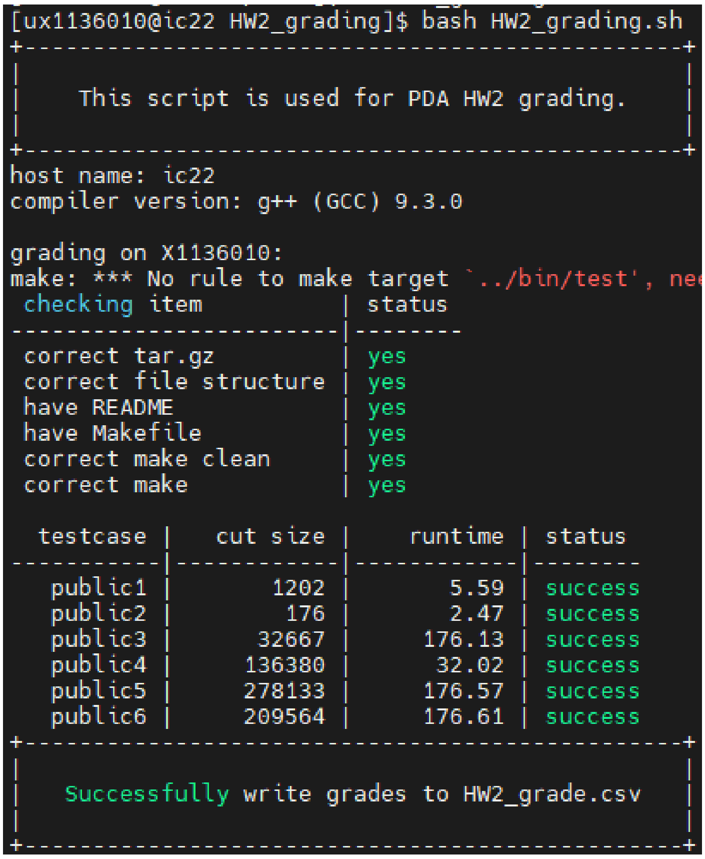
黃偉祥 X1136010

How to

1. Go to HW2/src/ and use `make` command to compile
2. Then go to /HW2/bin and use `./hw2 input.txt output.txt timelimit maxG` to execute

```
cd HW2/src/
make
cd HW2/bin
./hw2 ../testcase/public2.txt ../output/public1.txt 170 100
```

Result



```
[ux1136010@ic22 HW2_grading]$ bash HW2_grading.sh
+-------------------------------------------------+
|                                                 |
|    This script is used for PDA HW2 grading.     |
|                                                 |
+-------------------------------------------------+
host name: ic22
compiler version: g++ (GCC) 9.3.0

grading on X1136010:
make: *** No rule to make target `../bin/test', nee
 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 |  cut size |    runtime | status
-----------|-----------|------------|--------
   public1 |      1202 |       5.59 | success
   public2 |       176 |       2.47 | success
   public3 |     32667 |     176.13 | success
   public4 |    136380 |      32.02 | success
   public5 |    278133 |     176.57 | success
   public6 |    209564 |     176.61 | success
+-------------------------------------------------+
|                                                 |
|   Successfully write grades to HW2_grade.csv    |
|                                                 |
+-------------------------------------------------+
```

Algorithm

*** Functions from deepseek and edit by myself***

1. string getNextMovableCell(BucketList& bl, const map<string, Cell>& tmp_cells,const Die& tmp_dieA, const Die& tmp_dieB)
2. BucketList Get_bucket(map<string, Cell>& cells)
3. void Update_bucket_for_cell(const string& cellname, map<string, Cell>& cells, BucketList& bucket, const int oldGain)

*** Functions from lecture ppt ***

1. void Update_Gain(const string& cellname,map<string, Net>& nets,map<string, Cell>& cells,BucketList& bucket)
2. void update_net_critical(const string& netname,map<string, Net>& nets,map<string, Cell>& cells)
3. void compute_cell_gain(const string &cellname,map<string, Net>& nets,map<string, Cell>& cells)

*** Functions by myself ***

1. void parseInput(const string &Infile,map<string, Technology>& techs,Die& dieA,Die& dieB,map<string, Net>& nets,map<string, Cell>& cells)
2. void initialPartition(pair<ll,ll> range_r,ll r,map<string, Net>& nets,map<string, Cell>& cells,Die& dieA,Die& dieB)

```
int main(int argc, char* argv[]) // main program
    input file and parseInput    //Get input file and store the data
    initialPartition
    calculate_all_cell_gain

    set G = 1
    while(G>0) then
        if exceed time limit then break
        create temp die, cells, nets (tmp_dieA,tmp_dieB,…) for pseudo exchange
        Get bucket list from temp die, cells, nets
        Create moveRecords //For recoding which cell moved and calculate partial sum

    For all cells check
        If exceed time limit then break
        Get next moveable cell from bucket list
        // Moveable = max gain and after move area.used < area.max
        If no moveable cell then break
        Move cell (lock, update die)
        moveRecord.append(gain, cell)
        Update gain // Algorithm from lecture ppt
    End For
    G,k = maximum partial sum, how many move
    If G<=0 then break
    If exceed time limit then
        Move cell until k
        Break
    End if
    Move cell until k
    Unlock all cell and nets
    Compute all cell gain
End while

Output file

End
```

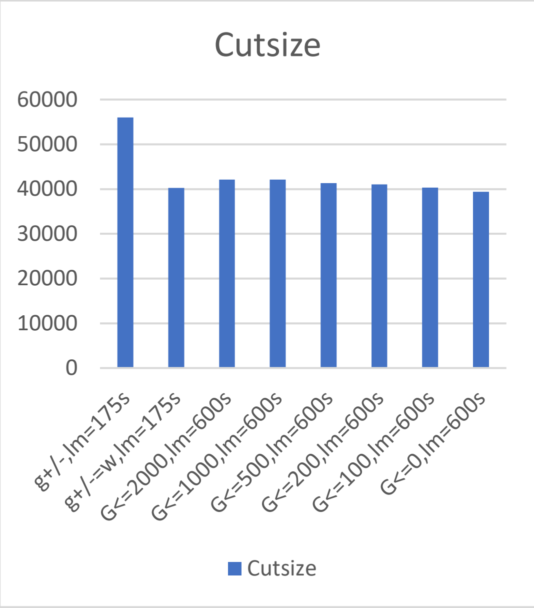
```
initialParition
    sort cells by different area on dieA and dieB
    // if cell1 at dieA is 10 and at dieB is 100, diff = 90
    // if cell2 at dieA is 100 and at dieB is 120, diff = 20
    // After sort cell1 -> cell2
    For all cells
        If cell.areaA < cell.areaB and dieA still have space then put in dieA
        Else if cell.areaA > cell.areaB and dieB still have space then put in dieB
        Else if dieA still have space then put in dieA
        Else put in dieB
    End For
End
```

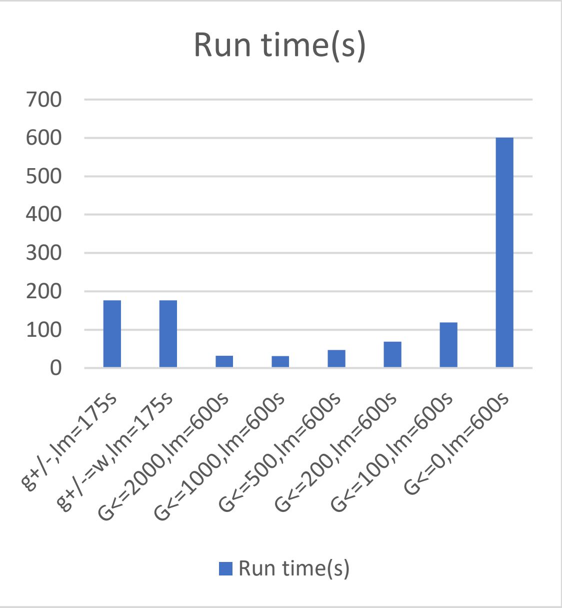
Solutions' qualities compare

- public3.txt

Using unordered_map		
Method	Cutsize	Runtime
gain++/--, limit 175s	55975	176.42
gain+/-=weight, limit 175s	40223	176.64
maxG = 2000, limit 600s	42082	32
maxG = 1000, limit 600s	42082	31
maxG = 500, limit 600s	41319	47
maxG = 200, limit 600s	41058	69
maxG = 100, limit 600s	40315	119
maxG = 0, limit 600s	39395	601

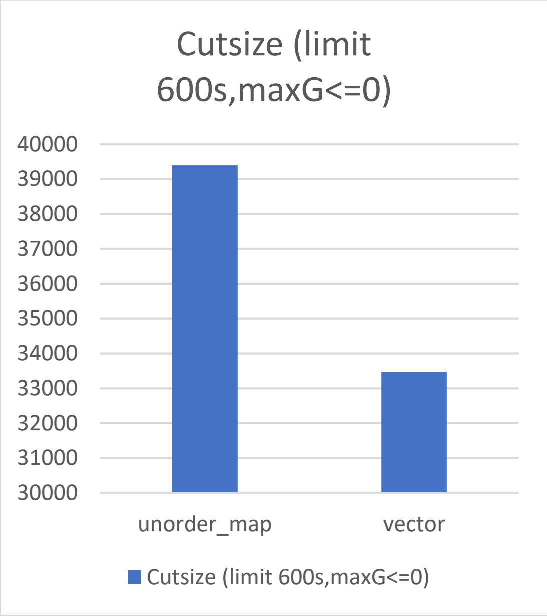
- only method 1 use gain++/gain--, others use gain+/-=net.weight
- When calculating gain, use net's weight will have better result.
- Break the pass if maximum partial sum (maxG) is acceptable to save time, when maximum partial sum is small it may not improve much on result but taking so much time to continue the pass

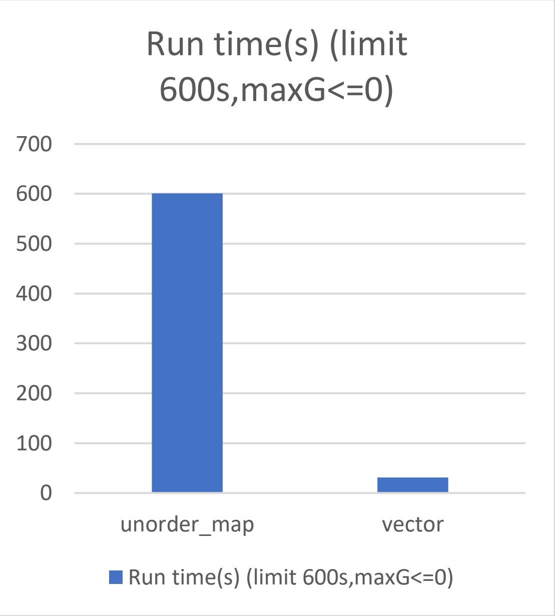




Using vector		
Method	Cutsize	Runtime
maxG = 0	33471	31.23
maxG = 500	33936	8

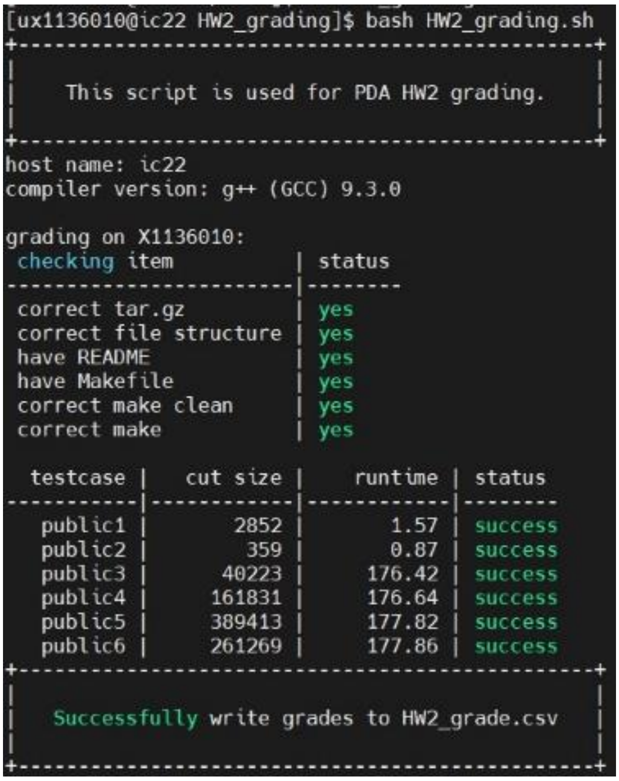
- Using vector to store and manipulate can make the program run faster and also lead to a better result.
- Because when large size of data, unordered_map may meet collision O(N)/O(C) frequently and using idx and vector can get target cell/net in O(1).





Results comparison using HW2_grading.sh

Unordered_map

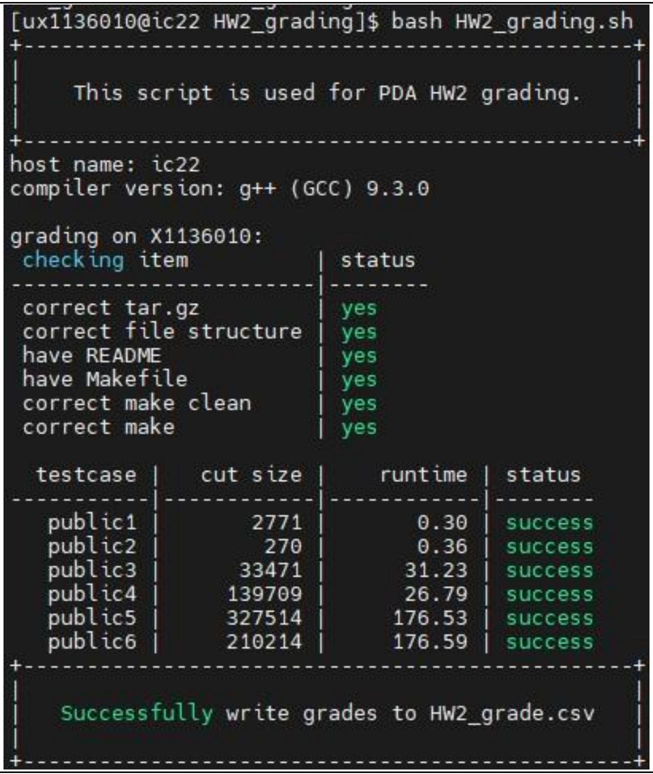


```
[ux1136010@ic22 HW2_grading]$ bash HW2_grading.sh
+-----------------------------------------------+
|                                               |
|    This script is used for PDA HW2 grading.   |
|                                               |
+-----------------------------------------------+
host name: ic22
compiler version: g++ (GCC) 9.3.0

grading on X1136010:
 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 |   cut size |     runtime | status
-----------|------------|-------------|--------
   public1 |       2852 |        1.57 | success
   public2 |        359 |        0.87 | success
   public3 |      40223 |      176.42 | success
   public4 |     161831 |      176.64 | success
   public5 |     389413 |      177.82 | success
   public6 |     261269 |      177.86 | success
+-----------------------------------------------+
|                                               |
|   Successfully write grades to HW2_grade.csv  |
|                                               |
+-----------------------------------------------+
```

Vector

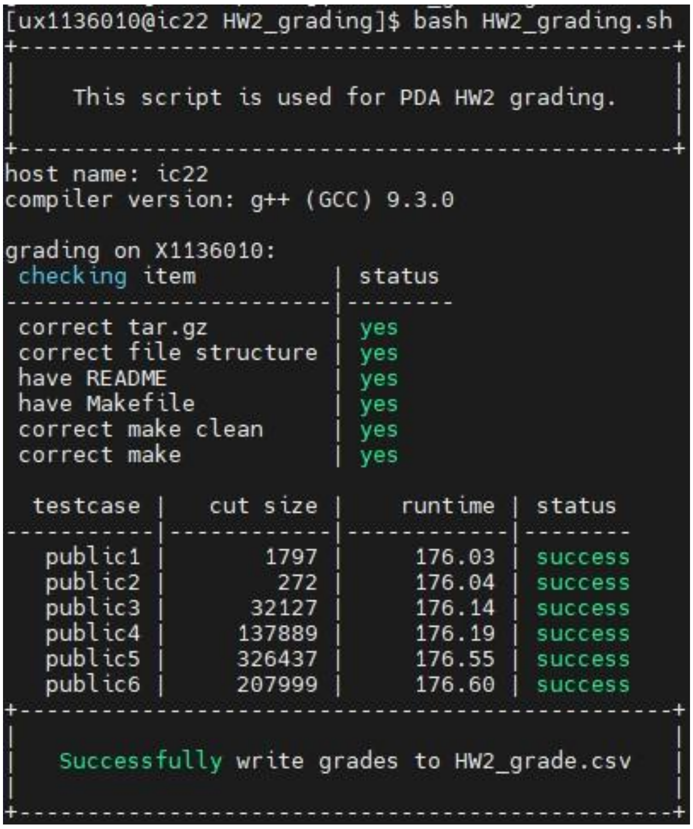


```
[ux1136010@ic22 HW2_grading]$ bash HW2_grading.sh
+-----------------------------------------------+
|                                               |
|    This script is used for PDA HW2 grading.   |
|                                               |
+-----------------------------------------------+
host name: ic22
compiler version: g++ (GCC) 9.3.0

grading on X1136010:
 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 |   cut size |     runtime | status
-----------|------------|-------------|--------
   public1 |       2771 |        0.30 | success
   public2 |        270 |        0.36 | success
   public3 |      33471 |       31.23 | success
   public4 |     139709 |       26.79 | success
   public5 |     327514 |      176.53 | success
   public6 |     210214 |      176.59 | success
+-----------------------------------------------+
|                                               |
|   Successfully write grades to HW2_grade.csv  |
|                                               |
+-----------------------------------------------+
```

Vector + simulated annealing

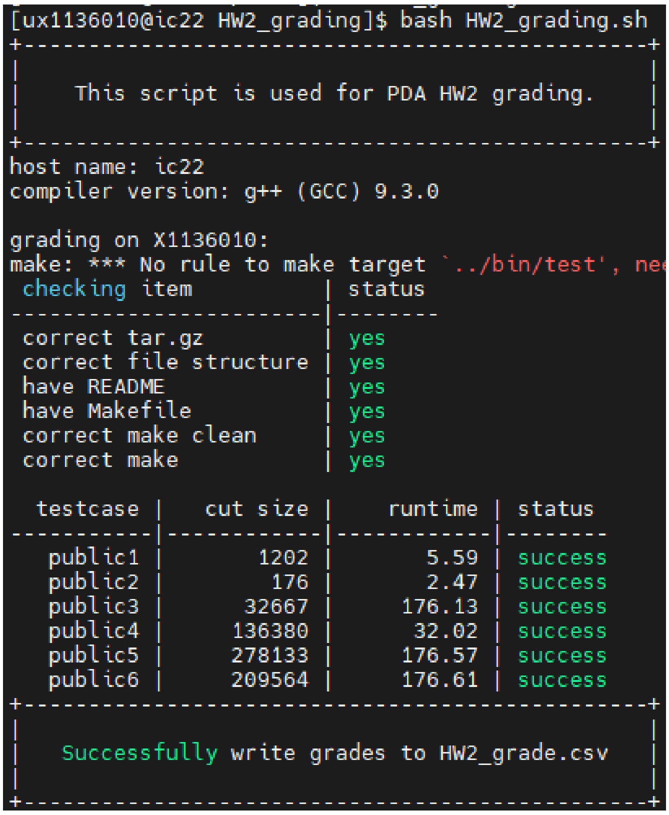


```
[ux1136010@ic22 HW2_grading]$ bash HW2_grading.sh
+-------------------------------------------------+
|                                                 |
|    This script is used for PDA HW2 grading.     |
|                                                 |
+-------------------------------------------------+
host name: ic22
compiler version: g++ (GCC) 9.3.0

grading on X1136010:
 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 |   cut size |     runtime | status
-----------|------------|-------------|--------
   public1 |       1797 |      176.03 | success
   public2 |        272 |      176.04 | success
   public3 |      32127 |      176.14 | success
   public4 |     137889 |      176.19 | success
   public5 |     326437 |      176.55 | success
   public6 |     207999 |      176.60 | success
+-------------------------------------------------+
|                                                 |
|   Successfully write grades to HW2_grade.csv    |
|                                                 |
+-------------------------------------------------+
```

Different initial strategy



```
[ux1136010@ic22 HW2_grading]$ bash HW2_grading.sh
+-------------------------------------------------+
|                                                 |
|    This script is used for PDA HW2 grading.     |
|                                                 |
+-------------------------------------------------+
host name: ic22
compiler version: g++ (GCC) 9.3.0

grading on X1136010:
make: *** No rule to make target `../bin/test', nee
 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 |   cut size |     runtime | status
-----------|------------|-------------|--------
   public1 |       1202 |        5.59 | success
   public2 |        176 |        2.47 | success
   public3 |      32667 |      176.13 | success
   public4 |     136380 |       32.02 | success
   public5 |     278133 |      176.57 | success
   public6 |     209564 |      176.61 | success
+-------------------------------------------------+
|                                                 |
|   Successfully write grades to HW2_grade.csv    |
|                                                 |
+-------------------------------------------------+
```

- Simulated annealing can help to improve cut size in general.
- After get partial maximum sum, run simulated annealing to get more cell move at this pass, it may help to get better result in future pass.
- But simulated annealing will take more time, so set a time limit for it will be a good choice.
- Finally, different initial strategy can lead to better result, ex: use net weight as a information when doing initial partition.

Parallelization

Using OpenMP

- Unlock_all_cells_nets
- compute_all_cell_gain
- move cell for moveRecords

Implementations

- #include <omp.h>
- Add `-fopenmp` while compiling (Makefile)
- Set using maximum of threads
    - omp_set_num_threads(omp_get_max_threads());
- The target for loop to do parallelization
    - #pragma omp parallel for schedule(dynamic)
- To protect critical area
    - #pragma omp critical

Results

Using unordered_map				
Methods	Normal		Parallel	
Limits 1000 second	Cut size	Run time(s)	Cut size	Run time(s)
public3.txt, G<=100	40315	128	40315	119
public4.txt, G<=500	163048	86	163048	85
public4.txt, G<=200	162461	151	162461	141
Limits 10000 second				
public5.txt, G<=2000	350996	1057	350996	1039

public6.txt,G<=2000	248267	875	248267	834

Using vector				
Methods	Normal		Parallel	
Limits 1000 second	Cut size	Run time(s)	Cut size	Run time(s)
public5.txt, G<=1000	330260	121	330260	121
public5.txt, G<=0	326047	350	326047	347
public6.txt, G<=0	210137	305	210137	302

- Parallel is slightly faster than sequence, but still take too much time due to using too many sequential operations.
- When using parallel method, we need to protect the critical area, ex: when doing math operations(++/--/*/divide) to shared variable.
- OpenMP is easy to use to do parallelization to your program but with some limitations.
- But parallelization may not suitable for every problem, because some problem is sequential and not easy to do parallelization.

Lessons Learned

1. c++ struct and OpenMP
2. net weight should be considered when calculating cell's gain
3. map will be slower than unorder_map
4. Using bucket list to stored cells' gain information will speed up so much
5. When scale become large, we can have early stop point for maximum partial sum, instead of only check <= 0, we can give a number like 1000, for example if maximum partial sum smaller equal than 1000 then we stop the program.
   A. It will let program stop early and give us a good result.
   B. Because from experimental above, cut size will not change so much but the runtime will increase so much.
   C. We can also set a runtime limit to stop the program early

Problems

1. Initial partition
   A. At the very beginning, my initial partition algorithm is worse and always cannot fit all cell in both dieA and dieB.

B. I realize the difference company have very difference Libcell area

C. I check the difference between 2 companies, and much of the difference is so big

D. Then I change my algorithm, I sort through the difference and start partition from biggest difference and use the smaller Libcell according to the Tech company

E. And done!

2. Time limit

   A. My program always run over 180 seconds due to using map to store the data

   B. I change to unordered_map to store data but still exceed time limit.

   C. I change to vector to store data but still exceed time limit on some case.

   D. I try set a maxG to let program stop earlier but the HW2_grading.sh will not dynamic give a suitable maxG

   E. Then I set time limit in my program and pass all the base line now!