```
Algorithms
Containers
Functions
Iterators
```

different algorithms and different functions that do exist in

```
1  3 2
2  1 2
3  2 3
```

output.txt

```
[Finished in 7.3s]
```

5:34 / 1:07:36 • C STL >

```cpp
// Pairs
void explainPair() {

    pair<int, int> p = {1, 3};


    cout << p.first << " " << p.second;


    pair<int, pair<int, int>> p = {1, {3, 4}};


    cout << p.first << " " << p.second.second << " " << p.second.first;


    pair<int, int> arr[] = { {1, 2}, {2, 5}, {5, 1}};


    cout << arr[1].second;

}
```

```
1  3 2
2  1 2
3  2 3
```

output.txt

functions that do exist in c plus plus
stl so before moving on

[Finished in 7.3s]

5:37 / 1:07:36 • Pairs >

```
1  3 2
2  1 2
3  2 3
```

```cpp
void explainVector() {

    vector<int> v;

    v.push_back(1);
    v.emplace_back(2);

    vector<pair<int, int>>vec;

    v.push_back({1, 2});
    v.emplace_back(1, 2);


    vector<int> v(5, 100);


    vector<int> v(5);

    vector<int> v1(5, 20);
    vector<int> v2(v1);



    vector<int>::iterator it = v.begin();

    it++;
    cout << *(it) << " ";
```

{} → {1} → {1,2}

{1,2}

(1,2)

output.txt

[Finished in 7.3s]

13:15 / 1:07:36 · Vectors >

```
void explainVector() {

    vector<int> v;

    v.push_back(1);
    v.emplace_back(2);

    vector<pair<int, int>>vec;

    v.push_back({1, 2});
    v.emplace_back(1, 2);


    vector<int> v(5, 100);


    vector<int> v(5);

    vector<int> v1(5, 20);
    vector<int> v2(v1);


    vector<int>::iterator it = v.begin();

    it++;
    cout << *(it) << " ";
```

```
1   3 2
2   1 2
3   2 3
```

output.txt

[Finished in 7.3s]

```
117
118     vector<int>::iterator it = v.begin();
119
120     it++;
121     cout << *(it) << " ";
122
123
124     it = it + 2;
125     cout << *(it) << " ";
126
127     vector<int>::iterator it = v.end();
128
129     vector<int>::iterator it = v.rend();
130
131     vector<int>::iterator it = v.rbegin();
132
133
134
135     cout << v[0] << " " << v.at(0);
136
137     cout << v.back() << " ";
138
139
140
141
142     for (vector<int>::iterator it = v.begin(); it != v.end(); it++) {
143         cout << *(it) << " ";
144     }
145
146
147     for (auto it = v.begin(); it != v.end(); it++) {
148         cout << *(it) << " ";
149     }
```

V → {20, 10, 15, 5, 7}
0 1 2 3 4

v[1] ← 10

v[3] ← 5

```
1  3 2
2  1 2
3  2 3
```

output.txt

[Finished in 7.3s]

```
117
118     vector<int>::iterator it = v.begin();
119
120     it++;
121     cout << *(it) << " ";
122
123
124     it = it + 2;
125     cout << *(it) << " ";
126
127     vector<int>::iterator it = v.end();
128
129     vector<int>::iterator it = v.rend();
130
131     vector<int>::iterator it = v.rbegin();
132
133
134
135     cout << v[0] << " " << v.at(0);
136
137     cout << v.back() << " ";
138
139
140
141
142     for (vector<int>::iterator it = v.begin(); it != v.end(); it++) {
143         cout << *(it) << " ";
144     }
145
146
147     for (auto it = v.begin(); it != v.end(); it++) {
148         cout << *(it) << " ";
149     }
```

```
1  3 2
2  1 2
3  2 3
```

output.txt

[Finished in 7.3s]

```
117
118     vector<int>::iterator it = v.begin();
119
120     it++;
121     cout << *(it) << " ";
122
123
124     it = it + 2;
125     cout << *(it) << " ";
126
127     vector<int>::iterator it = v.end();
128
129     vector<int>::iterator it = v.rend();
130
131     vector<int>::iterator it = v.rbegin();
132
133
134
135     cout << v[0] << " " << v.at(0);
136
137     cout << v.back() << " ";
138
139
140
141
142     for (vector<int>::iterator it = v.begin(); it != v.end(); it++) {
143         cout << *(it) << " ";
144     }
145
146
147     for (auto it = v.begin(); it != v.end(); it++) {
148         cout << *(it) << " ";
149     }
```

```
1   3 2
2   1 2
3   2 3
```

output.txt

1

[Finished in 7.3s]

20:20 / 1:07:36 • End and Reverse >

```
117
118     vector<int>::iterator it = v.begin();
119
120     it++;
121     cout << *(it) << " ";
122
123
124     it = it + 2;
125     cout << *(it) << " ";
126
127     vector<int>::iterator it = v.end();
128
129     vector<int>::iterator it = v.rend();
130
131     vector<int>::iterator it = v.rbegin();
132
133
134
135     cout << v[0] << " " << v.at(0);
136
137     cout << v.back() << " ";
138
139
140
141
142     for (vector<int>::iterator it = v.begin(); it != v.end(); it++) {
143         cout << *(it) << " ";
144     }
145
146
147     for (auto it = v.begin(); it != v.end(); it++) {
148         cout << *(it) << " ";
149     }
```

```
1   3 2
2   1 2
3   2 3
```

output.txt

1

{10, 20, 30, 40}

it++

{40 30 20 10}

[Finished in 7.3s]

21:33 / 1:07:36 · End and Reverse >

```cpp
    vector<int>::iterator it = v.begin();

    it++;
    cout << *(it) << " ";


    it = it + 2;
    cout << *(it) << " ";

    vector<int>::iterator it = v.end();

    vector<int>::iterator it = v.rend();

    vector<int>::iterator it = v.rbegin();


    cout << v[0] << " " << v.at(0);

    cout << v.back() << " ";


    for (vector<int>::iterator it = v.begin(); it != v.end(); it++) {
        cout << *(it) << " ";
    }


    for (auto it = v.begin(); it != v.end(); it++) {
        cout << *(it) << " ";
    }
```

{10, 20, 30}

```
1 3 2
2 1 2
3 2 3
```

output.txt

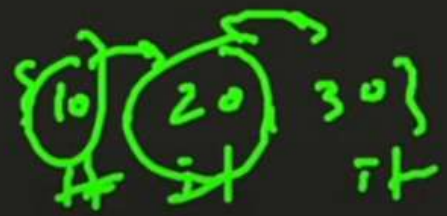
1

[Finished in 7.3s]

```cpp
    vector<int>::iterator it = v.end();

    vector<int>::iterator it = v.rend();

    vector<int>::iterator it = v.rbegin();



    cout << v[0] << " " << v.at(0);

    cout << v.back() << " ";



    for (vector<int>::iterator it = v.begin(); it != v.end(); it++) {
        cout << *(it) << " ";
    }


    for (auto it = v.begin(); it != v.end(); it++) {
        cout << *(it) << " ";
    }

    for (auto it : v) {
        cout << it << " ";
    }

    // {10, 20, 12, 23}
    v.erase(v.begin()+1);

    // {10, 20, 12, 23, 35}
    v.erase(v.begin() + 2, v.begin() + 4); // // {10, 20, 35} [start, end)

[Finished in 7.3s]
```

```
1  3 2
2  1 2
3  2 3
```

output.txt



```
    v.erase(v.begin()+1);

    // {10, 20, 12, 23, 35}
    v.erase(v.begin() + 2, v.begin() + 4); // // {10, 20, 35} [start, end)


    //Insert function

    vector<int>v(2, 100); // {100, 100}
    v.insert(v.begin(), 300); // {300, 100, 100};
    v.insert(v.begin() + 1, 2, 10); // {300, 10, 10, 100, 100}

    vector<int> copy(2, 50); // {50, 50}
    v.insert(v.begin(), copy.begin(), copy.end()); // {50, 50, 300, 10, 10, 100, 100}

    // {10, 20}
    cout << v.size(); // 2

    //{10, 20}
    v.pop_back(); // {10}

    // v1 -> {10, 20}
    // v2 -> {30, 40}
    v1.swap(v2); // v1 -> {30, 40} , v2 -> {10, 20}

    v.clear(); // erases the entire vector

    cout << v.empty();
}

void explainList() {
    list<int> ls;
```

```
1  3 2
2  1 2
3  2 3
```

output.txt

1

1 → False

0 → True

[Finished in 7.3s]

```
199
200
201
202
203  void explainList() {
204      list<int> ls;
205
206      ls.push_back(2); // {2}
207      ls.emplace_back(4); // {2, 4}
208
209      ls.push_front(5); // {5, 2, 4};
210
211      ls.emplace_front(); {2, 4};
212
213      // rest functions same as vector
214      // begin, end, rbegin, rend, clear, insert, size, swap
215  }
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230  void explainDeque() {
231
```

insert

```
1  3 2
2  1 2
3  2 3
```

output.txt

1

[Finished in 7.3s]

```cpp
void explainDeque() {

    deque<int>dq;
    dq.push_back(1); // {1}
    dq.emplace_back(2); // {1, 2}
     dq.push_front(4); // {4, 1, 2}
     dq.emplace_front(3); // {3, 4, 1, 2}

    dq.pop_back(); // {3, 4, 1}
    dq.pop_front(); // {4, 1}

    dq.back();

    dq.front();

    // rest functions same as vector
    // begin, end, rbegin, rend, clear, insert, size, swap
}
```

```
3 2
1 2
2 3
```

output.txt

1

[Finished in 7.3s]

32:01 / 1:07:36 · DQ >

```cpp
void explainStack() {
    stack<int> st;
    st.push(1); // {1}
    st.push(2); // {2, 1}
    st.push(3); // {3, 2, 1}
    st.push(3); // {3, 3, 2, 1}
    st.emplace(5); // {5, 3, 3, 2, 1}

    cout << st.top(); // prints 5  "** st[2] is invalid **"

    st.pop(); // st looks like {3, 3, 2, 1}

    cout << st.top(); // 3

    cout << st.size(); // 4

    cout << st.empty();

    stack<int>st1, st2;
    st1.swap(st2);

}
```

LIFO

Last In First Out

false

4

push
pop
top

```
1  3 2
2  1 2
3  2 3
```

output.txt

1

[Finished in 7.3s]

33:49 / 1:07:36 · DQ >

```cpp
void explainStack() {
    stack<int> st;
    st.push(1); // {1}
    st.push(2); // {2, 1}
    st.push(3); // {3, 2, 1}
    st.push(3); // {3, 3, 2, 1}
    st.emplace(5); // {5, 3, 3, 2, 1}

    cout << st.top(); // prints 5  "** st[2] is invalid **"

    st.pop(); // st looks like {3, 3, 2, 1}

    cout << st.top(); // 3

    cout << st.size(); // 4

    cout << st.empty();

    stack<int> st1, st2;
    st1.swap(st2);

}
```

LIFO

Last In First Out

push
pop
top

false

4

O(1)

```
1  3 2
2  1 2
3  2 3
```

output.txt

1

[Finished in 7.3s]

```
288
289
290
291
292
293 void explainQueue() {
294     queue<int> q;
295     q.push(1); // {1}
296     q.push(2); // {1, 2}
297     q.emplace(4); // {1, 2, 4}
298
299     q.back() += 5
300
301     cout << q.back(); // prints 9
302
303     // Q is {1, 2, 9}
304     cout << q.front(); // prints 1
305
306     q.pop(); // {2, 9}
307
308     cout << q.front(); // prints 2
309
310     // size swap empty same as stack
311 }
312
313
314
315
316
317
318
319
320
321
```

FIFO

First In First out

```
1  3 2
2  1 2
3  2 3
```

output.txt

1

[Finished in 7.3s]

35:52 / 1:07:36 · Queue >

```cpp
void explainPQ() {
    priority_queue<int>pq;

    pq.push(5); // {5}
    pq.push(2); // {5, 2}
    pq.push(8); // {8, 5, 2}
    pq.emplace(10); // {10, 8, 5, 2}

    cout << pq.top(); // prints 10

    pq.pop(); // {8, 5, 2}

    cout << pq.top(); // prints 8

    // size swap empty function same as others

    // Minimum Heap
    priority_queue<int, vector<int>, greater<int>> pq;
    pq.push(5); // {5}
    pq.push(2); // {2, 5}
    pq.push(8); // {2, 5, 8}
    pq.emplace(10); // {2, 5, 8, 10}

    cout << pq.top(); // prints 2

}
```

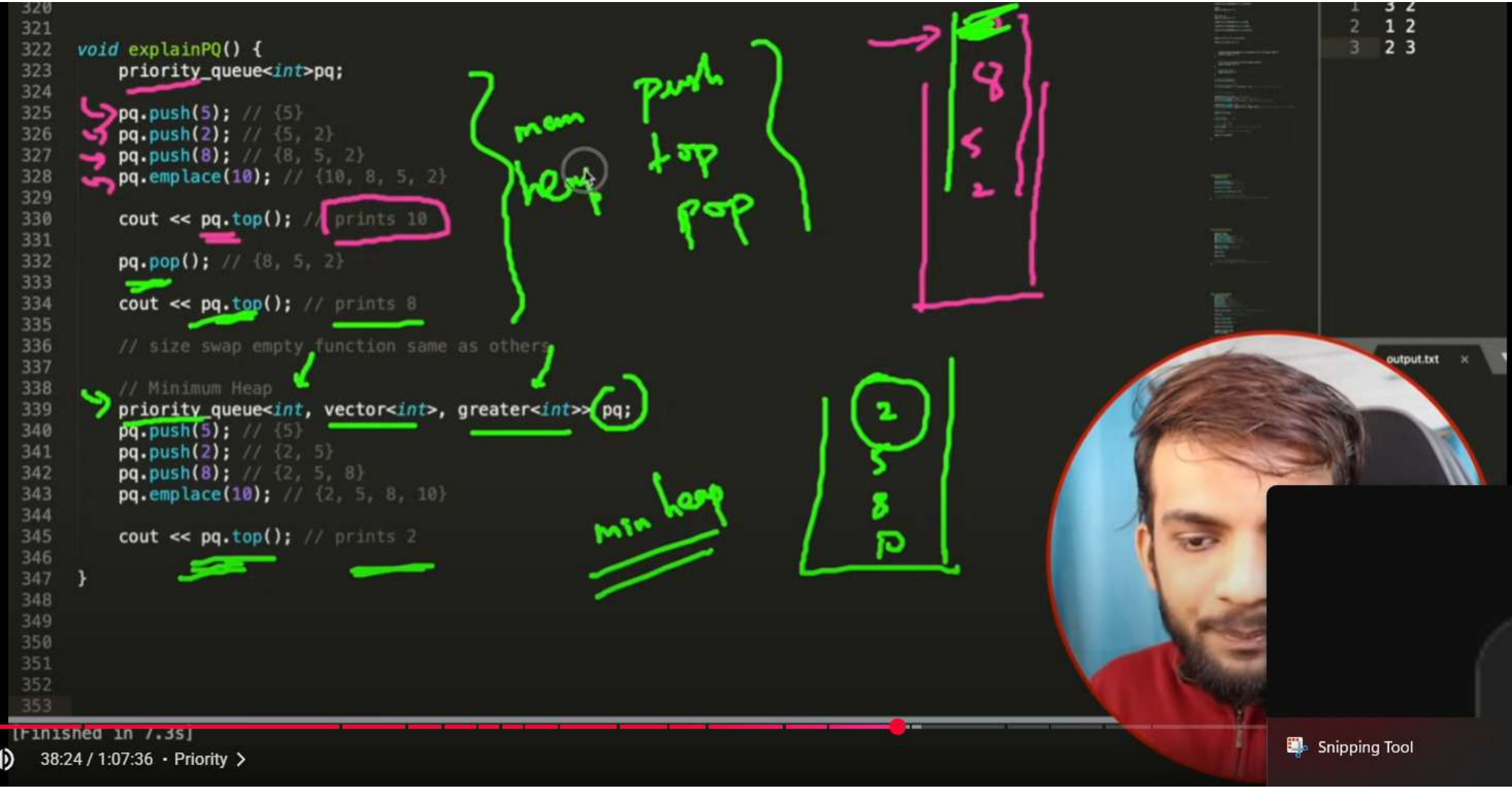
push
top
pop


```
3 2
1 2
2 3
```

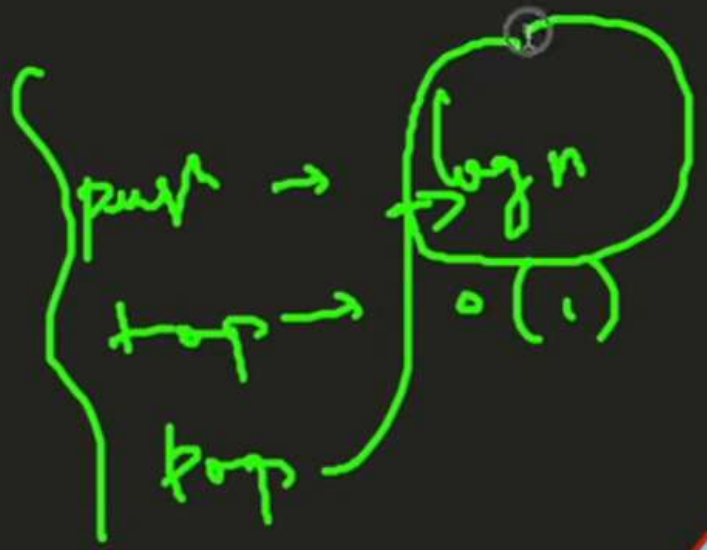
output.txt

[Finished in 7.3s]

```cpp
void explainPQ() {
    priority_queue<int>pq;

    pq.push(5); // {5}
    pq.push(2); // {5, 2}
    pq.push(8); // {8, 5, 2}
    pq.emplace(10); // {10, 8, 5, 2}

    cout << pq.top(); // prints 10

    pq.pop(); // {8, 5, 2}

    cout << pq.top(); // prints 8

    // size swap empty function same as others

    // Minimum Heap
    priority_queue<int, vector<int>, greater<int>> pq;
    pq.push(5); // {5}
    pq.push(2); // {2, 5}
    pq.push(8); // {2, 5, 8}
    pq.emplace(10); // {2, 5, 8, 10}

    cout << pq.top(); // prints 2

}
```



```cpp
void explainPQ() {
    priority_queue<int>pq;

    pq.push(5); // {5}
    pq.push(2); // {5, 2}
    pq.push(8); // {8, 5, 2}
    pq.emplace(10); // {10, 8, 5, 2}

    cout << pq.top(); // prints 10

    pq.pop(); // {8, 5, 2}

    cout << pq.top(); // prints 8

    // size swap empty function same as others

    // Minimum Heap
    priority_queue<int, vector<int>, greater<int>> pq;
    pq.push(5); // {5}
    pq.push(2); // {2, 5}
    pq.push(8); // {2, 5, 8}
    pq.emplace(10); // {2, 5, 8, 10}

    cout << pq.top(); // prints 2

}
```




```
1  3 2
2  1 2
3  2 3
```

output.txt

[Finished in 7.3s]

```cpp
void explainSet() {
    set<int>st;
    st.insert(1); // {1}
    st.emplace(2); // {1, 2}
    st.insert(2); // {1, 2}
    st.insert(4); // {1, 2, 4}
    st.insert(3); // {1, 2, 3, 4}

    // Functionality of insert in vector
    // can be used also, that only increases
    // efficiency

    // begin(), end(), rbegin(), rend(), size(),
    // empty() and swap() are same as those of above

    // {1, 2, 3, 4, 5}
    auto it = st.find(3);

    // {1, 2, 3, 4, 5}
    auto it = st.find(6);

    // {1, 4, 5}
    st.erase(5); // erases 5 // takes logarithmic time

    int cnt = st.count(1);


    auto it = st.find(3);
    st.erase(it); // it takes constant time
```

```
1  3 2
2  1 2
3  2 3
```

[Finished in 7.3s]

41:24 / 1:07:36 • Set >