

```
1 // 執行程式: Ctrl + F5 或 [偵錯] > [啟動但不偵錯] 功能表
2 // 偵錯程式: F5 或 [偵錯] > [啟動偵錯] 功能表
3 // 開始使用的提示:
4 // 1. 使用 [方案總管] 視窗, 新增/管理檔案
5 // 2. 使用 [Team Explorer] 視窗, 連線到原始檔控制
6 // 3. 使用 [輸出] 視窗, 參閱組建輸出與其他訊息
7 // 4. 使用 [錯誤清單] 視窗, 檢視錯誤
8 // 5. 前往 [專案] > [新增項目], 建立新的程式碼檔案, 或是前往 [專案] > [新增現有項
    目], 將現有程式碼檔案新增至專案
9 // 6. 之後要再次開啟此專案時, 請前往 [檔案] > [開啟] > [專案], 然後選取 .sln 檔案
10
11 // HWK1.cpp : 此檔案包含 'main' 函式。程式會於該處開始執行及結束執行。
12 //作者:YUDA LU
13 //日期:2020.03.20(1.0), 2020.03.24(1.1), 2020.04.09(1.2), 2020.04.10(2.0)
14
15 //目前 >> back_edge 錯誤運算 (low錯誤)
16
17 #include <iostream>
18 #include <fstream>
19 #include <windows.h>
20 #include <dos.h>
21 #include <conio.h>
22 #include <typeinfo>
23 #define SIZE 20
24 using namespace std;
25
26 class point
27 {
28 private:
29     //number >> 表格數字
30     //number_line[20] >> 數字連接到的數字
31     //line_n >> 數字連接到總數
32
33     //老師提點 >> 可以在分支點上座分支紀錄 >> 帶入遞迴函數座判斷
34
35     int number, number_line[SIZE], line_n,
36         dfn = -1, back_edge, low, input_number;
37     bool bool_number_line[SIZE] = { false },
38         bool_number_end = false;
39
40 public:
41     //導入檔案
42     void in_number(int n);
43     void in_number_line(int i, int n);
44     void in_line_n(int n);
45     void in_bool_number_line(int i, bool op);
46     void in_dfn(int i);
47     void in_low(int i);
48     void in_back_edge(int i);
49     void in_bool_number_end(bool i);
50     void in_input_number(int i);
51
52     //導出檔案
```

```

53     int out_number();
54     int out_number_line(int i);
55     int out_line_n();
56     int out_dfn();
57     int out_low();
58     int out_back_edge();
59     int out_bool_total();
60     int out_input_number();
61     bool out_bool_number_line(int i);
62     bool out_bool_number_end();
63
64     //function at class point
65     int return_number_i(int n);
66     bool cheak_number(int number);
67     bool cheak_number_ok(int number);
68 };
69
70 void read_date(string* in_number_date);
71 void string_to_point_array(string str_number, point point_array[SIZE], int* n);
72
73 void print_number_array(int n, point point_array[SIZE]);
74 void print_numbber_array_line(int n, point point_array[SIZE]);
75 void print_bool_array(int n, point point_array[SIZE]);
76 void print_articulation_point_array(int articulation_point[SIZE], int          ↗
    articulation_point_n);
77 void print_dfs_end(int n, point point_array[SIZE]);
78 void print_all_date(int n, point point_array[SIZE]);
79
80 void insert_number_to_point_array(int n, point point_array[SIZE]);
81 void insert_dfs(int n, point point_array[SIZE], int dfs_array[SIZE]);
82 void insert_dfs2(int n, point point_array[SIZE], int dfs_array_n, int dfs_array    ↗
    [SIZE]);
83 void insert_back_edge(int n, point point_array[SIZE], int dfs_array[SIZE]);
84 void insert_articulation_point(int n, point point_array[SIZE], int dfs_array[SIZE], ↗
    int articulation_point[SIZE], int* articulation_point_n);
85
86 int search_next_number(int search_number, point point_array[SIZE], int dfs_array_n, ↗
    int dfs_array[SIZE]);
87 bool ok_number(int next_number, int dfs_array_n, int dfs_array[SIZE]);
88
89 //主程式
90 int main(void)
91 {
92     /*
93         名詞
94         articulation_point >> 關節點
95         變數
96         n                總共有多少數字
97         dfs_array        DFS找到的順序
98         in_number_date   讀入黨案驚寫入字串中
99         point_array      數字陣列
100        articulation_point 關節點陣列
101        articulation_point_n 關節點陣列n

```

```
102     思考路線
103         1.讀黨
104         2.將檔案轉換成Point形式
105         3.使用DFS找出DFN陣列
106         4.使用back_edge找出low
107         5.使用low & dfn 找出關節點
108     */
109
110     int n, articulation_point_n, dfs_array[SIZE], articulation_point[SIZE];
111     string in_number_date;
112     point point_array[SIZE];
113
114     read_date(&in_number_date); //讀黨
115     string_to_point_array(in_number_date, point_array, &n); //轉換檔案中資料
116
117     print_number_array(n, point_array);
118     print_numbber_array_line(n, point_array);
119
120     //建置point_array中number數字
121     insert_number_to_point_array(n, point_array);
122
123     //算出dfs
124     insert_dfs(n, point_array, dfs_array);
125     insert_dfs2(n, point_array, n, dfs_array);
126
127     print_bool_array(n, point_array);
128     print_numbber_array_line(n, point_array);
129
130     //用back_edge算出low
131     insert_back_edge(n, point_array, dfs_array);
132
133     //用low and dfn 算出關節點
134     insert_articulation_point(n, point_array, dfs_array, articulation_point, //列 ↗
        &articulation_point_n);
135
136     //列印檔案
137     print_bool_array(n, point_array);
138     print_all_date(n, point_array);
139     print_articulation_point_array(articulation_point, articulation_point_n); //列 ↗
        印關節點
140
141     system("pause");
142     return 0;
143 }
144
145 //class point 程式區段 開始
146 void point::in_number(int n)
147 {
148     number = n;
149 }
150
151 void point::in_number_line(int i, int n)
152 {
```

```
153     number_line[i] = n;
154 }
155
156 void point::in_line_n(int n)
157 {
158     line_n = n;
159 }
160
161 void point::in_bool_number_line(int i, bool op)
162 {
163     bool_number_line[i] = op;
164 }
165
166 void point::in_dfn(int i)
167 {
168     dfn = i;
169 }
170
171 void point::in_low(int i)
172 {
173     low = i;
174 }
175
176 void point::in_back_edge(int i)
177 {
178     back_edge = i;
179 }
180
181 void point::in_bool_number_end(bool i)
182 {
183     bool_number_end = i;
184 }
185
186 void point::in_input_number(int i)
187 {
188     input_number = i;
189 }
190
191 int point::out_number()
192 {
193     return number;
194 }
195
196 int point::out_number_line(int i)
197 {
198     return number_line[i];
199 }
200
201 int point::out_line_n()
202 {
203     return line_n;
204 }
205
```

```
206 int point::out_dfn()
207 {
208     return dfn;
209 }
210
211 int point::out_low()
212 {
213     return low;
214 }
215
216 int point::out_back_edge()
217 {
218     return back_edge;
219 }
220
221 int point::out_bool_total()
222 {
223     int sum = 0;
224     for (int i = 0; i < line_n; i++)
225     {
226         if (bool_number_line[i] == true)
227         {
228             ++sum;
229         }
230     }
231     return sum;
232 }
233
234 int point::out_input_number()
235 {
236     return input_number;
237 }
238
239 bool point::out_bool_number_line(int i)
240 {
241     return bool_number_line[i];
242 }
243
244 bool point::out_bool_number_end()
245 {
246     return bool_number_end;
247 }
248
249 int point::return_number_i(int n)
250 {
251     int out = 0;
252     for (int i = 0; i < line_n; i++)
253     {
254         if (number_line[i] == n)
255         {
256             out = i;
257             break;
258         }
259     }
```

```
259     }
260     return out;
261 }
262
263 bool point::cheak_number(int number)
264 {
265     bool op = false;
266     for (int i = 0; i < line_n; i++)
267     {
268         if (number_line[i] == number) op = true;
269     }
270     if (op) return true;
271     else return false;
272 }
273
274 bool point::cheak_number_ok(int number)
275 {
276     bool op = false;
277     for (int i = 0; i < line_n; i++)
278     {
279         if (number_line[i] == number && bool_number_line[i] == 0) op = true;
280     }
281     if (op == true) return true;
282     else return false;
283 }
284 //class point 程式區段 結束
285
286 //開檔讀檔
287 void read_date(string* in_number_date)
288 {
289     fstream InF;
290     int n = 0;
291     char FName[20], ch;
292
293     cout << "輸入方程式檔名:";
294     cin >> FName;
295     InF.open(FName, ios::in);
296
297     if (!InF)
298     {
299         cout << "檔案無法開啟\n";
300         exit(1);
301     }
302     else
303     {
304         while (InF.get(ch))
305         {
306             *in_number_date += ch;
307         }
308         InF.close();
309     }
310 }
311
```

```
312 //string to point_array
313
314 void string_to_point_array(string str_number, point point_array[SIZE], int* n)
315 {
316     int i = 0, number_i = 0, number_n = 0, number_line_n = 0;
317     bool get_n = false;
318     string t;
319
320     /*
321     t >> 站存n字串
322     get_n >> 判讀n讀完了嗎?
323     number_i >> 紀錄number_i 的 i
324     number_n >> 紀錄number_i
325     number_line_n >> point_number_line 長度紀錄
326     */
327
328     while (str_number[i])
329     {
330         if (!get_n)
331         {
332             //讀入n
333
334             if (str_number[i] != '\n')
335             {
336                 t += str_number[i];
337             }
338             else
339             {
340                 *n = atoi(t.c_str());
341                 get_n = true;
342             }
343             i++;
344         }
345         else
346         {
347             if (str_number[i] != '\n')
348             {
349                 //如果有1才紀錄至資料中
350
351                 if (str_number[i] == '1')
352                 {
353                     point_array[number_n].in_number_line(number_line_n, number_i);
354
355                     number_i++;
356                     number_line_n++;
357                 }
358                 else if (str_number[i] == '0')
359                 {
360                     number_i++;
361                 }
362                 else if (str_number[i] == ' ');
363                 else
364                 {
```

```
365         exit(1);
366     }
367 }
368 else
369 {
370     //紀錄當行訊息並歸零
371
372     point_array[number_n].in_line_n(number_line_n);
373     number_line_n = 0;
374     number_i = 0;
375     number_n++;
376 }
377 i++;
378 }
379 }
380
381 //防止字串最後無換行而無紀錄最後一行訊息
382
383 if (str_number[i - 1] != '\n')
384 {
385     point_array[number_n].in_line_n(number_line_n);
386 }
387 }
388
389 void print_articulation_point_array(int articulation_point[SIZE], int articulation_point_n)
390 {
391     cout << "Articulation Point(關節點): ";
392     for (int i = 0; i < articulation_point_n; i++)
393     {
394         cout << articulation_point[i] << " ";
395     }
396     cout << "\n完成運算結束囉!!\n";
397 }
398
399 void print_dfs_end(int n, point point_array[SIZE])
400 {
401     for (int i = 0; i < n; i++)
402     {
403         cout << point_array[i].out_bool_number_end() << " ";
404     }
405     cout << endl;
406 }
407
408 void print_number_array(int n, point point_array[SIZE])
409 {
410     cout << "n = " << n << " 矩陣: " << endl;
411     printf("    ");
412     for (int i = 0; i < n; i++) printf("%2d ", i);
413     cout << endl;
414     for (int i = 0; i < n; i++)
415     {
416         printf("%2d > ", i);
```



```
417     for (int j = 0; j < n; j++)
418     {
419         if (point_array[i].cheak_number(j)) cout << "1 ";
420         else cout << "0 ";
421     }
422     cout << endl;
423 }
424 cout << "\n";
425 }
426
427 void print_all_date(int n, point point_array[SIZE])
428 {
429     cout << "Print all date" << endl;
430     for (int i = 0; i < n; i++)
431     {
432         printf("number = %2d, dfs_number = %2d, low = %2d, input_number = %2d\n", i, ↗
               point_array[i].out_dfn(), point_array[i].out_low(), point_array ↗
               [i].out_input_number());
433     }
434     cout << endl;
435 }
436
437 void print_bool_array(int n, point point_array[SIZE])
438 {
439     cout << "bool 連接結果狀況\n";
440     for (int i = 0; i < n; i++)
441     {
442         printf("%2d > ", i);
443         for (int j = 0; j < point_array[i].out_line_n(); j++)
444         {
445             cout << point_array[i].out_bool_number_line(j) << " ";
446         }
447         cout << " \ttotal >> " << point_array[i].out_bool_total() << endl;
448     }
449     cout << "\n";
450 }
451
452 void print_numbbber_array_line(int n, point point_array[SIZE])
453 {
454     cout << "簡單矩陣：\n";
455     for (int i = 0; i < n; i++)
456     {
457         printf("%2d > ", i);
458         for (int j = 0; j < n; j++)
459         {
460             if (point_array[i].cheak_number(j)) cout << j << " ";
461         }
462         cout << endl;
463     }
464     cout << "\n";
465 }
466
467 void insert_number_to_point_array(int n, point point_array[SIZE])
```

```
468 {
469     for (int i = 0; i < n; i++)
470     {
471         point_array[i].in_number(i);
472     }
473 }
474
475 bool ok_number(int next_number, int dfs_array_n, int dfs_array[SIZE])
476 {
477     bool op = true;
478     for (int i = 0; i < dfs_array_n; i++)
479     {
480         if (dfs_array[i] == next_number)
481         {
482             op = false;
483             //break;
484         }
485     }
486     return op;
487 }
488
489 int search_next_number(int search_number, point point_array[SIZE], int dfs_array_n,
490     int dfs_array[SIZE])
491 {
492     int next_number;
493     /*
494     尋找下一個點
495     確認search_number的連接有沒有下一個點
496     如果有找到點，return 點
497     如果沒找到點，return NULL
498     */
499     for (int i = 0; i < point_array[search_number].out_line_n(); i++)
500     {
501         next_number = point_array[search_number].out_number_line(i);
502
503         if (point_array[search_number].out_bool_number_line(i) == false && ok_number
504             (next_number, dfs_array_n, dfs_array) == true)
505         {
506             int get_number = point_array[search_number].out_number_line(i);
507             int k = point_array[get_number].return_number_i(search_number);
508
509             point_array[search_number].in_bool_number_line(i, true);
510             point_array[get_number].in_bool_number_line(k, true);
511
512             return next_number;
513             break;
514         }
515     }
516     return -1;
517 }
518
519 void insert_dfs(int n, point point_array[SIZE], int dfs_array[SIZE])
520 {
```

```
519     //cursor 目前收尋位置
520
521     int search_total = 0, next_number, cursor = 0, dfs_array_n = 0;
522     bool op = false, go_break = false;
523     while (cursor < n && cursor >= 0)
524     {
525         //利用dfs_array的n判斷是否全部點已經找到
526         if (search_total == n) break;
527         if (op == false)
528         {
529             //將起始值輸入dfs_array
530             point_array[cursor].in_dfn(search_total);
531             point_array[cursor].in_input_number(0);
532
533             dfs_array[dfs_array_n] = point_array[cursor].out_number();
534
535             dfs_array_n++, search_total++, op = true;
536         }
537         else if (op == true)
538         {
539             //search_next_number >> 尋找下一個數字 沒找到回傳NULL
540             next_number = search_next_number(cursor, point_array, search_total,
541             dfs_array);
542             if (next_number != -1)
543             {
544                 if (ok_number(cursor, search_total, dfs_array) == true)
545                 {
546                     point_array[cursor].in_dfn(search_total);
547                     dfs_array[search_total++] = cursor;
548
549                     point_array[next_number].in_input_number(cursor);
550                     cursor = next_number;
551                     dfs_array_n = search_total;
552                     go_break = true;
553                 }
554                 else if (next_number == -1)
555                 {
556                     if (go_break == true)
557                     {
558                         dfs_array[search_total++] = cursor;
559                         point_array[cursor].in_bool_number_end(true);
560                         go_break = false;
561                     }
562                     cursor = dfs_array[--dfs_array_n];
563                 }
564                 else
565                 {
566                     cout << "GG!!" << endl;
567                     exit(1);
568                 }
569             }
570         }
```

```
571 }
572
573 void insert_dfs2(int n, point point_array[SIZE], int dfs_array_n, int dfs_array
    [SIZE])
574 {
575     for (int i = 0; i < n; i++) point_array[i].in_dfn(dfs_array[i]);
576 }
577
578 //back_edge >> 目前 low 錯誤運算
579 void insert_back_edge(int n, point point_array[SIZE], int dfs_array[SIZE])
580 {
581     /*
582         hold_number 目前最小的路徑
583         find_number 找到的最短路徑
584         next_op      控制有沒有找到新的(find_number)最短路進
585     */
586
587     int hold_number = n - 1, find_number = n - 1;
588     bool next_op = false;
589
590     // i >> dfn
591     for (int i = n - 1; i >= 0; i--)
592     {
593         for (int k = 0; k < i; k++)
594         {
595             //利用dfs_array最後一個找回來
596             if (point_array[dfs_array[i]].cheak_number_ok(dfs_array[k]) == true)
597             {
598                 int a_line_i = 0, b_line_i = 0;
599                 // a b 代提連接變數 a接b
600
601                 //跟改bool
602                 a_line_i = point_array[dfs_array[i]].return_number_i(dfs_array[k]);
603                 b_line_i = point_array[dfs_array[k]].return_number_i(dfs_array[i]);
604
605                 point_array[dfs_array[i]].in_bool_number_line(a_line_i, true);
606                 point_array[dfs_array[k]].in_bool_number_line(b_line_i, true);
607
608                 find_number = k;
609                 next_op = true;
610                 break;
611             }
612             next_op = false;
613             //find_number = -1;
614         }
615     }
616
617     point_array[0].in_low(0);
618
619     if (find_number < hold_number && next_op == true) //當找到其他
        最短路徑 比上一個小就跟新並存入
620     {
621         hold_number = find_number;
622         point_array[dfs_array[i]].in_low(hold_number);
623     }
624 }
```

```

620         next_op = false;
621     }
622     else if (find_number >= hold_number && next_op == true)           //找到的路徑 ➤
        太沒有比上一短不跟新
623     {
624         point_array[dfs_array[i]].in_low(hold_number);
625         next_op = false;
626     }
627     else if (i <= n - 2 && point_array[dfs_array[i + 1]].out_low() < i && next_op ➤
        == false)                                           //沒找到路徑直接紀錄
628     {
629         find_number = i;
630         point_array[dfs_array[i]].in_low(point_array[dfs_array[i + 1]].out_low ➤
            ());
631     }
632     else if(next_op == false)
633     {
634         find_number = i;
635         point_array[dfs_array[i]].in_low(i);
636     }
637     else
638     {
639         point_array[dfs_array[i]].in_low(i);
640     }
641
642
643     //當回到樹根 hold_number 跟著回到最大值
644     if (point_array[dfs_array[i]].out_number() == 0 || find_number == 0 && i != ➤
        0)
645     {
646         hold_number = n - 1, find_number = n - 1;
647     }
648 }
649
650 //強制讓樹根變成0
651 point_array[0].in_low(0);
652 }
653
654 //articulation_point >> 關節點
655 void insert_articulation_point(int n, point point_array[SIZE], int dfs_array[SIZE], ➤
    int articulation_point[SIZE], int* articulation_point_n)
656 {
657     int out_n = 0;
658
659     for (int dfn = 0; dfn < n; dfn++)
660     {
661         if (dfn == 0)
662         {
663             if (point_array[dfs_array[dfn]].out_bool_total() > 2)           ➤
                //判斷樹根
664             {
665                 articulation_point[out_n++] = point_array[dfs_array[dfn]].out_number ➤
                    ();

```

```
666         }
667     }
668     else
669     {
670         int a = point_array[dfs_array[dfn]].out_input_number(), b = dfs_array    ↗
            [dfn];
671         if (point_array[b].out_low() >= point_array[a].out_dfn() && a != 0)
672         {
673             articulation_point[out_n++] = point_array[a].out_number();    //利 ↗
            用公式找出關節點
674         }
675     }
676 }
677 *articulation_point_n = out_n;
678 }
```