1. What is the time complexity of finding the maximum element in a sliding window of size 'k' using a deque?
   1. O(k)
   2. O(log k)
   3. O(n)
   4. **O(1)**
2. In a sliding window maximum problem, when do you remove an element from the front of the deque?
   1. When it becomes the minimum element
   2. **When its index is outside the current window**
   3. When its index is inside the current window
   4. When it is the maximum element
3. Which of the following problems can be efficiently solved using the sliding window technique?
   1. Finding all prime numbers in an array
   2. Finding the longest palindromic substring in a string
   3. Finding the shortest path in a graph
   4. **Finding the maximum sum subarray of size 'k'**
4. How does the sliding window technique work?
   1. **It slides a window of fixed size over the data**
   2. It divides the data into multiple windows of equal size
   3. It sorts the data using a sliding window
   4. It finds the maximum element in a sorted window
5. What is the key feature of deque in C++?
   1. **It supports constant time insertion at both ends**
   2. It supports constant time insertion at the front and linear time insertion at the back
   3. It supports constant time insertion at the back and linear time insertion at the front
   4. It supports constant time insertion at the front and logarithmic time insertion at the back
6. Which header file should be included to use the deque container in C++?
   1. <vector>
   2. <list>
   3. **<deque>**
   4. <algorithm>
7. How do you add an element to the back of a deque in C++?
   1. **push\_back()**
   2. push\_front()
   3. insert()
   4. add\_back()
8. Which of the following operations is not supported by a deque in C++?
   1. Adding elements to both ends
   2. Removing elements from both ends
   3. **Random access by index**
   4. Iterating in reverse order
9. What is the time complexity of adding an element to the back of a deque?
   1. **O(1)**
   2. O(log n)
   3. O(n)
   4. O(n log n)
10. How do you remove an element from the front of a deque in C++?
    1. pop\_back()
    2. **pop\_front()**
    3. erase()
    4. remove\_front()
11. The sliding window technique is used for which type of problems?
    1. Sorting problems
    2. Dynamic programming problems
    3. String matching problems
    4. **Subarray or substring problems**
12. Which algorithm is commonly used to efficiently solve the maximum sliding window problem?
    1. Dijkstra's algorithm
    2. Kruskal's algorithm
    3. Prim's algorithm
    4. **Monotonic queue algorithm**
13. In a sliding window maximum problem, why do we use a monotonic queue?
    1. To keep the elements in the window sorted
    2. **To keep track of the maximum element in the window**
    3. To find the median of elements in the window
    4. To ensure all elements in the window are unique
14. Which of the following problems can be efficiently solved using the deque in C++?
    1. Finding the shortest path in a graph
    2. Implementing a queue with constant time access to both ends
    3. **Implementing a stack with constant time access to both ends**
    4. Finding the longest increasing subsequence in an array
15. What is the time complexity of finding the minimum element in a sliding window of size 'k' using a deque?
    1. O(k)
    2. O(log k)
    3. O(n)
    4. **O(1)**
16. Which of the following operations is supported by a deque in C++?
    1. pop()
    2. **push()**
    3. enqueue()
    4. resize()
17. Which of the following problems can be efficiently solved using the deque in C++?
    1. Finding the longest common subsequence of two strings
    2. **Finding the maximum element in each subarray of size 'k'**
    3. Finding the shortest path between two nodes in a graph
    4. Sorting an array of integers
18. What is the key advantage of using a deque over a vector or list in certain scenarios?
    1. Deque provides faster random access by index
    2. **Deque provides faster insertion and deletion at both ends**
    3. Deque requires less memory compared to vector or list
    4. Deque allows duplicate elements in the container
19. In the sliding window technique, when do we slide the window to the right?
    1. When the window is empty
    2. When the window contains the maximum element
    3. When the window's size becomes smaller than the required size
    4. **When the window's size becomes equal to the required size**
20. How does the sliding window technique efficiently solve problems?
    1. It sorts the data first and then applies a sliding window
    2. It stores all possible windows and then selects the correct one
    3. **It maintains a data structure to track relevant information about the window**
    4. It uses nested loops to check each possible window
21. How can you access the first element of a deque?
    1. **deque.front()**
    2. deque.begin()
    3. deque[0]
    4. deque.at(0)
22. What is the main advantage of using a deque over a vector in C++ STL?
    1. Deques have faster element access time
    2. Deques have a smaller memory footprint
    3. **Deques allow constant time insertion at both ends**
    4. Deques can store elements of different data types
23. Which operation can be performed on both ends of a deque in constant time?
    1. Insertion
    2. Deletion
    3. **Access**
    4. Search
24. What is the underlying data structure used to implement a deque?
    1. **Array**
    2. Linked list
    3. Vector
    4. Heap
25. How is the memory allocated for a deque when it grows beyond its capacity?
    1. **The operating system allocates a new block of memory**
    2. Elements are shifted to the new memory location
    3. The deque size remains fixed, and new elements are discarded
    4. The deque uses a linked list to dynamically allocate memory
26. Which method is used to remove the last element from a deque?
    1. deque.remove()
    2. **deque.pop\_back()**
    3. deque.delete\_last()
    4. deque.erase()
27. There are the following statements that are given below, which of them are correct about deque container in C++?
28. The deque stands for the double-ended queue.
29. The deque is a sequence container.
30. The deque is a derived container.
31. In the deque container, data can be inserted and deleted from both the front and back ends the side.
    1. A and B
    2. A, C, and D
    3. **A, B, and D**
    4. C and D
32. What is the correct output of the given code snippets?

#include <iostream>

#include <deque>

using namespace std;

int main()

{

deque<int> d;

d.add(10);

d.add(20);

d.add(30);

for (int i = 0; i < d.size(); i++) {

cout << d[i] << " ";

}

return 0;

}

* 1. 10 20 30
  2. Garbage Value
  3. **Syntax error**
  4. Runtime error

1. What is the correct output of the given code snippets?

#include <iostream>

using namespace std;

int main()

{

deque<int> d;

d.push\_back(10);

d.push\_back(20);

d.push\_back(30);

for (int i = 0; i < d.size(); i++) {

cout << d[i] << " ";

}

return 0;

}

1. 10 20 30
2. Garbage Value
3. **Syntax error**
4. Runtime error
5. What is the correct output of the given code snippets?

#include <iostream>

#include <deque>

using namespace std;

int main()

{

deque<int> d;

d.push\_back(10);

d.push\_front(20);

d.push\_back(30);

for (int i = 0; i < d.size(); i++) {

cout << d[i] << " ";

}

return 0;

}

1. 10 20 30
2. **20 10 30**
3. 30 10 20
4. Syntax error
5. What is the correct output of the given code snippets?

#include <iostream>

#include <deque>

using namespace std;

int main()

{

deque<int> d;

d.push\_back(10);

d.push\_front(20);

d.pop\_back();

d.push\_back(40);

for (int i = 0; i < d.size(); i++) {

cout << d[i] << " ";

}

return 0;

}

* 1. 10 20 40
  2. 10 40
  3. **20 40**
  4. Syntax error