一、 執行結果

```
[g114062529@ic21 ~/HW2_grading]$ bash HW2_grading.sh
+-----------------------------------------------+
|                                               |
|    This script is used for PDA HW2 grading.   |
|                                               |
+-----------------------------------------------+
host name: ic21
compiler version: g++ (GCC) 7.3.0

grading on 114062529:
 checking item          | status
------------------------|--------
 correct tar.gz         | yes
 correct file structure | yes
 have README            | yes
 have Makefile          | yes
 correct make clean     | yes
 correct make           | yes

  testcase |      #ways |   cut size |    runtime | status
-----------|------------|------------|------------|--------
   public1 |          2 |        803 |       0.29 | success
   public1 |          4 |       1374 |       1.10 | success
   public2 |          2 |       2841 |      16.70 | success
   public2 |          4 |       5325 |      67.01 | success
+-----------------------------------------------+
|                                               |
|   Successfully write grades to HW2_grade.csv  |
|                                               |
+-----------------------------------------------+
[g114062529@ic21 ~/HW2_grading]$
```

二、 實作細節說明

1. 2-way FM 整流程與課堂簡報相似，並且為了獲得更好的 cut size，實作 multi-level FM，使用 Edge coarsening 演算法。
2. 對於 4-way FM，則使用 6 次 2-way FM 來完成，詳細做法如下。
   (1) 將全部 cell 平均分配到 group A、B 並做一次 FM。
   (2) 將 group A 中的 cell 平均分配到 group A、C 並做一次 FM。
   (3) 將 group B 中的 cell 平均分配到 group B、D 並做一次 FM。
   (4) 依序在 group A、D，group B、C，group C、D 做一次 FM。
3. Edge coarsening 採用與課堂簡報相同的邏輯。但計算 weight 時，為了減少計算量，若 net 的 $|h| > 100$，略過該 net，因為其影響太小。並且先計算每個 cell 的鄰接表(adjacency list)，加速兩兩配對 cell 的過程。

4. 主要資料結構如下。我使用 vector<vector<BucketH*>> buckets 來建立 bucket list。其中 down/up 會指向 gain 較小/較大的 bucket，因此將某 cell 的 gain 加減一，要移動它到鄰近的 bucket 時，能達到 O(1)的複雜度。而 s 會儲存具有相同 gain 的 cell id，並且使用 unordered_set 即可符合需求。

```
struct Cell {
    int id;
    int size;
    int gain = 0;
    int part = 0;
    int parent1 = -1;              // for coarsening
    int parent2 = -1;              // for coarsening
    bool locked = false;
    unordered_set<int> pins;       // nets connected to
    Cell(int i, int s) : id(i), size(s) {}
};
```

```
struct BucketH {
    BucketH* down = NULL;
    BucketH* up = NULL;
    unordered_set<int> s;
    int size;
    BucketH(int size) : size(size) {}
};
```

5. FM 演算法每次選擇 cell 的優先順序為（對於兩個 group A、B）：
   (1) 若 A 已經沒有可移動的 cell，則從 B 移動至 A，反之亦然。
   (2) 若 A 的 size 太大，從 A 移動至 B，反之亦然。
   (3) 若以上兩點皆沒執行，從 A、B 中選擇 gain 較大的移動至另一方。
6. FM 會移動全部 cell 並選出 cut size 最佳的前 n 步。假設總共 N 步，在全部 cell 移動完畢後，第 n 步是最佳結果，我反轉最後結果 (移動完畢的 cell) 中第 n+1~N 步移動的 cell 的 group，再去重新設定 bucket list。這樣可以快速回溯到指定步數，並且不用額外記錄下先前的舊資訊。
7. 一次 FM 演算法會重複做 5.、6.，直到 cut size 並不會變的更好為止。
8. 在 7.完整移動完所有 cell 之前，當某 cell 被移出 bucket list 後，不會再插入到目的地的 bucket list 裡，因為該 cell 已被鎖定，暫時不能再次移動，所以不必插回去，可以減少計算量並簡化程式碼。
9. 將某 cell 從 group A 移動到 B 後，可能會需要更新該 cell 的 net 上其他 cell 的 gain，此步驟需要大量計算量。因此我紀錄某個 net 在 A、B 上是否都已經有被鎖定的 cell，若都有，則那條 net 不會再對 gain 造成影響，因此更新時就可以略過，減少計算量。
10. 作業指定問題
    I. A. 在 bucket list 中，使用 unordered_set 來取代 linked list。詳見 4.
       B. 在 coarsening 時略過 size > 100 的 net。詳見 3.
       C. 暫時不將移動後的 cell 插入到目的地 bucket list。詳見 8.
    II. 在 bucket list 中，使用 unordered_set 來取代 linked list。詳見 4.
    III. 在移動過程中紀錄下每一步移動哪個 cell，以及哪一步擁有最好的 cut size，最後反轉指定 cell 來恢復紀錄。詳見 6.
    IV. 詳見 2.

三、 學到什麼/遇到的困難

1. 實作 bucket list 時，原本深受指標管理困擾，但仔細思考後發現可以使用 unordered_set 來取代橫向指標，使實作困難度大大降低。
2. 完成 2-way FM 後發現結果不佳，為了使結果更好，使用 multi-level FM，但這會使計算量大大提高，因此使用各種方法來盡量減少計算量，包含「二、實作細節說明」中的 3、6、8、9。
3. 原先想實作整體性的 4-way FM，但在 gain 的計算上有些問題，導致 cut size 會跑到無限小，因此最終只能改用 2-way FM 的組合來實現。
4. 在這項作業中主要學到的是，課堂上的演算法或許並不複雜，但是在實作時往往需要注意資料結構的使用以及程式碼複雜度，否則程式可能會效率極低，並且很多看似不起眼的小細節可能會造成很大的影響。

四、 AI Tool Usage

使用 chatgpt 來輔助我完成這項作業，具體用法如下。

1. 詢問讀取及寫入 txt 檔的相關語法，包含 ifstream、getline、ofstream 等。
2. 詢問對於 coarsening 次數的建議。
3. 詢問對於 coarsening 消耗大量時間的問題，它建議我忽略 size 過大的 net。
4. 協助我發現要遍歷 buckets 來進行初始化的 bug，若沒有遍歷 buckets，所有 buckets[p][i]會指向同一個 new BucketH(0)。
5. 協助我發現 initial_bucket 中忘記寫 c->gain = 0; 的 bug。
6. 提醒我 FM 需要移動完所有 cell，才去尋找最好的 partial gain。