Today's plan

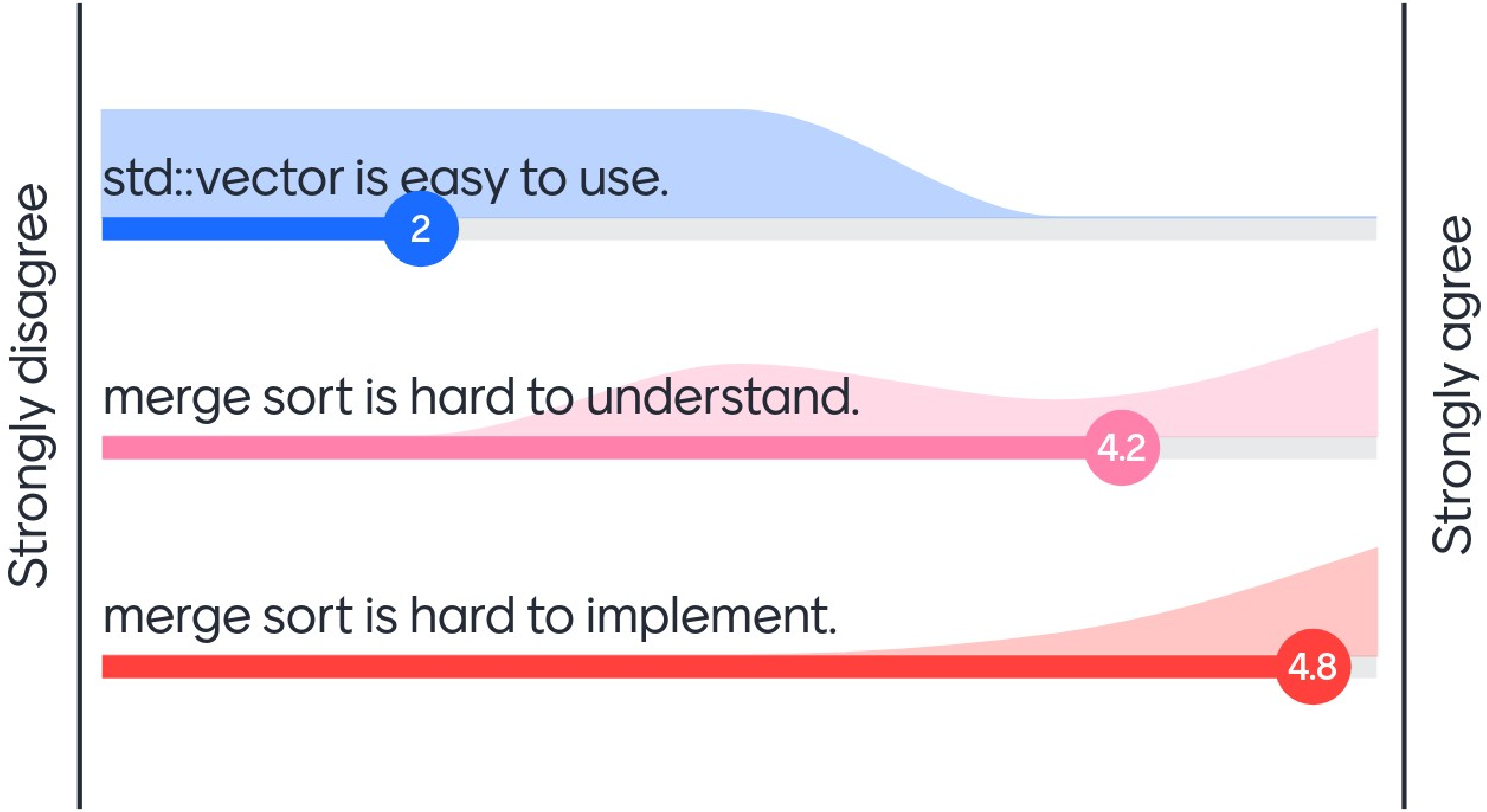
→ Review Ex 8-1

→ Recap of today's content

→ Ex 8-2

What's your opinion?

Mentimeter



6

Ex 8-1: Read an integer in C++

In C, we declare a variable of int type:

```
int temp;
```

Then, we read from the console input:

```
scanf("%d", &temp)
```

In C++, to read from the console, we use:

```
std::iostream
```

Instead of the placeholder syntax in C, we do:

```
std::cin >> temp;
```

If there are multiple inputs, we do a chain, e.g.

```
std::cin >> temp1 >> temp2 >> temp3;
```

Ex 8-1: How to use **std::vector**

We declare an integer vector by:

```
std::vector<int> vec;
```

or

```
std::vector<int> vec(vec_size);
```

or

```
std::vector<int> vec = {1, 2};
```

We resize the vector by:

```
vec.resize(new_size);
```

We get the vector size by:

```
vec.size();
```

Ex 8-1: Merge sort - split

```
void sort(vector<int>* vec);
```

Compute the mid index:

```
int mid_idx = vec->size() / 2;
```

Nothing to do if mid_idx is 0:

```
if (!mid_idx) return; // the base case
```

Otherwise, declare and define two int vectors:

```
vector<int> left, right;
```

and then split *vec into left and right

```
for(…) {…} // the split logic
```

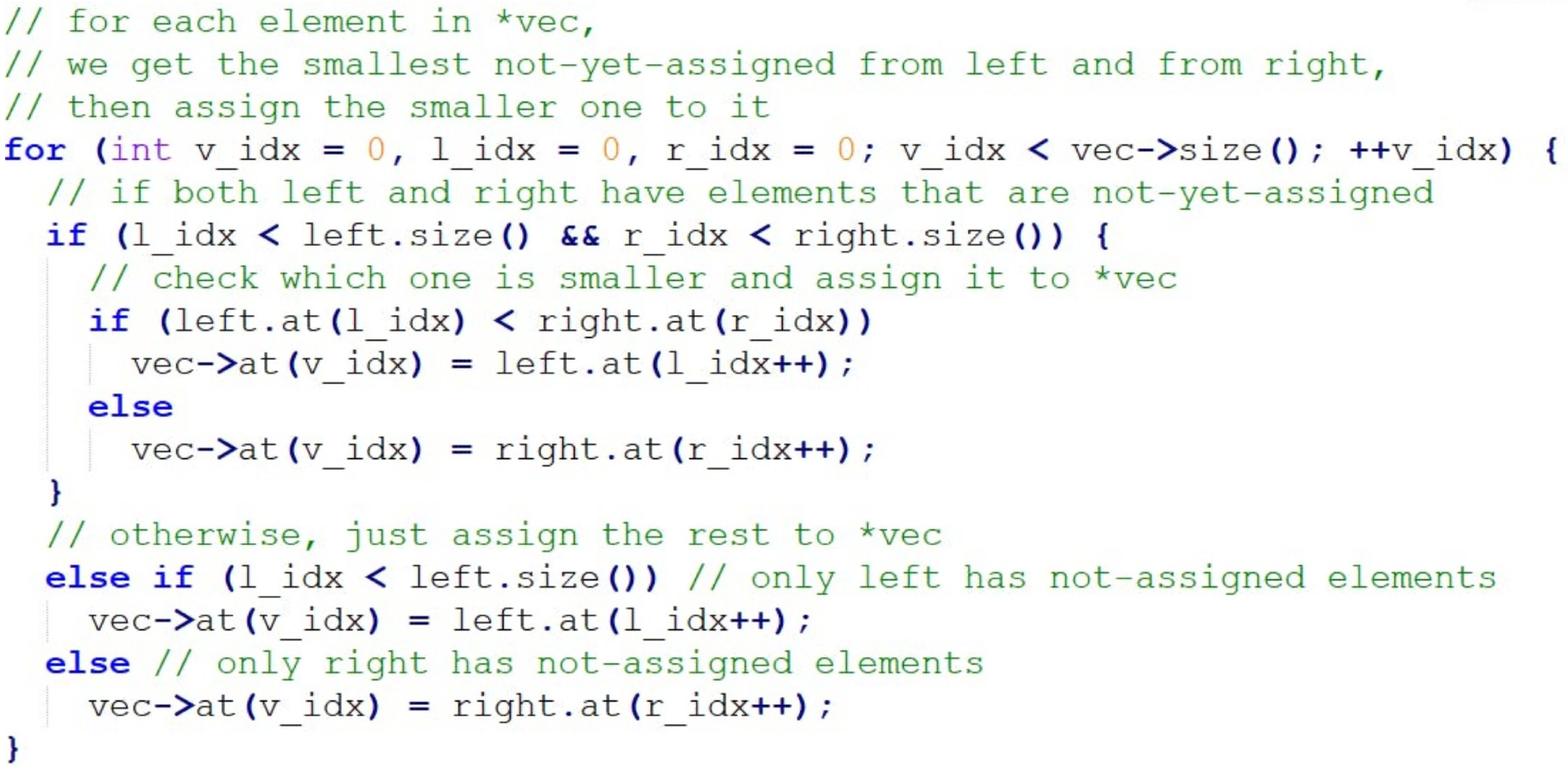
♥ ? 👍 👎

Ex 8-1: Merge sort - recursion calls

After filling left and right, we use recursion calls to sort left and right:

```
sort(&left);
sort(&right);
```

Now, we have left and right are sorted.

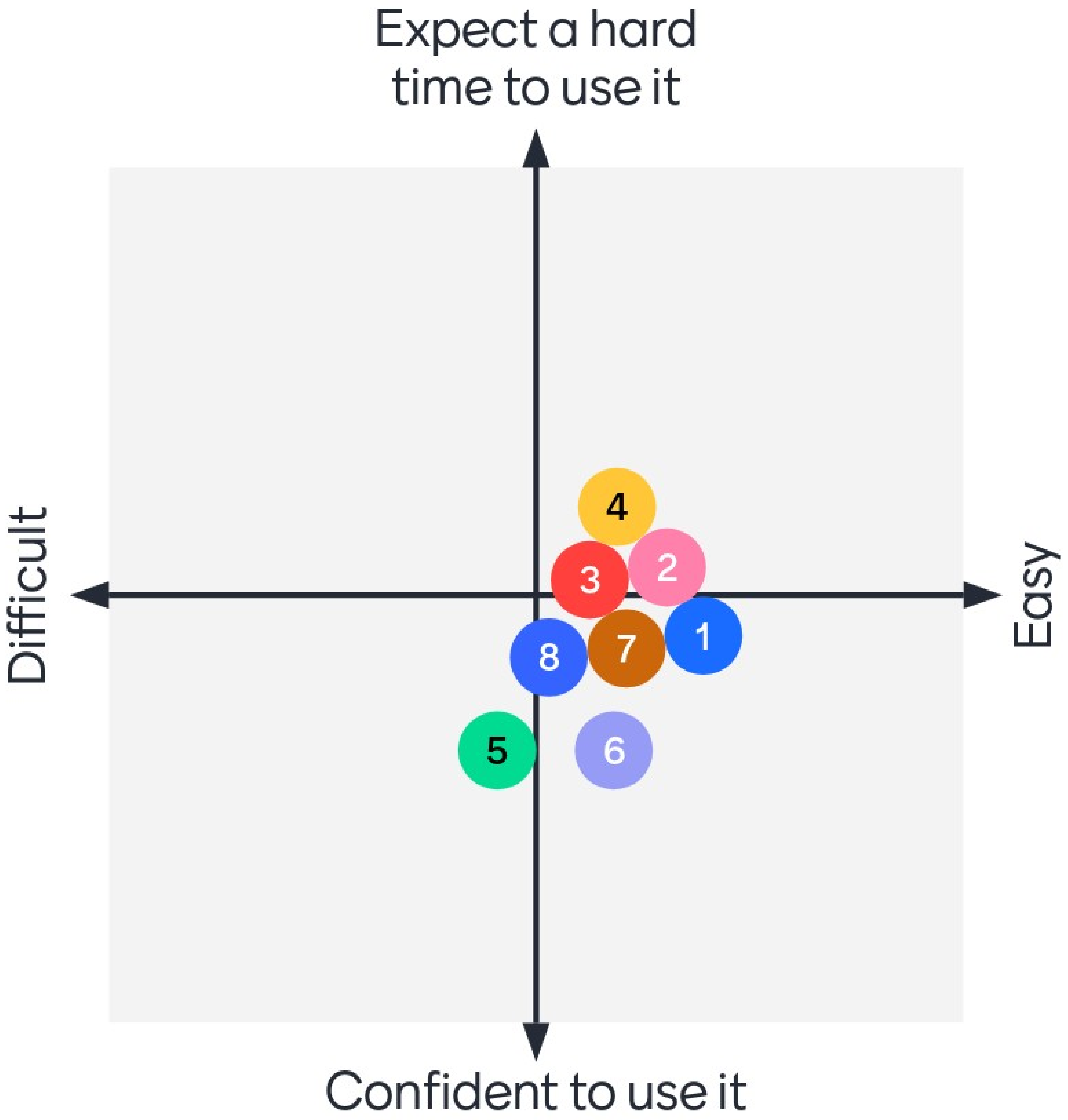


```
// for each element in *vec,
// we get the smallest not-yet-assigned from left and from right,
// then assign the smaller one to it
for (int v_idx = 0, l_idx = 0, r_idx = 0; v_idx < vec->size(); ++v_idx) {
  // if both left and right have elements that are not-yet-assigned
  if (l_idx < left.size() && r_idx < right.size()) {
    // check which one is smaller and assign it to *vec
    if (left.at(l_idx) < right.at(r_idx))
      vec->at(v_idx) = left.at(l_idx++);
    else
      vec->at(v_idx) = right.at(r_idx++);
  }
  // otherwise, just assign the rest to *vec
  else if (l_idx < left.size()) // only left has not-assigned elements
    vec->at(v_idx) = left.at(l_idx++);
  else // only right has not-assigned elements
    vec->at(v_idx) = right.at(r_idx++);
}
```

Ex 8-1: Merge sort - merge

Tell me how you feel about the new topics:

Mentimeter



1 std::map

2 std::pair

3 std::tuple

4 iterators

5 std::sort

6 std::find

7 std::count

8 std::is_permutation

6

Which of the following about **std::map** is true?

1. It is templated.
2. Given std::map<int, char> m; Each key can have multiple char values.
3. We can use iterator to visit all elements of a map.
4. We can use indices to visit all elements of a map.
5. Each element in std::map is in fact a std::pair.

Which of the following about **std::pair**, **std::tuple** and **iterators** is true?

1. They provide a mean to return multiple values in C++.
2. `std::pair` must be a pair of the same data type.
3. We can use `std::get<N>` to get the Nth field of `std::pair` variables.
4. If we don't need to modify the content, `const_iterator` is preferred.
5. If `it` is an iterator, then `--it` will iterate one element backward.

Check online and find the answers for the below questions.

1. Is it true that std::sort can sort any data type?
2. What does std::copy do? When and how should you use it?
3. Which function in STL algorithms can you use to find both maximum and minimum values in a STL data container?
4. What does std::move do? What happens to the elements in the original container after moving?
5. What is the difference between std::swap and std::iter_swap? If you need to swap, which one do you prefer?

♥ ? 👍 👎

Ask me anything

0 questions

0 upvotes