HW2 report

111065519 林詰珊

1. How to compile:

step1. 進入資料夾

```
$ cd HW2/src
```

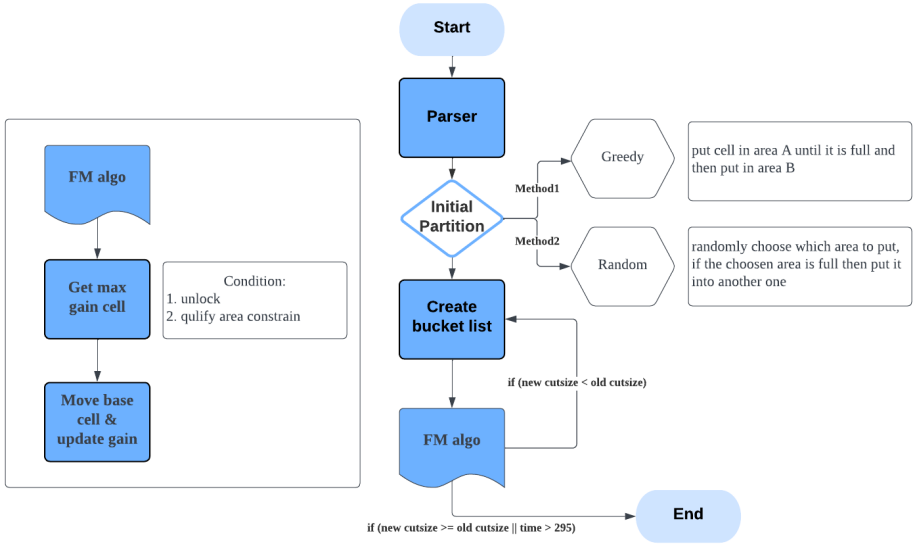
step 2. 執行程式

```
$ make
$ ../bin/HW2 ../testcase/public1.txt ../output/public1.out
格式為(../bin/HW2 ../testcase/<input filename>.txt ../output/<output filenema>.out
```

2. final cut size and runtime for each testcase

	public1	public2	public3	public4	public5	public6
Cutsize	155	1494	5519	2076	775	9295
Runtime(s)	0.04	7.75	27.76	1.55	17.10	191.46

3. Algo detail

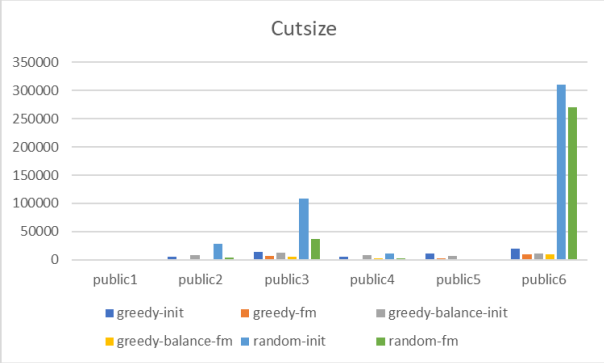


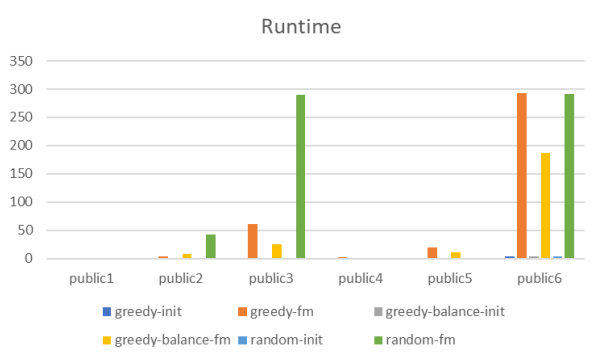
4. Solution enhance

我的作法是會執行FM algo一直到cutsize無法再增加，但這樣的作法在public6會超時將近一分鐘，所以我有設計一個timer讓iteration可以在超過一定時間時停止，同時我發現FM雖然很有用，但initailize的方法對於最後cutsize會有很大的影響，我有嘗試使用兩種不同的方法做initialize:

1. random 決定要放a或b，如果被隨機到的地方放滿了就放到另一個(但這個方法在某些case上不適用，public5就是不管怎麼random都放不進去)
2. 先全部想辦法放到die a，然後放滿後再放到die b
3. balance initial: 因為如果先放die 某個die可能會導致它很滿，所以我後來設定說先將某個die 放到95%滿後，再去放另一個die，這樣做完initial的結果或許會比較平均，同時實驗結果也證實我這樣使用的initial結果在後面的fm algo有正面的影響。拿public6來說好了，我用方法2 public6執行時間是293秒，而用balance initial的方式的話，只要180秒就收斂了，而且收斂效果也比較好，我推測原因就是因為我initial的結果不同，讓他找base cell的時候可以很快就找到可以替換的並很快達到平衡。

結論: 論速度的話一定都是只有initial是最快的，但為了盡量減小cutsize所以會在後面加上fm algo做優化，三種initial方法加上fm都可以基於原本initial 的結果再往上提升，然後三種initial方法由好到壞是3 → 2 → 1，不但fm後的結果比較好，而且runtime也因為initial的好也有些許的降低





Cutsize	public1	public2	public3	public4	public5	public6
greedy-init	374	4981	13552	5257	10606	19245
greedy-fm	152	1564	6466	**1604**	2546	10425
greedy-balance-init	412	8574	13148	7821	7224	11871
greedy-balance-fm	**155**	**1494**	**5519**	2156	**775**	**9295**
random-init	1383	27844	109027	10574	x	310426
random-fm	220	3937	36657	3170	x	269977

Runtime (s)	public1	public2	public3	public4	public5	public6
greedy-init	0.015572	0.169701	0.968979	0.0633	0.5017	3.3974
greedy-fm	0.004	4.49	60.82	2.03	19.65	293.15
greedy-balance-init	0.0158	0.1808	0.958	0.059	0.48012	3.4791
greedy-balance-fm	0.0387	7.5465	25.7656	1.4378	10.7307	187.138
random-init	0.017	0.16678	0.998	0.0606	x	3.43425
random-fm	0.06	41.95	290.236	0.9984	x	290.8

5. parallelization

No.

6. What I learned from this homework?

這次作業是我第一次寫c++，因為第一次就寫這麼龐大的程式，我在作業中學到很多資料結構的知識，像我一開始使用unorder map去儲存data，結果到最後發現public 2 以後的測資都要跑很久，才發現因為unorder map不保證O(1)的讀取時間，所以會導致我後面更大的data size時讀取時間過長而沒辦法在規定的時間內跑完，所以後來改為使用vector來儲存，時間就變快很多，也讓我的fm algo可以跑多次一點的iteration來達到更好的收斂結果。同時這次作業因為在儲存data的時候常常遇到沒有設計好index導致發生segment fault，所以之後如果也有要做類似的處理，我需要盡量簡化我的程式碼讓我在做trace code的時候更有效率