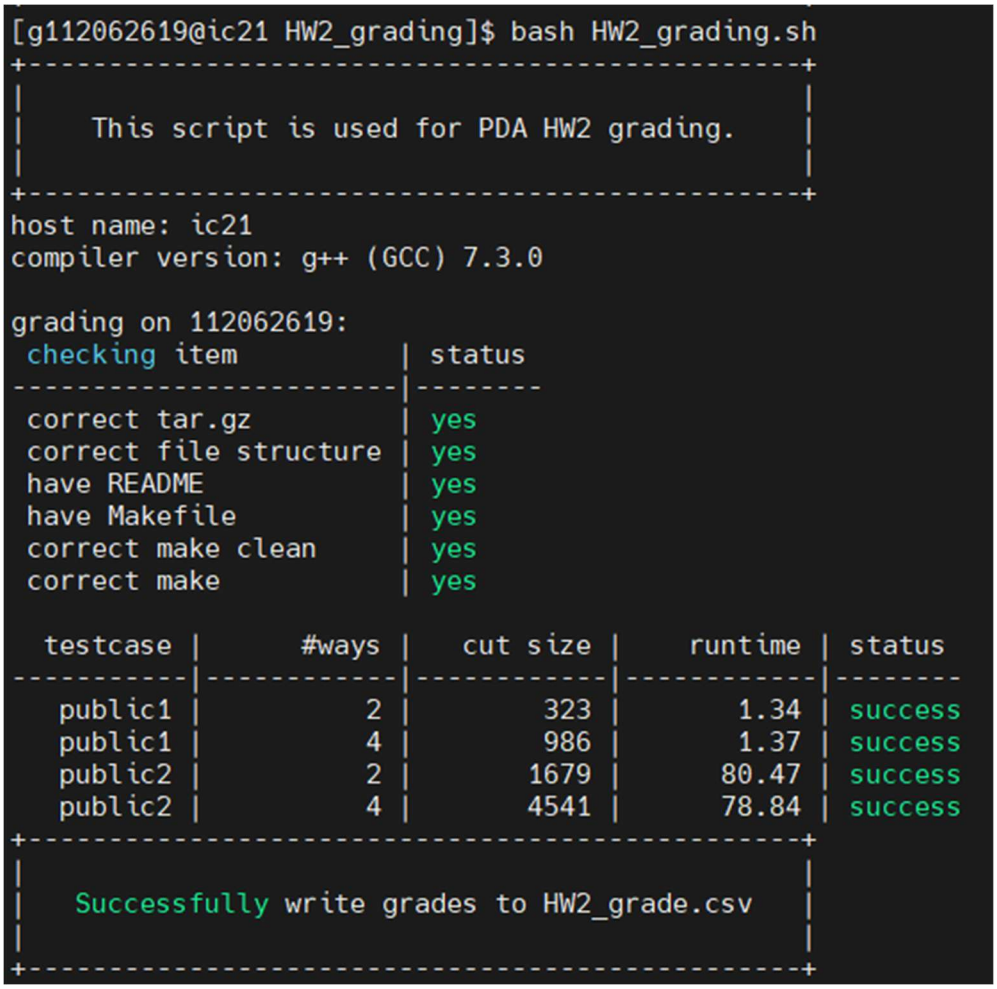
CS6135_HW2_112062619_report

1. 112062619, 陳航希

2.



3.

Overall Algorithm Flow:

整體演算法分為兩個主要部分：2-way 與 4-way。

程式會根據輸入參數 k 決定要執行哪一種分割。

Two-way Partitioning

2-way 部分採用了課堂教的fm演算法，整體流程如下:

我先根據不同策略 如: 面積排序、隨機, 產生初始分割，為了讓最後得到的結果最佳，我在初始化的時候嘗試了多次的隨機初始化，因為總共可以跑的時間為100秒，所以針對2-way的情況，我的初始化總共跑了10次，實驗證明跑越多次確實可以提升最後的效

果。進入fm的程式後，首先計算每個 net 的兩側 cell 數量 (cntA, cntB)，並據此初始化每個 cell 的 gain 值。接著為每個 partition 建立一個 bucket list, 這裡使用 vector<list<int>>，索引代表不同 gain 值。每個 bucket 會維護當前最大 gain 的索引 maxId，這樣能在 O(1) 時間內找到最佳的cell。

我在最佳化的過程當中會持續從 bucket 中挑選 gain 最大且符合面積平衡條件的 cell。這個部分的實作在chooseCell()，寫這個部分的code的時候因為一下子會陷入無窮迴圈或是遇到seg fault所以相當麻煩。

在執行 cell 移動後會更新 cut size，同時更新與該 cell 相連的 net 上其他 cell 的 gain，這個步驟需要仔細對照上課教過的演算法才不會出錯。過程當中會用 rebucketCell() 將受影響的 cell 重新插入 bucket。

Partial Sum Tracking的部分，我在每次移動過程中都會記錄 cut size 的變化，若出現比前一輪更小的 cut 值就會記下當前索引作為 rollback point。一輪結束後回復到該 rollback 點，確保保留該輪中最佳結果。若新的 cut size 小於全域最佳，則更新結果並進行下一輪。

Four-way Partitioning

為了將 2-way 延伸到4-way，我採用了 遞迴式 2-way 分割。

首先執行一次 2-way FM 將整體分成兩群 A、B。這個時候跟2-way一樣會嘗試三種初始策略 (面積排序、隨機、net degree)，隨機的部分也會多跑幾次以達到更好的效果，最後取 cut size 最小的結果。跟2-way較不同的地方在於因為4-way最後要求的結果為每個group佔的 0.225<=area大小<=0.275, 而recursive跑也需要確保符合這個條件。所以我將兩階段的fm條件設的嚴格一點，以確保不會越線。最後設為 lower: 0.475->0.475, upper:0.524->0.524, 這個數字可以確保最後的結果符合規則。

接著將群組 A、B 各自轉換成Subproblem，建立 local 與 global id 的映射表，並只保留屬於該子群的 cell 和 net。

接著對子群 A、B Z分別執行 2-way FM 這個時候也會做多次的fm，最後選擇5次，這樣可以分別得到 A1/A2、B1/B2。最後四個群組依序對應到 GroupA - GroupD。

相較於課堂上介紹的標準 FM 演算法，我的實作包含以下幾項差異:

1. 相較於課堂上只有使用面積做初始化，我使用三種不同初始策略(面積、隨機、degree)重複執行 FM，選取 cut size 最小結果，這樣會讓最終結果更好。

2. 根據作業要求每次執行都有設定時間上限，確保在系統規定的 100 秒內完成所有測資。
3. 課堂中所教的方法為使用 rW-S_max <= |A| <= rW+S_max 為balance的限制，與這次作業的限制稍微不同。

Bucket List的部分，我實作的bucket list 結構與課堂介紹相同，每次執行fm的時候只會有2個buckets，結構為 vector<list<int>>，index對應不同 gain 值(gain + pMax)。
透過 iterator 讓我可以在 O(1) 時間刪除或重新插入 cell。

4. What I Have Learned and Problems Encountered

在這次作業中，我對 FM 演算法的細節與行為有了更深的理解。
以往只在課堂上看過理論流程，但在實作過程中，才真正體會到資料結構與效能控制的重要性。在寫的過程裡面，如果沒有細心控制bucket list的結構, max index的大小整個演算法就會完全亂掉，可能會遇到seg fault或loss降不下來，所以需要謹慎處理。整體而言透過這次作業讓我又再複習了一次c++的實作跟partitioning的方法，學到相當多的東西。

5. AI Tool Usage Disclosure

在這份作業的撰寫過程中，我使用 GPT 作為輔助工具，用來協助我除錯與程式重構（如 segfault、bucket 更新邏輯、遞迴分割流程等部分）。