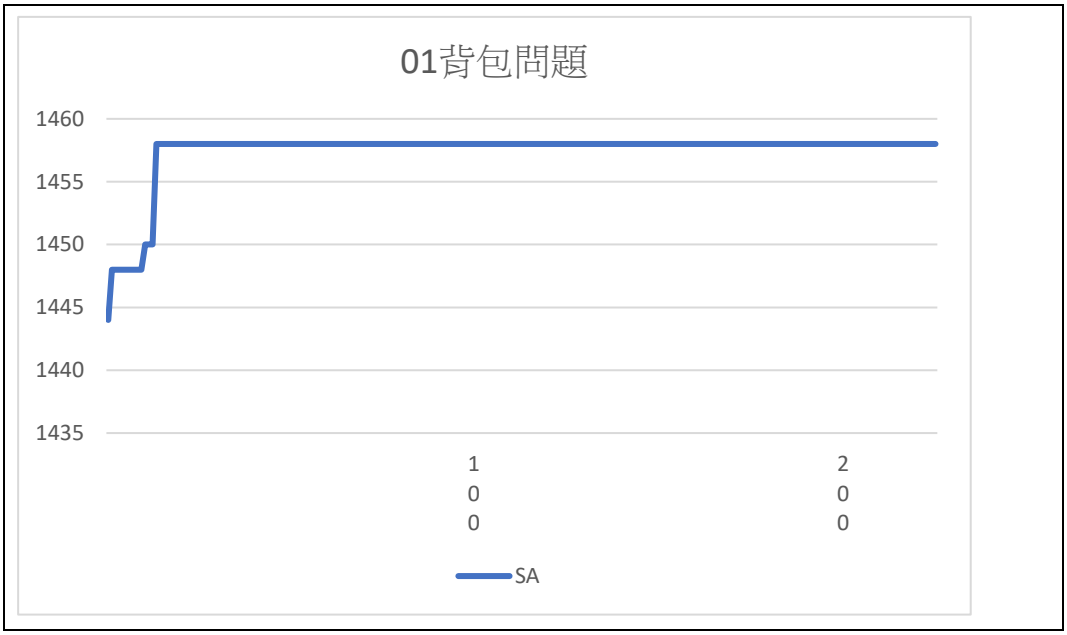
進行計算，若比當前最高價值優秀則保留下來並記錄

```
else
{
    dif2 = calprofit(a) - premaxp;
    if (dif2 > 0)
    {
        for (j = 0; j < M; j++)
        {
            preseq[j] = a[j];
        }
        premaxp = calprofit(a);
    }
    else
    {
        p = rand()%20001/20000.0;
        if (exp((dif2)/ t) > p)
        {
            for (j = 0; j < M; j++)
            {
                preseq[j] = a[j];
            }
        premaxp = calprofit(a);
        }
        else
        {
            for (j = 0; j < M; j++)
            {
                a[j] = preseq[j];
            }
        }
    }
}
```

若小於當前最佳解，則換取另一樣隨機物品，再次計算。

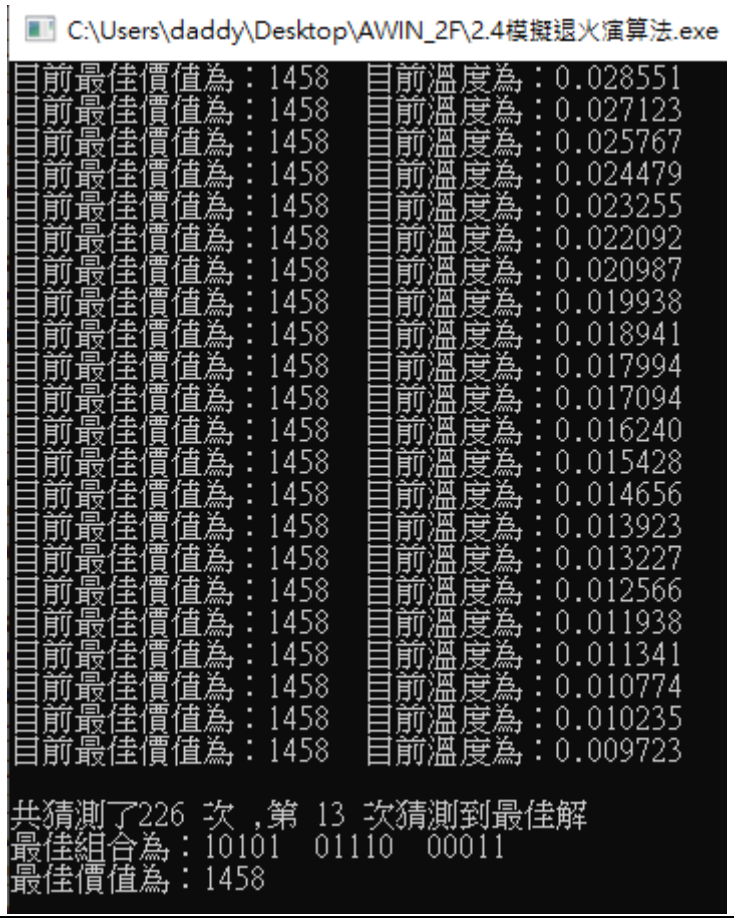
若大於當前最佳解則取代，若小於則以一定機率取代。

2. 模擬退火演算法迭代過程圖。



3. 執行方法

```
#define T0 1000 //初始溫度
#define TF 0.01 //結束溫度
#define T 0.95  //溫度變化率
#define N 500
#define M  15

int weight[M]= {  70, 73, 77, 80, 82, 87, 90, 94, 98,106,110,113,115,118,120},
    profit[M]= { 135,139,149,150,156,163,173,184,192,201,210,214,221,229,240},
    contain = 750;
```



輸入 物品數量(M)、價值、重量、以及背包重量即可開始計算

4. 演算法比較

如果我的爬山演算法是正確的話，這幾次測試下來模擬退火演算法的確比爬山演算法快得非常多也更為準確，在查資料的時候也發現，在實際應用上，鮮少有人使用爬山演算法，畢竟爬山演算法容易落入區域最佳解，我想這不是大家所需要的答案。

5. 遇到的困難

模擬退火的例子比起爬山好找多了，一開始馬上就看到了這個範例，但一開始苦惱的是它並沒有任何的說明註解，甚至連排版都沒有，花了一點時間還自己看懂了解他，這次花的時間比第一大題的模擬退火以及第二大題的動態規劃快了許多，想必是因為先前在寫的時候看了很多資料，所以對這些演算法的敏銳度開始有所提升了。想必未來需要做更多的練習，才能再將速度加快！