DATA STRUCTURE FINAL PROJECT

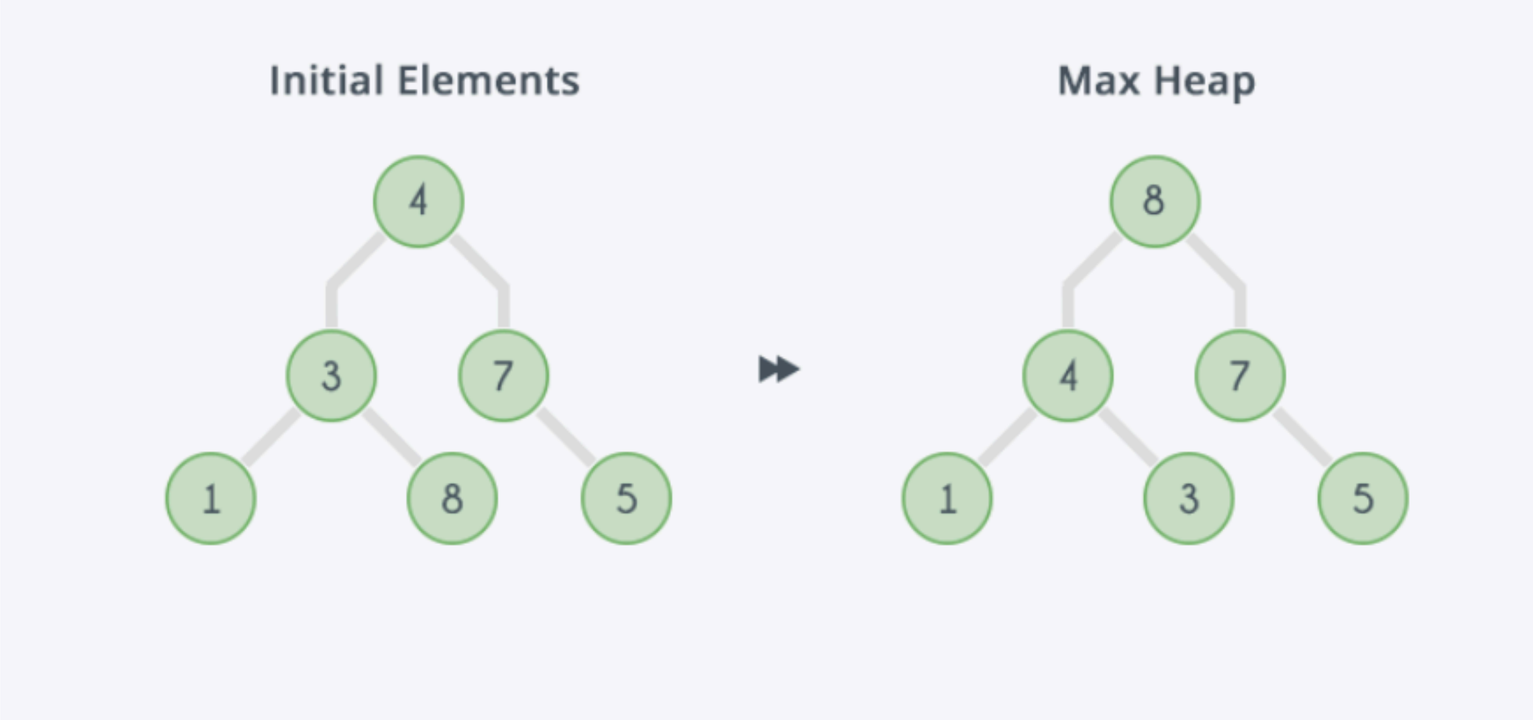
組員

A1115513 劉沛辰　　A1115530 劉柏均

A1115531 錢昱名　　A1115532 Jaxx Zu

目錄

資料結構 -HEAP

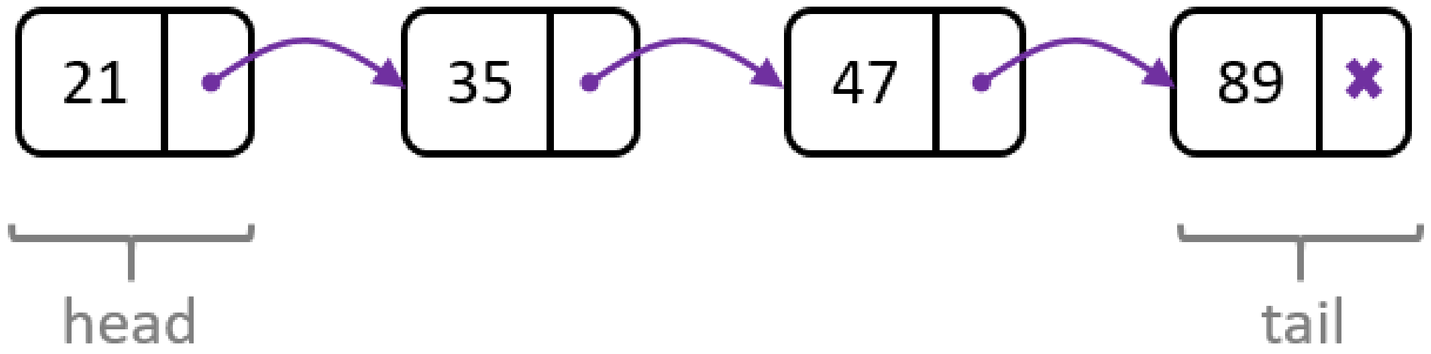


資料結構 -HEAP

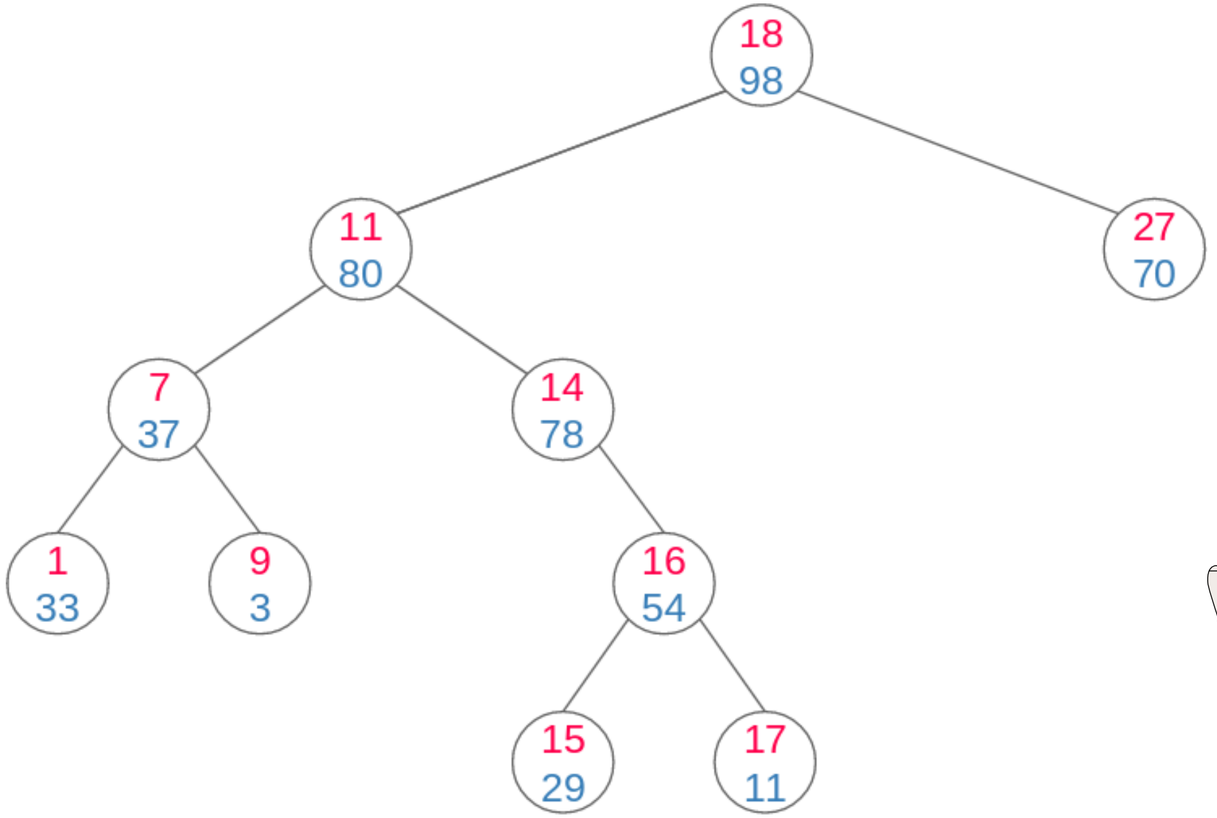
```
95      void pop(){
96          if(n == 0){
97              cout<<"pop Heap is empty"<<endl;
98              throw "Heap is empty";
99          }
100         swap(root[0],root[n-1]);
101         n--;
102         down_Heapify(0);
103     }
```

```
117     void sort(){ //heap sort but it is not Heap anymore
118         int tmp_Size=n;
119         // node *res;
120         while(!empty())
121         {
122             pop();
123         }
124         n=tmp_Size;
125         int j = 1;
126         for (int i = 1; i < n; ++i) {
127             if (root[i].value != root[i - 1].value) {
128                 root[j++].value = root[i].value;
129             }
130         }
131         n=j-1;
132
133         // return root;
134         return;
135     }
```

資料結構 -LINKED-LIST



資料結構 -TREAP

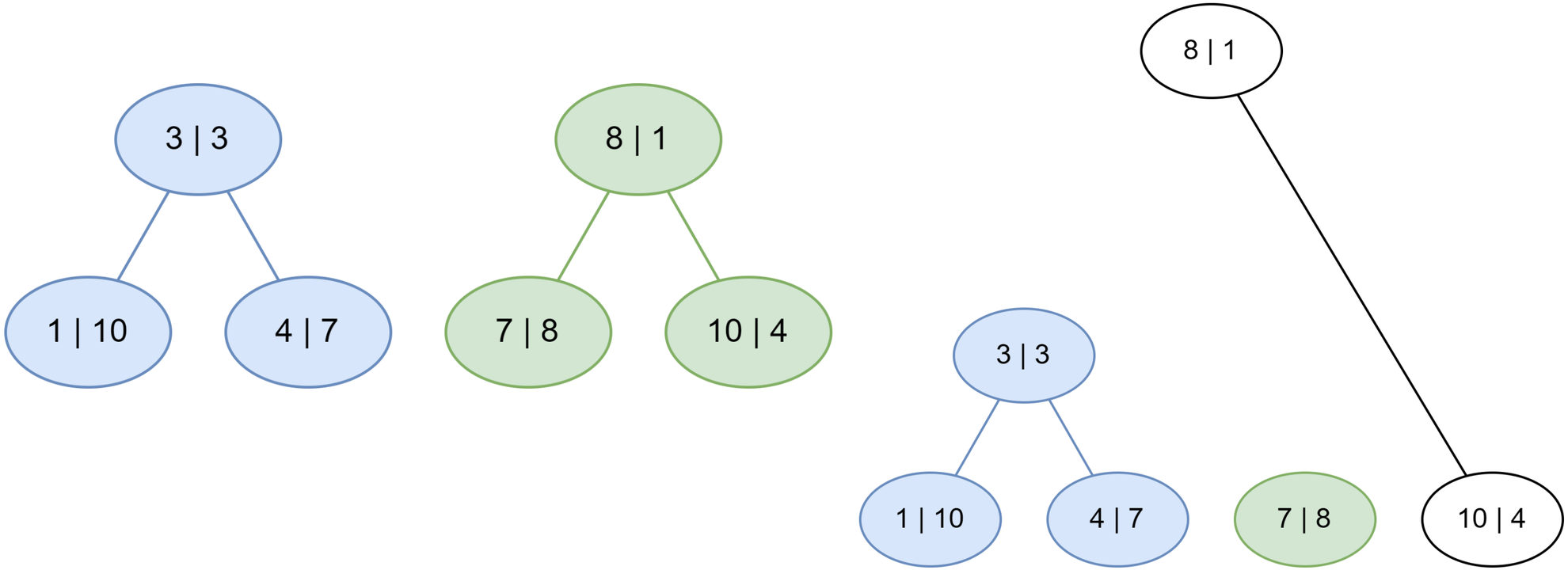


資料結構 -TREAP

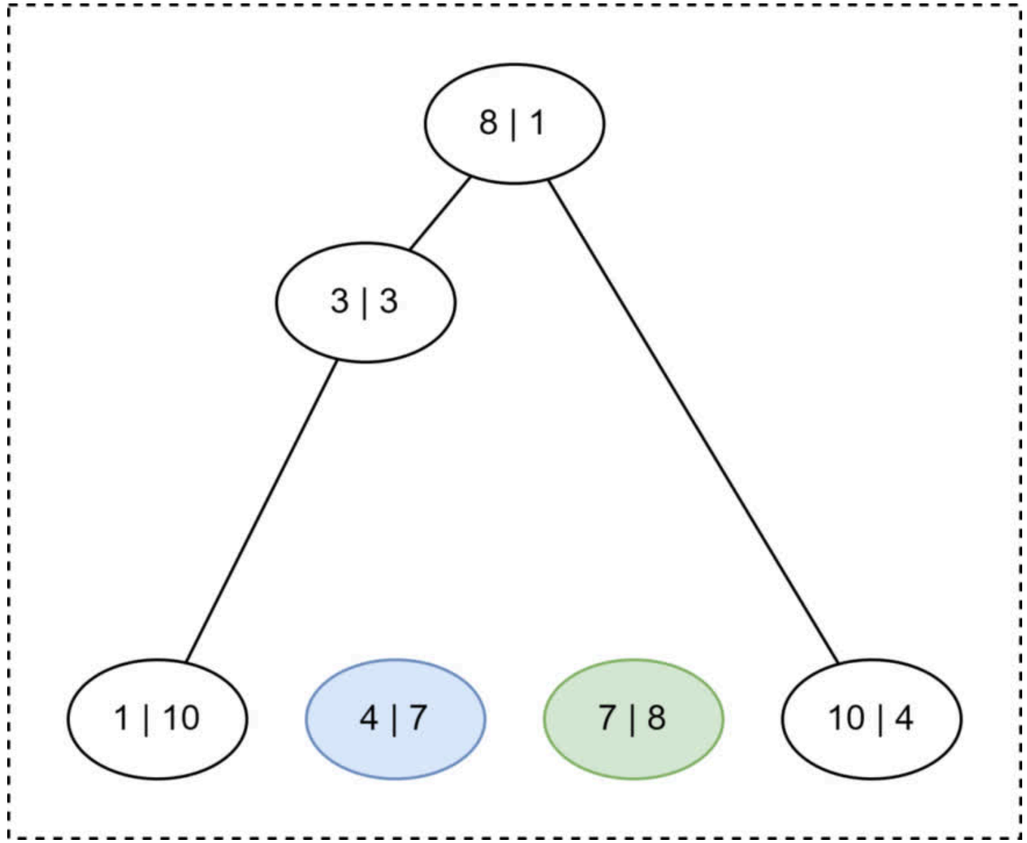
```
9   class treap_Value {
10  public:
11      virtual treap_Value& operator=(const treap_Value&) {
12          return *this;
13      }
14      virtual bool operator!=(const treap_Value &b) const = 0;
15      virtual bool operator<(const treap_Value&) const = 0;
16      virtual bool operator>(const treap_Value&) const = 0;
17      virtual bool operator==(const treap_Value& b) const=0;
18      virtual ~treap_Value() = default;
19  };
```

```
21  template<class T>
22  class Treap {
23  private:
        You, 前天 | 1 author (You)
24      class node {
25      public:
26          T value;
27          long long priority;
28          long long size;
29          node *left, *right;
30
31          node(T __value, int __priority): value(__value), priority(__priority), size(1), left(nullptr), right(nullptr) {}
32      };
33
34      node *root;
35      int n;
36      mt19937 rng;  // 隨機數生成器
37      uniform_int_distribution<int> dist;  // 均勻分布        You, 前天 • Refactor linklist.hpp to remove commented code …
```

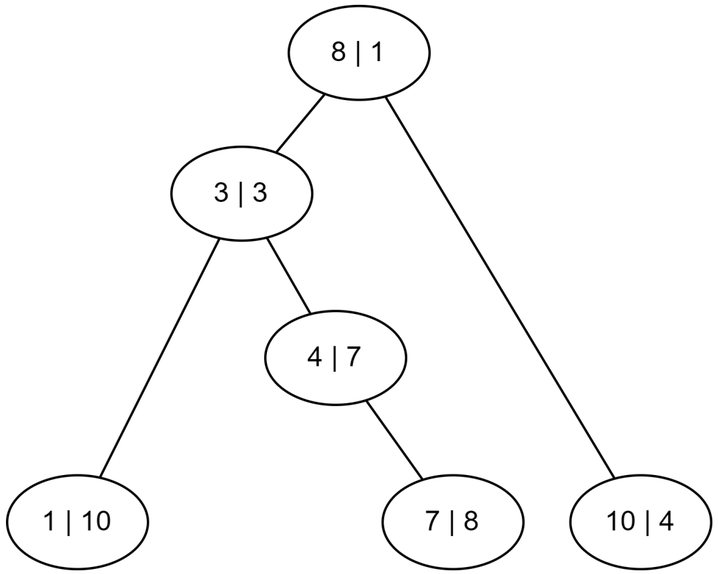
TREAP-MERGE

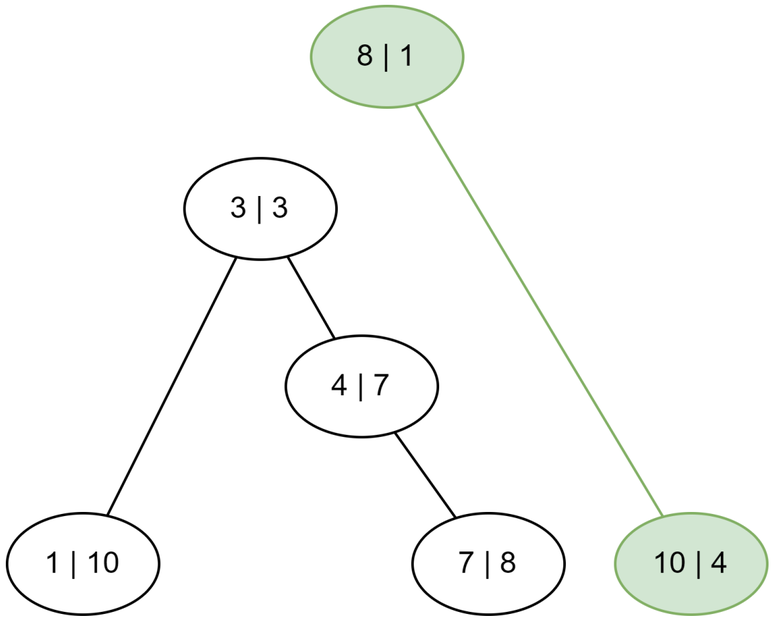


TREAP-MERGE

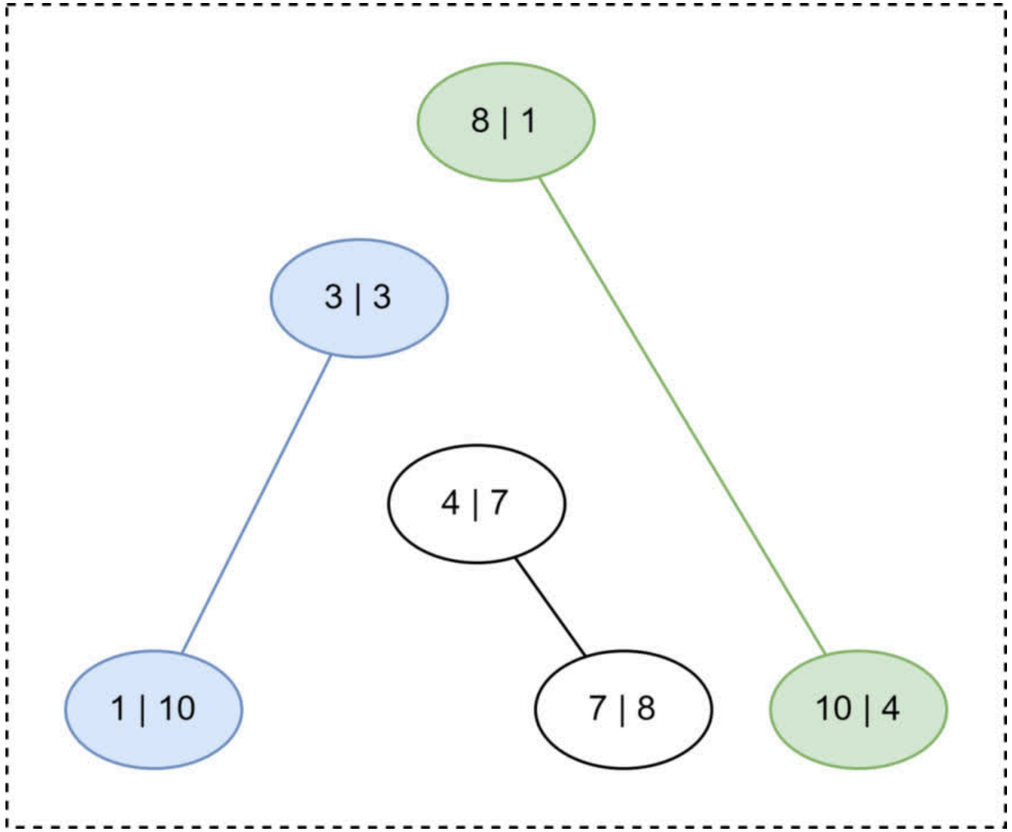


TREAP-SPLIT



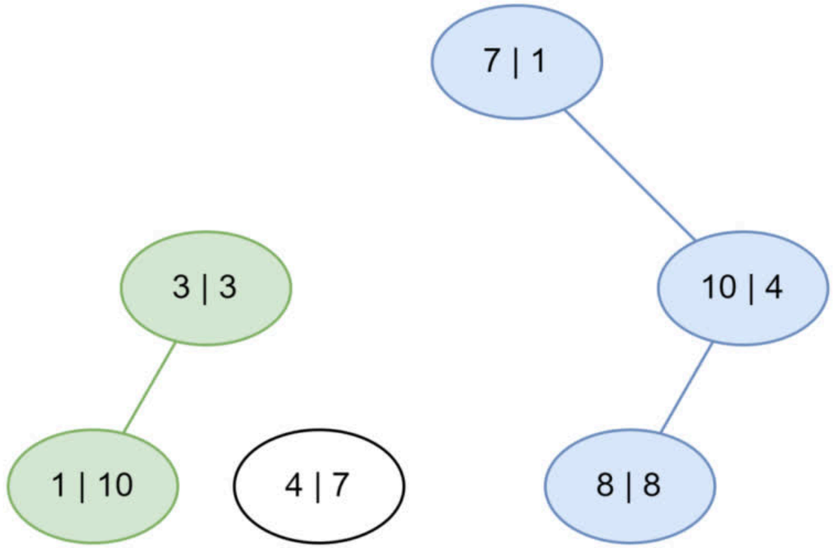


資料結構 -TREAP



TREAP-INSERT

Split



Q1-A-任務要求

- 1. 有幾個唯一日期

```
36      bool operator<(const treap_Value& b) const override {
37          return Date < dynamic_cast<const k_Bar&>(b).Date;
38      }
39      bool operator>(const treap_Value& b) const override {
40          return Date > dynamic_cast<const k_Bar&>(b).Date;
41      }
```

```
158     void insert(T value) {
159         if (find(value)) {
160             return;
161         }
162         insert(value, dist(rng));
163         n++;
164     }       penguin_72487, 2 週前 •
```

```
138     void toList(node* cur, linklist<T>& res){
139         if(cur == nullptr){
140             return;
141         }
142         toList(cur->left, res);
143         res.push_back(cur->value);
144         toList(cur->right, res);
145     }       You, 6 小時前 • refactor: Split tre
```

Q1-A-任務要求

- 2最大10個價格與日期
- 3最小10個價格與日期
- 中位價格及日期

```
105     cout << "2: Small 10" << endl;
106     Treap<k_Bar> a, b;
107     treap.slip_By_Size(a, b, 10);
108     cout << a << endl;
109
110     cout << "3: Big 10" << endl;
111     treap.slip_By_Size(a, b, treap.size()-10);
112     cout << b << endl;
113
114     cout << "4: Median" << endl;
115     if(treap.size()&1){
116         cout << treap[treap.size()/2] << endl;
117     }
118     else{
119         auto it = treap[treap.size()/2];
120         auto it2 = treap[treap.size()/2-1];
121         cout<<it<<it2;
122         cout << "Average: " << (it.get_close()+it2.get_close())/2 << endl;
123     }       penguin_72487, 昨天 • Refactor treap.cpp to remove unused code
```

Q1-A-輸出及圖表

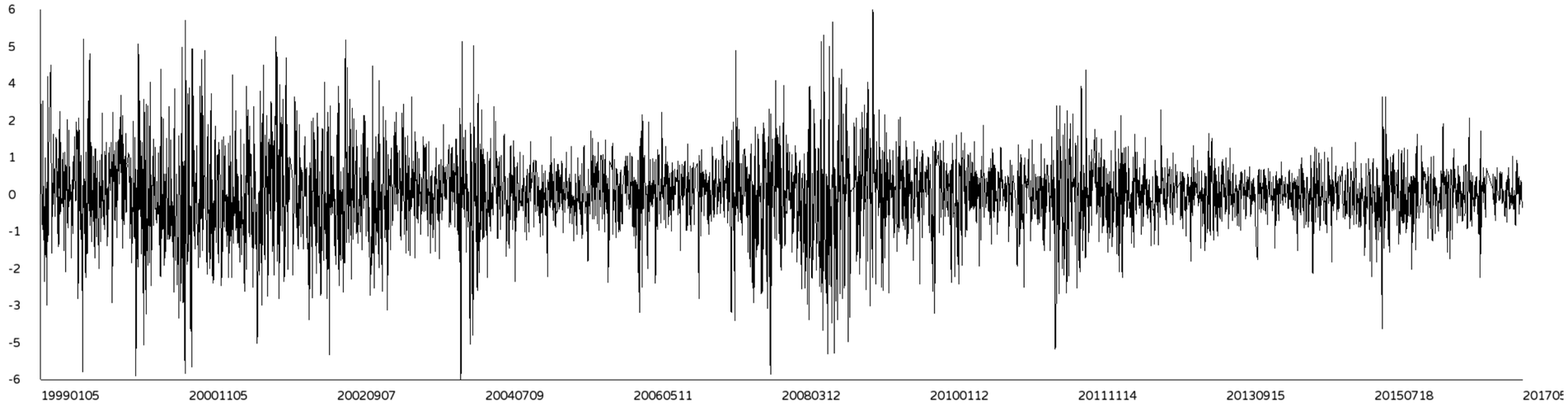
```
1:
4571

2:
20011003 3446.26
20011002 3492.12
20011004 3493.66
20010925 3493.78
20011008 3520.35
20010924 3533.51
20010927 3567.63
20011005 3585.46
20010921 3591.85
20011009 3618.93

3:
20000401 10050.4
20000210 10057.7
20000216 10064.5
20000411 10068
20000218 10096.4
20000410 10127.5
20000211 10128.7
20000219 10161
20000405 10186.2
20000217 10202.2

4:
20100706 7548.48

5:

max_return: 6.74218 on 20090430
min_return: -6.67886 on 20040322

6:
max_return: 6.96365 on 20001121
min_return: -7.20747 on 19990716

10:
max_open_price: 10266.8 on 20000218
min_open_price: 3475.87 on 20010926
max_high_price: 10393.6 on 20000218
min_high_price: 3511.38 on 20011003
max_low_price: 10096.5 on 20000211
min_low_price: 3411.68 on 20010926
max_close_price: 10202.2 on 20000217
min_close_price: 3446.26 on 20011003
mid_open_price: 7496.61 on 19990428
mid_high_price: 7460.31 on 20111103
mid_low_price: 7490.91 on 20091127
mid_close_price: 7548.48 on 20100706
```

A圖表


10202
9526
8851
8175
7499
6824
6148
5473
4797
4121
3446
19990103
20001104
20020906
20040708
20060510
20080311
20100111
20111113
20130914
20150717

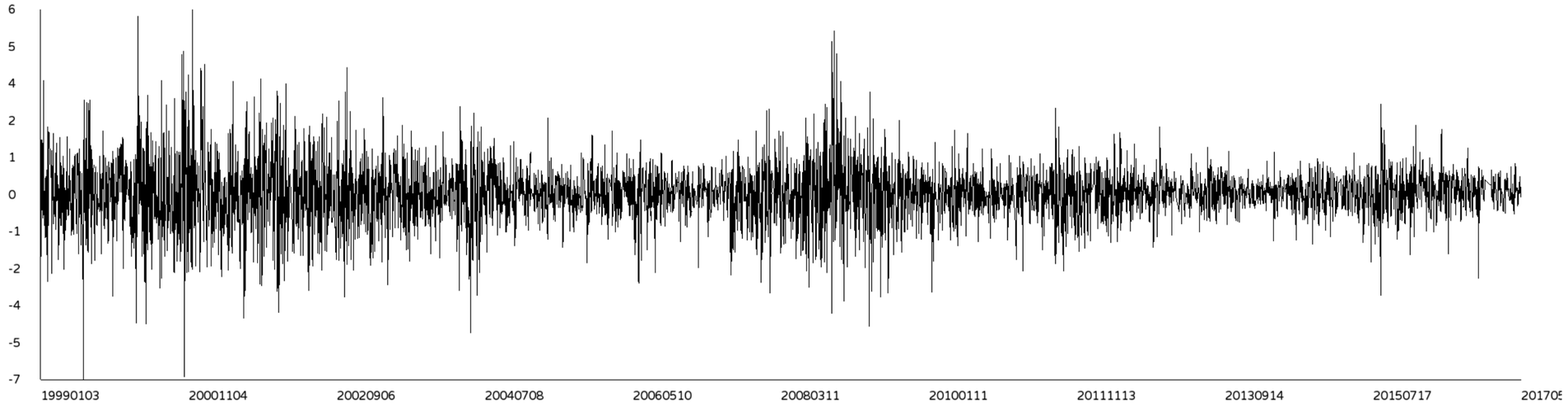
A-7

A圖表


6
5
4
2
1
0
-1
-2
-3
-5
-6
19990105
20001105
20020907
20040709
20060511
20080312
20100112
20111114
20130915
20150718

A-8

A圖表


6
5
4
2
1
0
-1
-2
-4
-5
-7
19990103
20001104
20020906
20040708
20060510
20080311
20100111
20111113
20130914
20150717

Q1-B-任務要求

1. 有幾個唯一日期

```
333     Treap<k_Bar_Date> treapB;
334     for (auto& it : k_line) {
335         treapB.insert(it);
336     }
337     Treap<k_Bar_Close> treap_close;
338     Treap<k_Bar_Date> treap_Date;
339     Treap<k_Bar_high> treap_high;
340     Treap<k_Bar_low> treap_low;
341     Treap<k_Bar_Open> treap_open;
342     int dat = 0;
343     linklist<k_Bar_Date> toListB = treapB.toList();
344     for (auto& it : toListB) {
345         if(dat%5==0){
346             treap_close.insert(k_Bar_Close(it.get_Date(), it.open, it.High, it.Low, it.close));
347             treap_Date.insert(k_Bar_Date(it.get_Date(), it.open, it.High, it.Low, it.close));
348             treap_high.insert(k_Bar_high(it.get_Date(), it.open, it.High, it.Low, it.close));
349             treap_low.insert(k_Bar_low(it.get_Date(), it.open, it.High, it.Low, it.close));
350             treap_open.insert(k_Bar_Open(it.get_Date(), it.open, it.High, it.Low, it.close));
351         }
352         dat++;
353     }       SiroKu_, 1 小時前 • refactor: Fix file path in main.cpp for output …
354     linklist<k_Bar_Close> toList_Close = treap_close.toList();
```

Q1-B-任務要求

1. 有幾個唯一日期
4. 中位價格及日期
5. 最大和最小回報及日期(每日)
6. 最大和最小回報及日期(日內)
10. 找以下4列價格的最大/最小/中位

開盤價、最高價、最低價、收盤價

Q1-B-輸出及圖表

```
1: 915
2: Small 10
20010927 3567.63
20011005 3585.46
20010920 3698.84
20011015 3712.82
20011022 3900.62
20021009 3947.61
20011029 4065.1
20011105 4080.51
20081121 4171.1
20021002 4171.76
3: Big 10
20170328 9876.45
20170406 9897.8
20170505 9899.94
20000412 9911.39
20170519 9947.62
20000406 9969.28
20170321 9972.49
20170512 9986.82
20000210 10057.7
20000217 10202.2
4: Median
20080130 7543.5
```

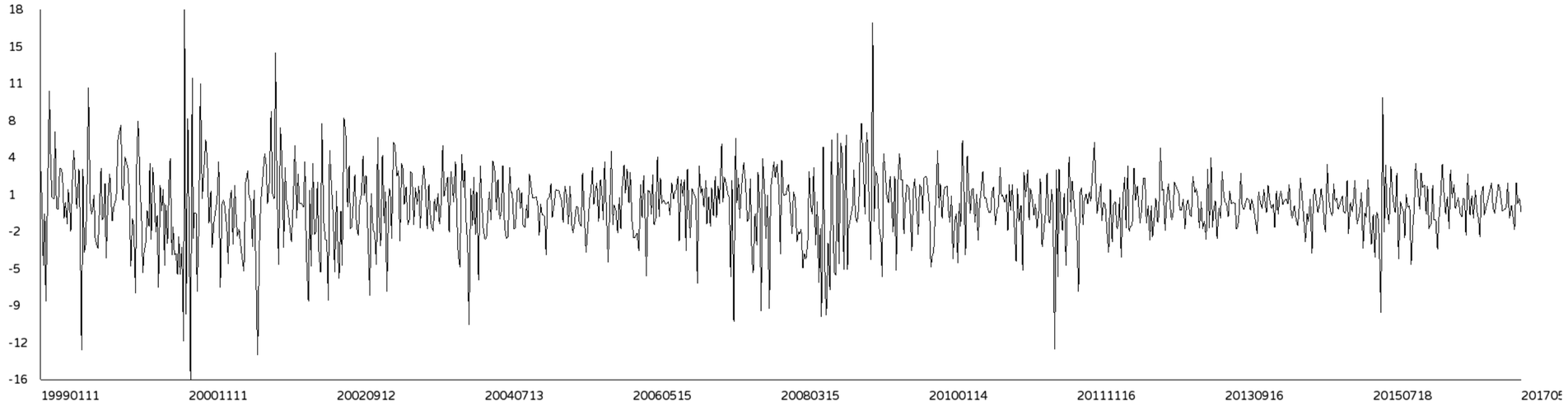
```
5Max: 18.5485
5Min: -16.0641

6Max: 6.13798
6Min: -7.20747
10:
max_open_price: 10225.8 on 20000406
mid_open_price: 7530.27 on 19990511
min_open_price: 3502.37 on 20011005
max_high_price: 10329 on 20000406
mid_high_price: 7597.79 on 20091009
min_high_price: 3585.64 on 20011005
max_low_price: 10089.9 on 20000217
mid_low_price: 7466.49 on 20121129
min_low_price: 3467.94 on 20011005
max_close_price: 10202.2 on 20000217
min_close_price: 3567.63 on 20010927
mid_close_price: 7543.5 on 20080130
```

B圖表

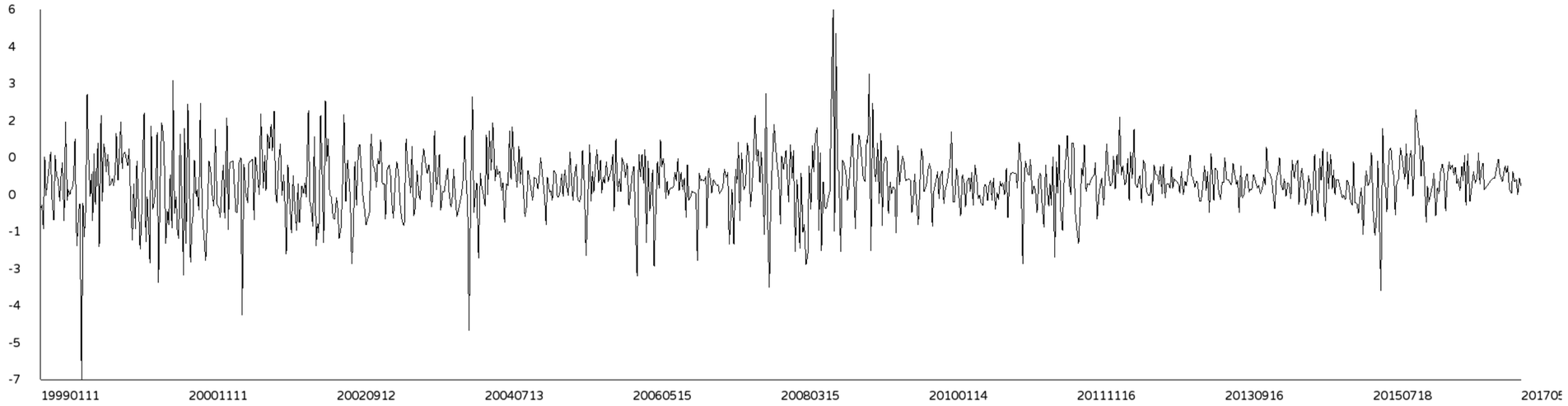

10202
9538
8875
8211
7548
6884
6221
5558
4894
4231
3567
19990103
20001104
20020906
20040708
20060510
20080311
20100111
20111113
20130914
20150717

B-7

B圖表


18
15
11
8
4
1
-2
-5
-9
-12
-16
19990111
20001111
20020912
20040713
20060515
20080315
20100114
20111116
20130916
20150718

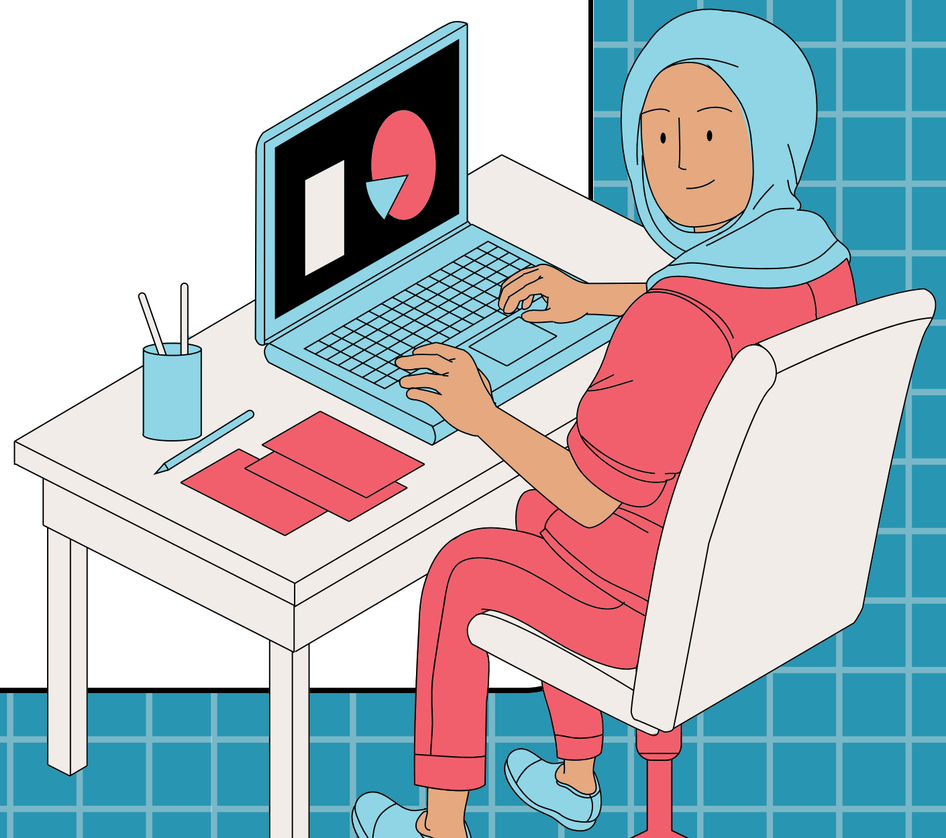
B圖表


6
4
3
2
0
0
-1
-3
-4
-5
-7
19990111
20001111
20020912
20040713
20060515
20080315
20100114
20111116
20130916
20150718

Q2-任務要求

- 1.有幾個獨特的產品?
- 2.TXO_1000_201706_P 是否存在
- 3. TXO_5500_201706_C是否存在
- 4. TXO_9900_201706_C是否存在

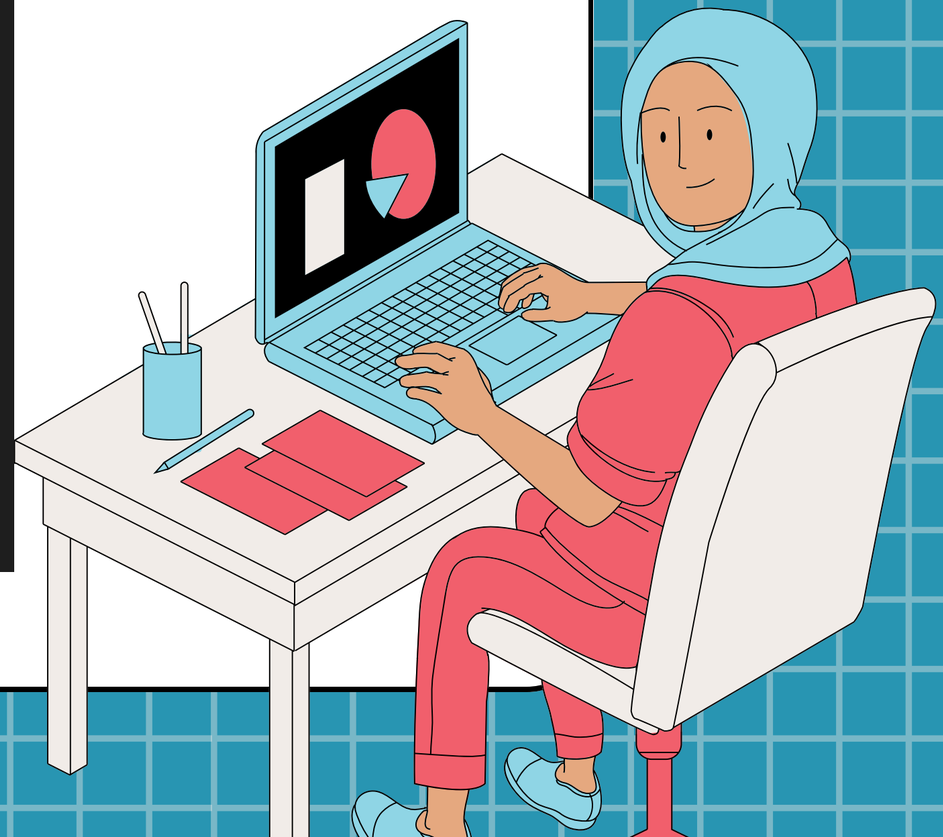
```
196     cout<<"totle number of unique products: "<<uniProduct.size()<<endl;
197     cout<<"totle number of transactions: "<<treap.size()<<endl;
198     cout<<"TXO_1000_201706_P exists?: "<<endl;
199     uniProduct.find("TXO 1000.000000 201706 P")? cout<<"Yes"<<endl: cout<<"No"<<endl;
200     cout<<"TXO_9500_201706_C exists?: "<<endl;
201     uniProduct.find("TXO 9500.000000 201706 C")? cout<<"Yes"<<endl: cout<<"No"<<endl;
202     cout<<"TXO_5500_201706_C exists?: "<<endl;
203     uniProduct.find("TXO 5500.000000 201706 C")? cout<<"Yes"<<endl: cout<<"No"<<endl;
204     cout << "TXO_9900_201705_C exists?: " << endl;
205     uniProduct.find("TXO 9900.000000 201705 C")? cout<<"Yes"<<endl: cout<<"No"<<endl;
```



Q2-任務要求

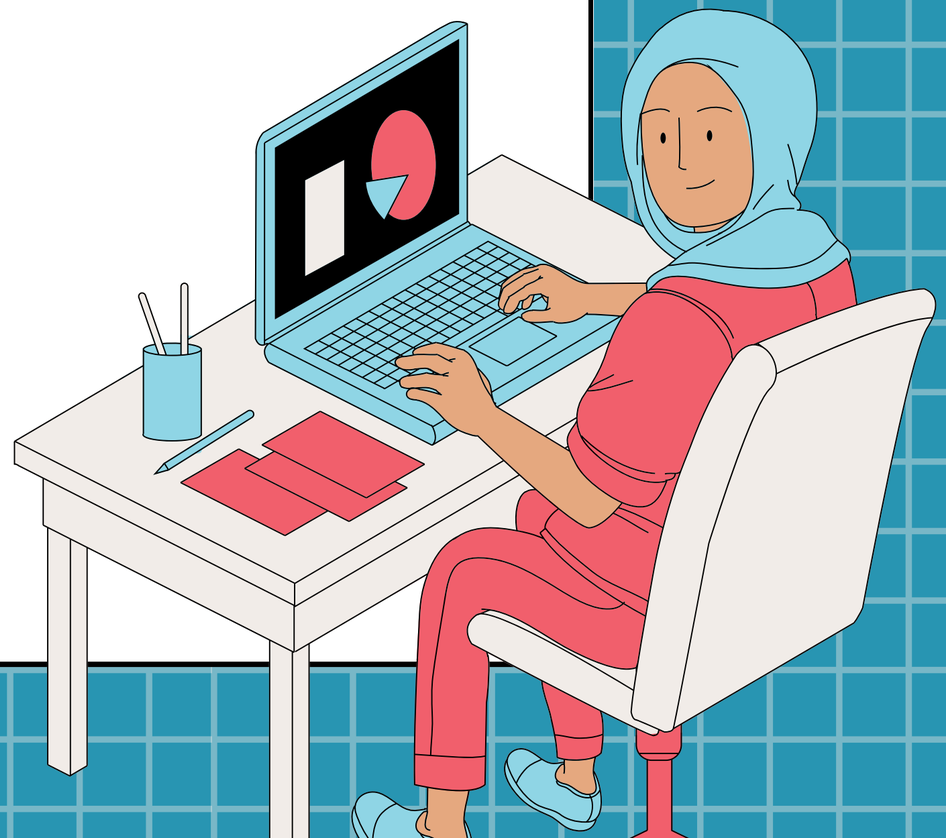
- 找到TXO_9900_201705_C
- 6找到產品的10個最小價格
- 7.找到產品的10個最大價格
- 8.找到產品的中位價格

```
208     treap.slip_By_Value(a, TXO_9900_201705_C, k_Bar(0, "TXO", 9900, "201705", 'C', 0, 0, 0, 0));
209     TXO_9900_201705_C.slip_By_Value(TXO_9900_201705_C, b, k_Bar(0, "TXO", 9900, "201705", 'C', 0, FLT_MAX, 0, 0));
210     cout<<TXO_9900_201705_C.size()<<endl;
211
212     Treap<k_Bar> e, f;
213     TXO_9900_201705_C.slip_By_Size(e, f, 10);
214     cout<<"Small 10"<<endl;
215     cout<<e<<endl;
216     cout<<"Big 10"<<endl;
217
218     TXO_9900_201705_C.slip_By_Size(e, f, TXO_9900_201705_C.size()-10);
219     cout<<f<<endl;
220     cout<<"Middle"<<endl;
221     if(TXO_9900_201705_C.size()&1){
222         cout<<TXO_9900_201705_C[TXO_9900_201705_C.size()/2]<<endl;
223     }
224     else{
225         auto it = TXO_9900_201705_C[TXO_9900_201705_C.size()/2];
226         auto it2 = TXO_9900_201705_C[TXO_9900_201705_C.size()/2-1];
227         cout << it << endl<< it2<<endl;
228         cout<<"Average: "<<(it.get_transactionPrice()+it2.get_transactionPrice())/2<<endl;
229     }
```

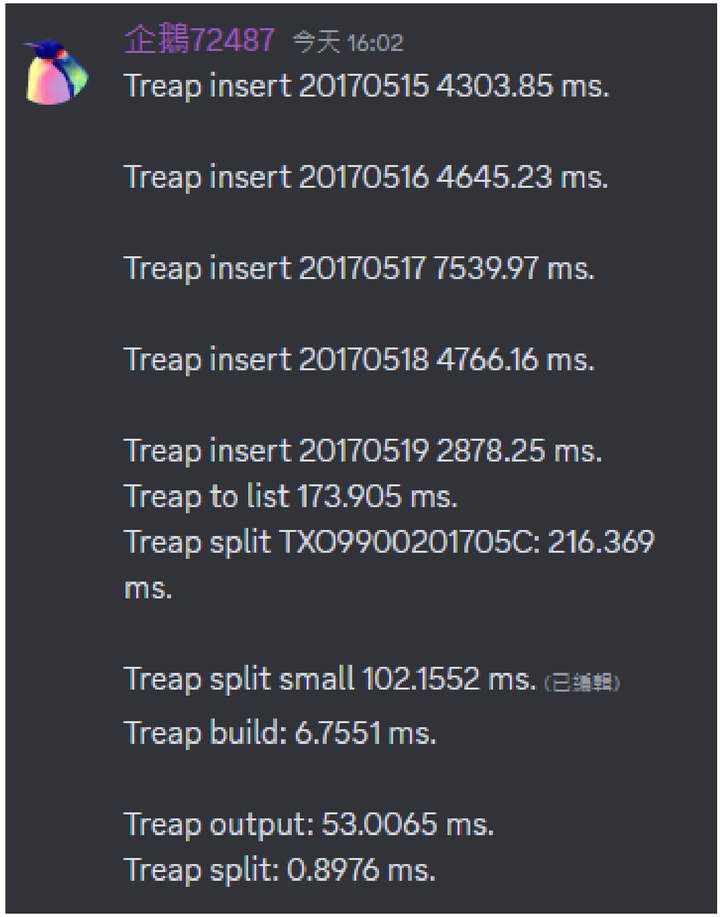


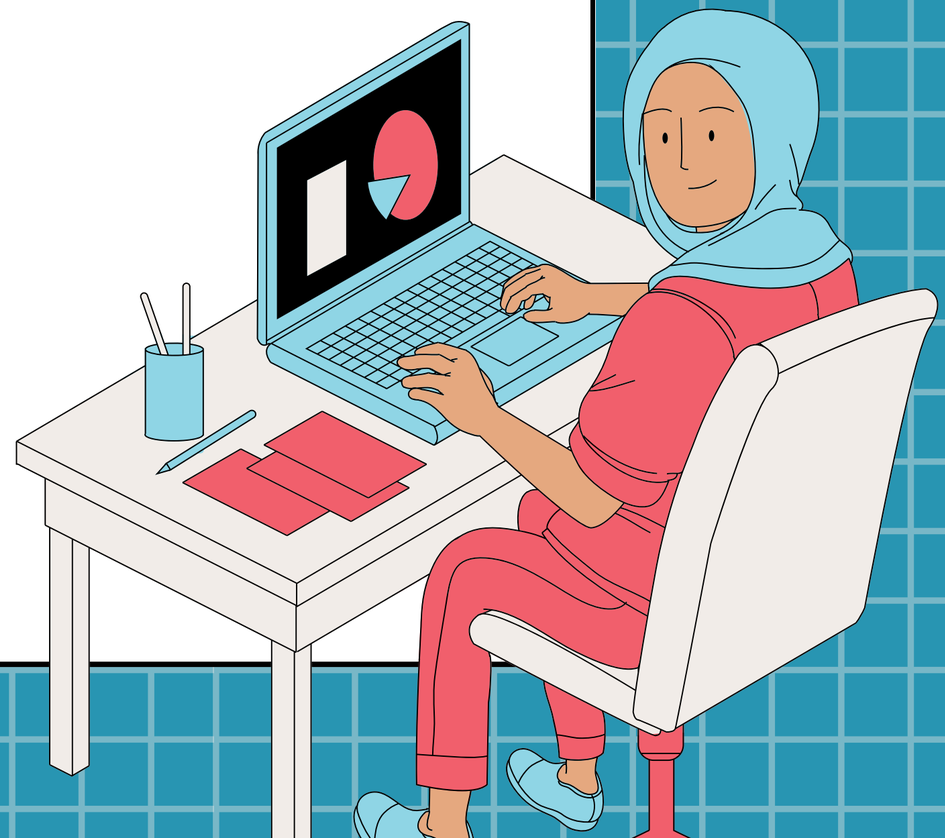
Q2-輸出及圖表

```
Total number of unique products: 1029
TXO_1000_201706_P exists: No
TXO_9500_201706_C exists: Yes
GIO_5500_201706_C exists: No
Top 10 smallest prices with times:
20170517 TXO 9900 201705 C 114907 76 1 0
20170517 TXO 9900 201705 C 114914 76 1 0
20170517 TXO 9900 201705 C 114914 76 4 0
20170517 TXO 9900 201705 C 114914 76 1 0
20170517 TXO 9900 201705 C 114914 76 3 0
20170517 TXO 9900 201705 C 114914 76 1 0
20170517 TXO 9900 201705 C 114914 76 10 0
20170517 TXO 9900 201705 C 114907 76 1 0
20170517 TXO 9900 201705 C 114907 77 17 0
20170517 TXO 9900 201705 C 114910 77 1 0

Top 10 largest prices with times:
20170516 TXO 9900 201705 C 90508 152 1 0
20170516 TXO 9900 201705 C 90508 152 2 0
20170516 TXO 9900 201705 C 90502 153 3 0
20170516 TXO 9900 201705 C 90503 153 1 0
20170516 TXO 9900 201705 C 90502 153 1 0
20170516 TXO 9900 201705 C 90502 153 5 0
20170516 TXO 9900 201705 C 90503 154 1 0
20170516 TXO 9900 201705 C 90503 154 2 0
20170516 TXO 9900 201705 C 90502 154 2 0
20170516 TXO 9900 201705 C 90502 154 1 0

Middle
112 at 20170515
112 at 20170515      You, 33 秒前 • Uncom
Average: 112

Max return: 0.0131579 at 20170517
Min return: 0 at 20170517

```

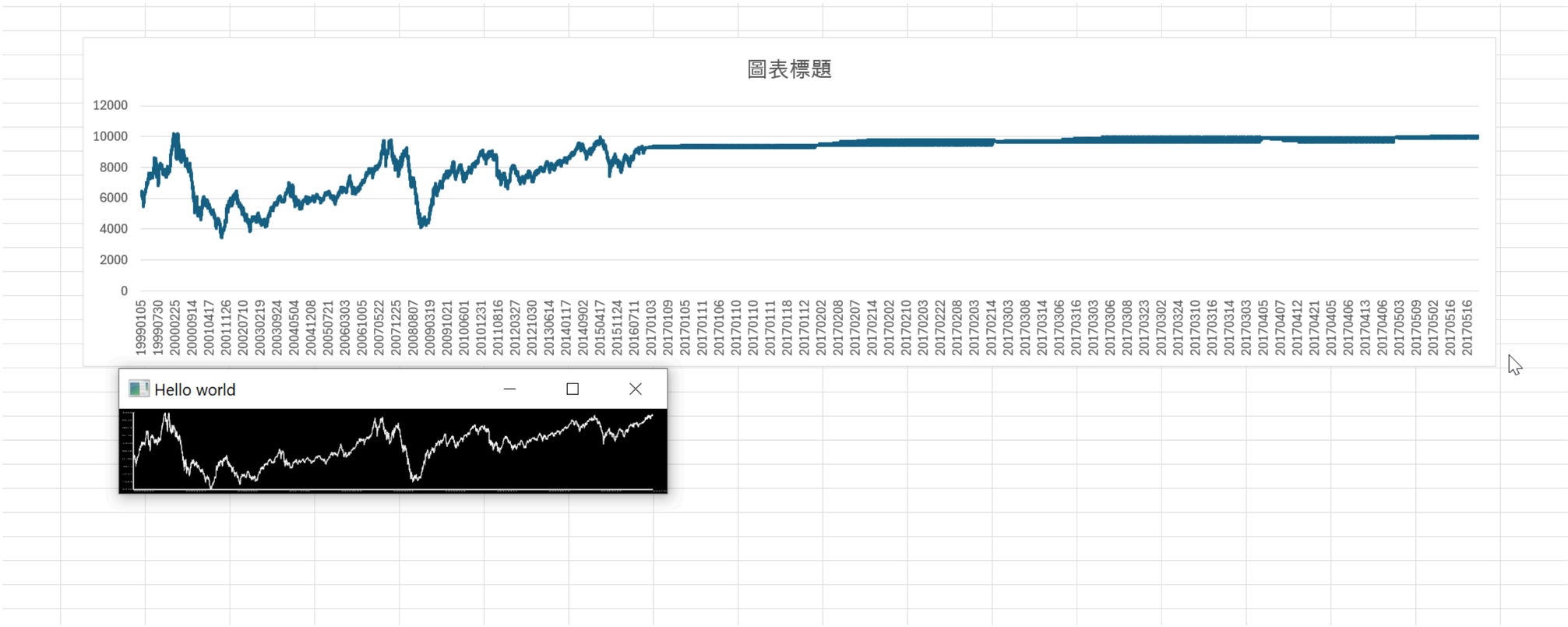


效能


企鵝72487 今天 16:02
Treap insert 20170515 4303.85 ms.
Treap insert 20170516 4645.23 ms.
Treap insert 20170517 7539.97 ms.
Treap insert 20170518 4766.16 ms.
Treap insert 20170519 2878.25 ms.
Treap to list 173.905 ms.
Treap split TXO9900201705C: 216.369 ms.
Treap split small 102.1552 ms. (已編輯)
Treap build: 6.7551 ms.
Treap output: 53.0065 ms.
Treap split: 0.8976 ms.

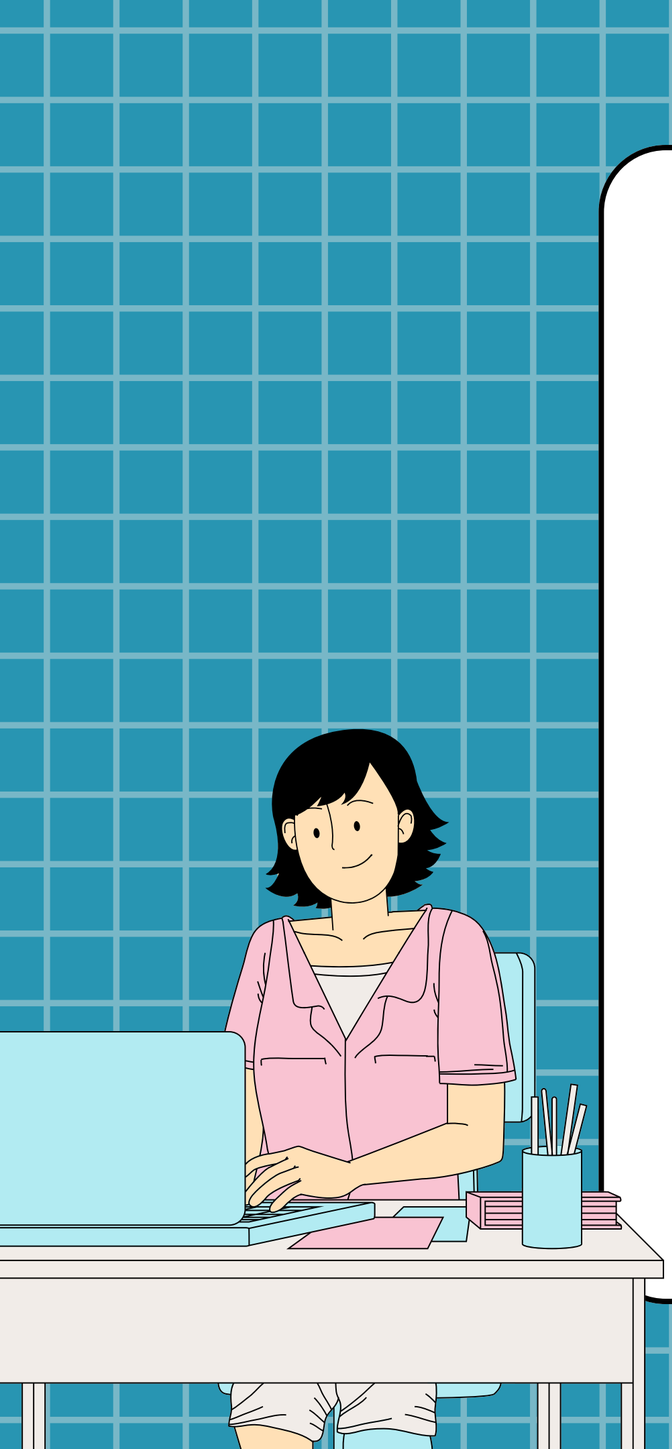


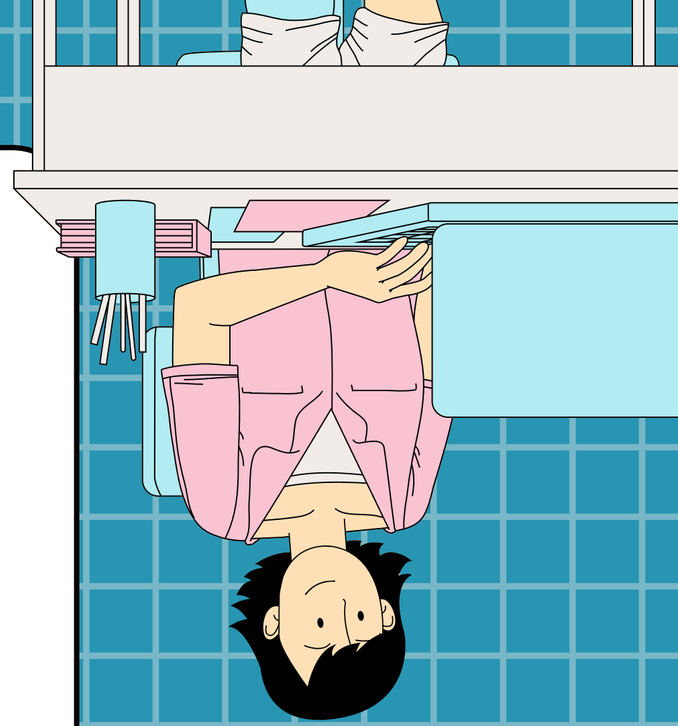
遇到的困難


圖表標題
12000
10000
8000
6000
4000
2000
0
19990105
19990730
20000225
20000914
20010417
20011126
20020710
20030219
20030924
20040504
20041208
20050721
20060303
20061005
20070522
20071225
20080807
20090319
20091021
20100601
20101231
20110816
20120327
20121030
20130614
20140117
20140902
20150417
20151124
20160711
20170103
20170109
20170105
20170111
20170106
20170110
20170110
20170111
20170118
20170112
20170202
20170208
20170207
20170214
20170202
20170210
20170203
20170222
20170208
20170203
20170214
20170303
20170308
20170314
20170306
20170316
20170303
20170306
20170308
20170323
20170302
20170324
20170310
20170316
20170314
20170303
20170405
20170407
20170412
20170421
20170405
20170406
20170413
20170406
20170503
20170509
20170502
20170516
20170516
Hello world

分工表

A1115513	報告ppt/word，資料閱讀
A1115530	使用資料結構演算法使輸出符合題意
A1115531	建構資料結構與演算法
A1115532	產生線圖 使用Python & Excel校驗答案





參考資料

Treap

Heap

字體-huninn

THANKS FOR LISTENING