Input 方式：

兩組 Data Flow Graph 分別以.txt 的方式做資料建構，分別為 p1.txt 及 p2.txt。

eg:
1 2 3 8
表示：v8=v2+v3

1　：表示加法 (若數字為 2 則表示乘法) 。
2 , 3 : 為運算元。
8　：為儲存的運算元。

範例檔案：

```
3                          //共有 3 個運算
1 2 3 8                     // v8= v2 + v3
1 6 7 11                   // v11= v6 + v7
1 10 3 12                  // v12= v10+v3
```

演算法架構：

此程式採用 List Scheduling 為演算基礎。

資料結構：

將.txt 的檔案做載入的動作放入 sample1。

```
struct cdfg{
        int op;             //運算子 eg:+,-
        int op1;            //運算元 eg:v1,v2…
        int op2;            //運算元
        int result;         //欲儲存的運算元 eg:r4,r5…
        }sample1[Size];
```

eg:　1 2 3 8　　// v8=v2+v3
　op=1　　op2=2　　op2=3　　result=8

將 sample1 依照規則排入 readylist，排進去之後，需要紀錄以下資料。

```
struct readylist{
        int state;          //第幾個 state
        int op;             //運算元
        int num;            //對應 sample 的第 i 個位置
        int dis;            //距離最後一個運算幾個距離
        }list1[Size],temp;
```

eg: 將 1 2 3 8 設定為第一個要擺的。
state=1　　op=1　　num=i

```
struct alu{
        int mult;
        int add;
        }alulist[Size];
```

用來判定 resource constraint 的作用。紀錄每個 state 用了幾個乘法器或加法器。在執行初期需要做初始化的動作，設定所有的 state 都有足夠的 mult 跟 add 可用。

eg:

限制乘法器為 2 個 加法器為 1 個

.[i]mult=1 .[i]add=2 from(i =1 to i=n)

程式結構：

含有兩個副程式 readIn()與 list_Scheduling()。

readIn() //從.txt 檔讀取檔案，並且寫入。

list_Scheduling() //完成整個演算法，依照順序從第一個運算式開始排程。

1. alulist 初始化

2. 開始將每個運算式排入。

- 是否有發生資料相依，如果有則需排入下一個 state。

eg:

A v1=v2+v3

B v4=v1*v1

此例中 B 運算式需要等 A 運算式算完才可排入。

- 是否有乘法器或加法器在該 state 已經用完。

eg:

Resource constrain add=2

A v1=v2+v3 state:1

B v4=v5+v6 state:1

C v7=v8+v9

此例中 C 需要排入 state2。

- 檢查 op1 與 op2 兩個運算元，分別是在那一個 state 被計算出來的。紀錄在 int opstate。

eg:

A v1=v2+v3 state:1

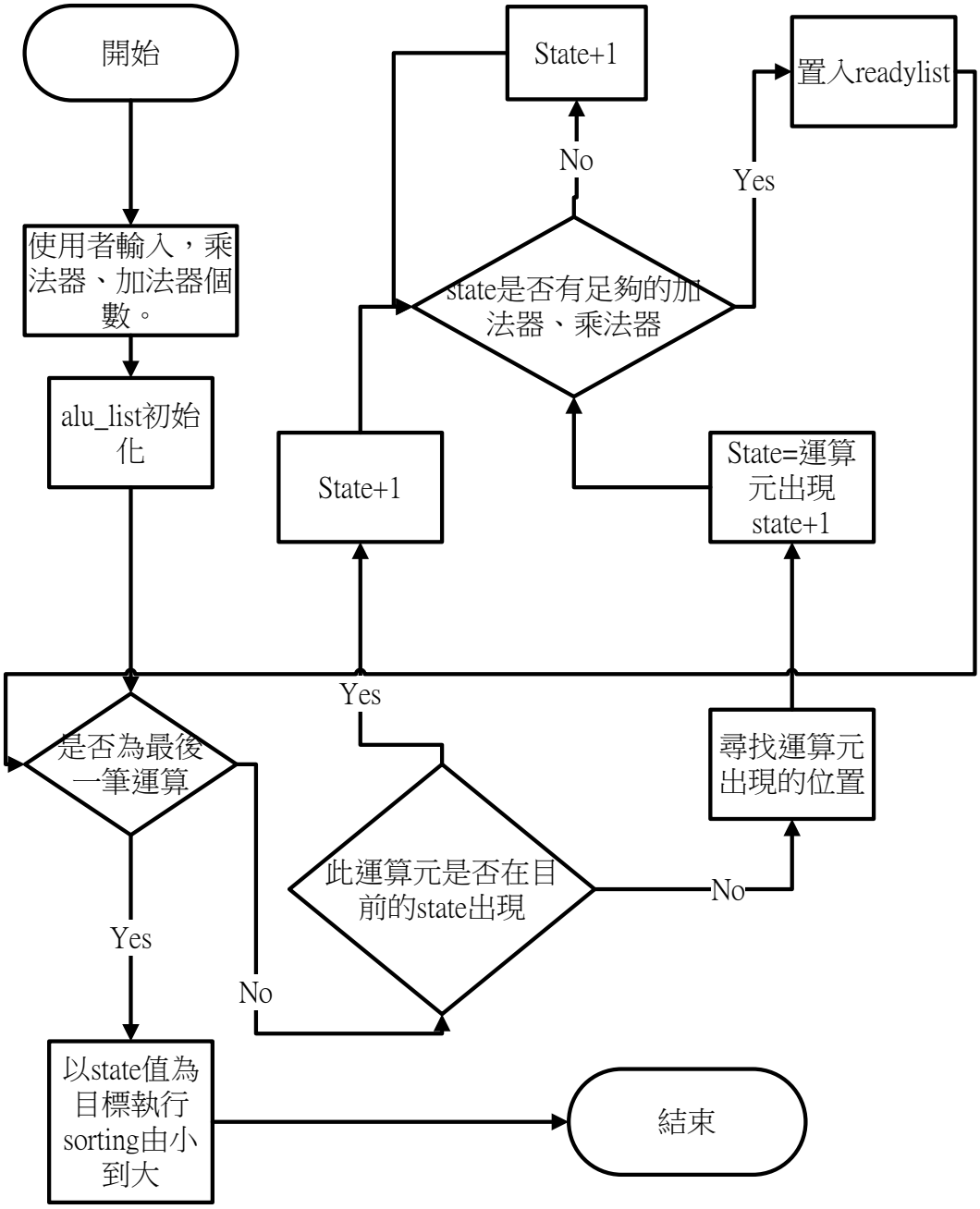
…

B v4=v1+v1 state:5

C v5=v1+v4

此例中 v1 在 state1 結束後算出，v4 在 state5 結束後算出。因此 v1 之 opstate 為 state1，v4 之 opstate 為 state5。由此可知 C 運算式必須等到 v4 做完才可置入排程。

3. 依照 state 先後順序由小到大做 sorting。



Output 方式：

產生一個 Scheduling_outcome.txt，裡面紀錄了每次運算的結果。

eg:

```
*******************************
*       Resource Constraint           *
*                                     *
*        Mult Constraint:2            *
*        Add Constraint:2             *
*                                     *
*******************************
```

State: 1 v 10 = v 1 + v 2
State: 1 v 11 = v 6 + v 7
State: 2 v 12 = v 10 + v 3
State: 2 v 13 = v 11 + v 8
State: 3 v 14 = v 12 + v 4
State: 3 v 15 = v 13 + v 9
State: 4 v 16 = v 14 + v 15
State: 5 v 17 = v 16 * v 16
State: 5 v 18 = v 16 * v 16
State: 7 v 19 = v 17 + v 12
State: 7 v 23 = v 18 + v 15
State: 8 v 22 = v 19 + v 16
State: 8 v 20 = v 12 + v 19
State: 9 v 21 = v 20 * v 20
State: 9 v 24 = v 22 + v 23
State: 9 v 25 = v 23 + v 15
State: 10 v 27 = v 25 * v 25
State: 11 v 26 = v 10 + v 21
State: 12 v 28 = v 10 + v 26
State: 12 v 29 = v 26 + v 19
State: 13 v 30 = v 28 * v 28
State: 13 v 31 = v 29 + v 5
State: 13 v 32 = v 27 + v 9
State: 14 v 33 = v 31 * v 31
State: 14 v 34 = v 23 + v 32
State: 14 v 35 = v 32 + v 9
State: 15 v 36 = v 1 + v 30
State: 15 v 37 = v 11 + v 34
State: 15 v 38 = v 35 * v 35
State: 16 v 39 = v 36 + v 26
State: 16 v 40 = v 33 + v 5
State: 16 v 41 = v 37 * v 37
State: 17 v 42 = v 31 + v 40
State: 17 v 43 = v 32 + v 38

使用說明：

1. 選擇要載入哪一組 Data Flow Graph。欲選擇 p1 則輸入 p1.txt，選擇 p2 則輸入 p2.txt。
2. 輸入 resource constraint 乘法器跟加法器數目。
3. 程式會將排程結果顯示在銀幕上，同時也會寫入 Scheduling_outcome.txt。
4. 如果需要其他數目的 resource constraint 則輸入 1 繼續，輸入 2 結束。