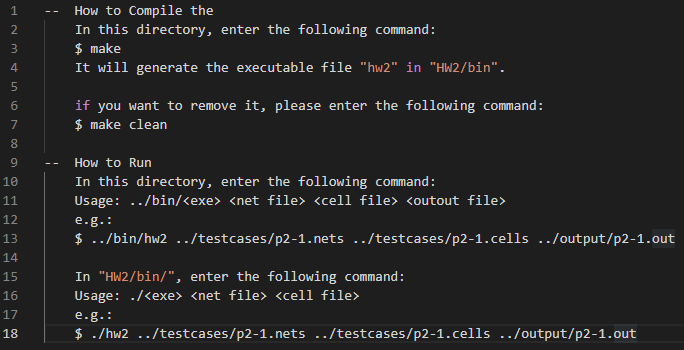
**Homework 2: Two-way Min-cut Partitioning**

**109062556 李濬安**

* **How to compile and execute your program?**

可以根據以下資料來去compile & Run. (src/README)



* **The final cut size and the runtime of each testcase**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Testcase | P2-1 | P2-2 | P2-3 | P2-4 | P2-5 |
| Cut size | 6 | 219 | 3596 | 47086 | 130591 |
| Runtime | 0.012 | 0.141 | 6.897 | 14.356 | 29.818 |

* **Analyze your runtime. (Runtime = TIO + Tcomputation.)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Testcase | P2-1 | P2-2 | P2-3 | P2-4 | P2-5 |
| TIO (s) | 0.002493 | 0.026437 | 0.489314 | 0.942929 | 3.39629 |
| Tcomputation(s) | 0.011222 | 0.116757 | 6.5239 | 13.6071 | 26.7455 |
| Runtime(s) | 0.012 | 0.141 | 6.897 | 14.356 | 29.818 |

可以發現到，大部份時間主要是花在FM algorithm上。但我的FM algorithm時間會比我上述的Tcomputation來的短，因為在我的方法中，我執行了17次FM algoritm，所以計算一次FM時間大概分別是{ 0.004, 0.027, 3.59, 8.47, 21.12(s)}

* **Question&Answer**

1. Where is the difference between your algorithm and FM Algorithm?

實作的FM algorithm和課本的一樣。

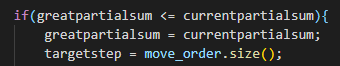
1. Did you implement the bucket list data structure?

有，但和課本的略為不同!

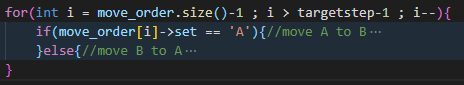
不同處是，我沒有分bucketlistA, bucketlistB，我是統一存到同一個bucketlist。目的為，在實作時發現當要選擇gain最大的那array裡的cell來當作移動的cell時，會需要先對bucketlistA 該gain值裡的cell去看看是否移動後會balance，但如果沒有可移動的cell，就換檢查bucketlistB該gain裡的cell，那如果通通都沒有，又回到bucketlistA找第二大的gain的cell，根據上述我發現如果存在同一個bucketlist，然後只要有存cell目前是在set A還是set B就好了，這樣不僅可以省掉很多的比對，也較容易去debug。

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

對於max partial sum會有一變數currentpartialsum紀錄是否加了現在移動cell的gain值會比之前的greatpartialsum還來的大，如果比較大的話，更新greatpartialsum，並且記錄該移動cell是第幾步。



對於restore the result，會有一vector存move order，那根據上述的targetstep，可以找到說從targetstep後面移動的cell是多餘的，因此只要把那些step所移動的cell移回來就行。



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

對於Speed:

* 有設一個countdown來倒數做了幾次move cell後，greatpartialsum並沒有變好，就直接early stop。
* 在同一個function裡，對於常只用的指標會統一設一個指標變數來代表他，這樣就不用每次寫到他的時候，要再去跑一次所指的位置的值。

e.g.: 

對於Quality:

* 我發現initial partition對於結果是相當影響的，因此我是根據net所連接的cell來先將她們分set，意思就是同一條net的cell在initial partition時會先盡量讓他們同一個set，這對我的cut size有一定程度的影響。
* 而根據上述的道理，我加入了另一個想法是將跑完FM algorithm的分群結果，再去做一次FM algorithm，這樣是因為經由前一次FM algorithm出來的data cut size有一定的減少，而將此data再當作下一次的initial partition，重複執行17次，發現cut size有顯著再下降。

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

未實作平行化

* **Please compare your results with the top 5 students’ results from last year.**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | cutsize | | | | | runtime | | | | | |
| Ranks | **p2-1** | **p2-2** | **p2-3** | **p2-4** | **p2-5** | **p2-1** | **p2-2** | **p2-3** | **p2-4** | **p2-5** |
| 1 | 6 | 191 | 4441 | 43326 | 122101 | 0.01 | 0.07 | 3.05 | 5.01 | 42.06 |
| 2 | 6 | 161 | 1065 | 43748 | 125059 | 0.01 | 0.1 | 3.11 | 9.84 | 18.77 |
| 3 | 6 | 358 | 2186 | 45430 | 122873 | 0.04 | 0.78 | 21.21 | 115.38 | 59.78 |
| 4 | 5 | 302 | 1988 | 46064 | 124862 | 0.03 | 0.17 | 7.04 | 6.93 | 8.22 |
| 5 | 6 | 411 | 779 | 46356 | 125151 | 0.01 | 0.16 | 5.49 | 12.31 | 13.27 |
| mine | 6 | 219 | 3596 | 47086 | 130591 | 0.01 | 0.14 | 6.89 | 14.35 | 29.81 |

Cutsize:

* P2-1跟之前一樣小
* P2-2比之前一些人還小
* P2-3只有比第rank 1的還小
* P2-4, P2-5都比之前的大，但沒有大太多

Runtime:

* P2-1跟之前一樣快
* P2-2比之前一些人還快
* P2-3跟大多數人一樣快
* P2-4跟大多數人一樣快
* P2-5比rank1, rank5還快
* **What have you learned from this homework? What problem(s) have you encountered in this homework?**

在此次作業，實作了Two-way Min-cut Partitioning，學到相當多實作上的經驗，包括資料結構多變的應用，抑或是演算法的完整度，也花很大的心力去完成這項作業並盡力去優化，是相當充實的!

需要進步的還很多，重要的是對於cut size。雖然在runtime上表現算是很好的，但是最重要的是演算法對於cut size的結果並不是很優的，我想可以再去思考如何讓cut size降低，這是我需要再去survey的。