CS6135_HW2_110062619_report

(1)Your name and student ID

Name: 楊淨富

Student ID: 110062619

(2)How to compile and execute your program, and give an execution example.

- How to Compile

In /HW2/src directory, enter the following command:

> 💡 $make

> 💡 It will generate the executable files "main" in "HW2/bin/".

- How to execute

In /HW2/src directory, enter the following command:

> 💡 Usage: ../bin/<exe> <net file> <cell file> <output file>

```
e.g.:
  $ ../bin/hw2 ../testcases/p2-1.nets ../testcases/p2-1.cells ../output/p2-1.txt
```

- How to verify

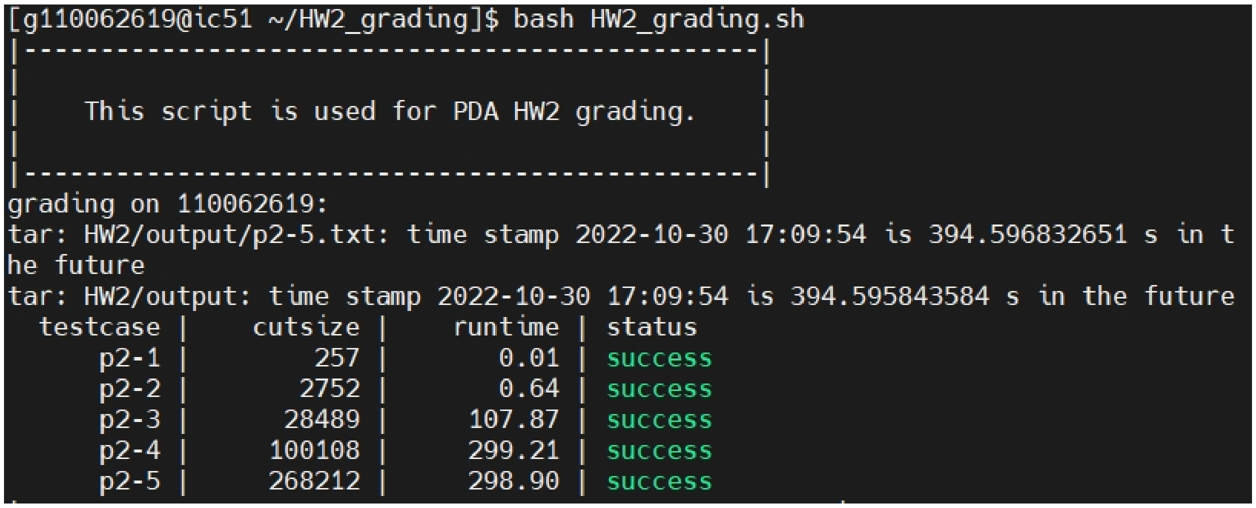
In /HW2/src directory, enter the following command:

> 💡 Usage: ../verifier/Verify <net file> <cell file> <output file>

```
e.g.:
  $ ../verifier/Verify ../testcases/p2-1.nets ../testcases/p2-1.cells ../output/p2-1.txt
```

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

Cases	p2-1	p2-2	p2-3	p2-4	p2-5
Cut size	257	2752	28489	100108	268212
Runtime(s)	0.01	0.64	107.87	299.21	298.9



```
[g110062619@ic51 ~/HW2_grading]$ bash HW2_grading.sh
|------------------------------------------------|
|                                                |
|    This script is used for PDA HW2 grading.    |
|                                                |
|------------------------------------------------|
grading on 110062619:
tar: HW2/output/p2-5.txt: time stamp 2022-10-30 17:09:54 is 394.596832651 s in t
he future
tar: HW2/output: time stamp 2022-10-30 17:09:54 is 394.595843584 s in the future
  testcase |    cutsize |    runtime | status
      p2-1 |        257 |       0.01 | success
      p2-2 |       2752 |       0.64 | success
      p2-3 |      28489 |     107.87 | success
      p2-4 |     100108 |     299.21 | success
      p2-5 |     268212 |     298.90 | success
```

(4)Runtime = $T\_IO + T\_computation$. 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).

Cases	p2-1	p2-2	p2-3	p2-4	p2-5
I/O time(s)	0.005	0.05	0.37	1.59	3.64
CPU time(s)	0.01	0.59	107.5	297.62	295.26

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

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

💡 最主要的差別在於我的每個pass並不會每次都做到partialSum小於0，而是只有第一個pass才會不斷找base cell直到partialSum ≤ 0，第二個pass開始都是只找gain值是正的cell並丟到另一個set而已，我這麼做的理由是有些測資我並無法在5分鐘內跑完，所以為了能夠在時間內產生相對可接受的cut size，從第二個pass開始我都只挑那些能為我穩定減少cut size的cell作為base，雖然丟一些-gain值的cell並做完updateGain後可能會讓結果更佳，但我是在時間和結果之間做一個tradeoff。

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

💡 我有實作「一條」bucket list就是用來存每顆cell的gain值，利用的是std::map，並且設定好comparator讓它key值由大到小排，主要是我這樣只要每次從begin(因為gain由大到小排序好了)找還沒被lock的cell，就可以直接拿它當作base，另外value則是set of cells，用來存放那些相同gain值的cells，但是和slides有很大區別的是我並沒有實作doubly-linked list，這會導致有時候我需要updateGain時會很慢，因為要到set裡面找cell，而不是slide裡的O(1)。

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

💡 partialSum就是累加每次的gain值，並用維護一個maxPartialSum，如果當前是maxPartialSum，就把當前兩邊set的狀況存起來，我認為我這步有改善的空間，主要是因為我是直接存成一個set，會導致大量的複製，如果可以用一個vector(或stack)來記錄每次搬動的base以及當前vector的size(或可稱為best step)，到時候需要backtrack時，就可以知道maxPartialSum應該如何pop stack或是emplace_back vector來得到。

4. What else did you do to enhance your solution quality (you are required to implement at least one method to enhance your solution quality) and to speed up your program?

💡 我有用下面兩行程式碼來加速I/O，主要是讓iostream跟stdio不同步以減少不必要的開銷，因為我整份都是c++寫的。另外我有嘗試用bash去亂數產生一些seed，讓我的init partition跑完FM後的cut size會好一點點。

```
std::cin.tie(nullptr);
std::ios::sync_with_stdio(false);
```

5. If you implement parallelization (for FM algorithm itself), please describe the implementation details and provide some experimental results.

💡 我並沒有實作平行化

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

💡 我學到挑選合理的資料結構來實作演算法很重要，由於我直接把cell跟net用int儲存，沒有用OOP的方式去把Cell跟Net這兩個class建出來，導致後面有些operation非常慢，例如有時候要去set of cells找某顆cell，可能就會花log (size of set)的時間，但若是我有指標可以指到cell的話，就能省下大量時間。

💡 我還學習到要適當設break point，一開始興沖沖的把講義的東西依照自己的理解一口氣打完(即算before和after的to跟from)，後面才發現很多地方都算錯或是跟想像中不同，那這時候debug就很廢日曠時，所以有時候要先從簡單的小case慢慢try，才不會做白工。

💡 有遇到一開始跑非常非常慢，後來發現是因為每次移完base後，我都整個重算cut size，但其實gain值的意義就是讓cut size減少的量，所以直接拿上一次的cut size扣掉gain值即可，但也因此發現了updateGain有問題的bug。

💡 關於updateGain()的bug，我一開始是先算好before的from和to值，並更新對應的cell gain，之後再移動base，再更新from和to值，然後一樣更新對應的cell gain，後來跟同學討論後，多利用了netGroupCnt，即某條net在兩邊set的cell數來幫助更新，就有成功更新gain值。