# Homework 2: Two-way Min-cut Partitioning

**107062612 熊祖玲**

(1) How to compile and execute your program, and give an execution example. If you implement parallelization, please let me know how to execute it with single thread.

**在 HW2/src/ 的路徑下，執行make指令，而程式執行檔 main 會產生在 HW2/bin 中，移動到 HW2/bin 後，即可執行。**
**第一個參數：.cells file path**
**第二個參數：.nets file path**
**第三個參數：.out file path**
**例如：./main p2-1.cells p2-1.nets p2-1.out**

(2) The final cut size and the runtime of each testcase.

| | Cutsize | Runtime |
|---|---|---|
| **p2-1** | 25 | 0.027 |
| **p2-2** | 174 | 0.7 |
| **p2-3** | 1866 | 37.974 |
| **p2-4** | 39854 | 72.54 |

(3) Runtime = $TIO + Tcomputation$. For each case, please analyze your runtime and find out how much time you spent on I/O and how much time you spent on the computation (FM Algorithm).

| | $T_{IO}$ | $T_{computation}$ |
|---|---|---|
| **p2-1** | 0.007 | 0.02 |
| **p2-2** | 0.023 | 0.7 |
| **p2-3** | 0.854 | 37.12 |
| **p2-4** | 1.12 | 72.54 |

**IO的時間都比計算時間來的少，但兩者皆會隨著資料量而增加，因為我的程式當中在做每個iteration的時候，因為要找到gain最大的cell，在worse case中會掃過每一個pin，所以每個iteration的時間複雜度為$O(P)$，而最多做N個iteration，則時間複雜度為$O(NP)$，當net的數量遠大於cell的數量時，時間會有很大幅的成長趨勢。**

(4) The details of your implementation containing explanations of the following questions:

I. Where is the difference between your algorithm and FM Algorithm described in class? Are they exactly the same?

**課堂上所提及的FM algorithm會不斷地做，直到partial sum不為正數為止，這麼做可能會花費大量的時間，卻得到很小的改變量，因此我設定一個終止條件：**

$$1 - \frac{cutsize_n}{cutsize_{n-1}} < 0.001$$

**$cutsize_n$為當前iteration的cutsize，$cutsize_{n-1}$為上一個iteration的cutsize。若改變量小於0.001時，表示下一個iteration改善幅度不會很大，即終止程式。**

II. Did you implement the bucket list data structure?
    I. If so, is it exactly the same as that described in the slide? How many are they?
    II. If not, why? You had a better data structure? Or, is bucket list useless?

**我在儲存cell的資料結構當中設定兩個指標，一個指向同gain的前一個cell，一個指向同gain的下一個cell，再另外用二個array存這些double linked lists，則可以用gain當index一一搜尋這些double linked lists，當其中一個cell要在double linked list中插入或移除時，直接更改cell的指標即可，不用在掃過整個double linked list。**

III. How did you find the maximum partial sum and restore the result?

**在每個iteration儲存要被換的cell、紀錄被換的cell目前的partial sum且與maximum partial sum做比較，如果目前的partial sum大於maximum partial sum，則紀錄是在第幾次的iteration發生的 (i)，在所有的iteration結束後，將儲存要被換的cell中取前 i 個做移動。**

IV. Please compare your results with the top 5 students' results from last year and show your advantage either in runtime or in solution quality. Are your results better than them?
    I. If so, please express your advantages to beat them.
    II. If not, it's fine. If your program is too slow, then what do you reckon the bottleneck of your program is? If your solution quality is inferior, what do you think that you could do to improve the result in the future?

**在第一、二筆測資中，我的runtime只比去年第四名的人快，第三筆則贏給第二名和第四名，第四筆只輸給第一、五名。對於cutsize來說，我在第二、四筆贏過所有人，第三筆贏第二、五名，而第一筆輸給所有人。我想可能是因為我都用指標做運算、傳遞而加速，且設定終止條件，讓FM algorithm的改善程度太小時，不會再繼續執行，因此在時間上會有比較好的結果。但在測資比較小的時候cutsize會有比較差的結果。**

V. What else did you do to enhance your solution quality or to speed up your program?

**我發現如果做很多次完整的FM algorithm，runtime不但會增加，cutsize也會增加，結果並沒有如預期的好，因此，我選擇只做一次完整的FM algorithm，所以runtime也很低。**

VI. What have you learned from this homework? What problem(s) have you encountered in this homework?

**以前可能對指標的實作不是很靈活，在這次作業當中，有很扎實的訓練，又因為測試資料都很龐大，因此會在意程式的效能，也深刻體會到指標對程式速度的影響真的很大，而對於debug時的敏感度也大幅提升，可以比較快地找出問題點。此外，Makefile在編譯時真的十分方便，雖然之前有學過一點Makefile的撰寫，但沒有很熟悉，經過這次作業的洗禮後，有比較熟悉一些了。**

VII. If you implement parallelization, please describe the implementation details and provide some experimental results.

**None.**