

### 1.Kaden's algo

```
practice.cpp > main()
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  int main(){
5      int n;
6      cin >> n;
7      int arr[n];
8      for(int i=0; i<n; i++){
9          cin >> arr[i];
10     }
11
12     int max_sum = INT_MIN;
13     int sum = 0;
14     for(int i=0; i<n; i++){
15         sum += arr[i];
16         max_sum = max(max_sum, sum);
17         if(sum < 0){
18             sum = 0;
19         }
20     }
21     cout << max_sum << endl;
22     return 0;
23 }
```

$T_c = o(n)$   $sc = o(1)$

OUTPUT:

```
...  input.txt  x
    input.txt
    1  9
    2  -2 1 -3 4 -1 2 1 -5 4

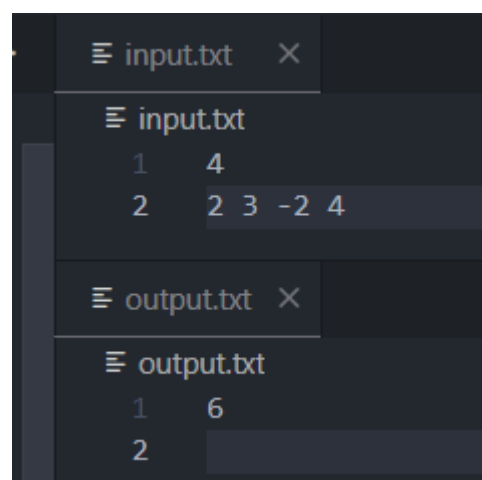
    output.txt  x
    output.txt
    1  6
    2
```

## 2.maximum product subarray

```
practice.cpp > ...
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  int main()
5  {
6      int n;
7      cin >> n;
8      int arr[n];
9      for (int i = 0; i < n; i++) {
10         cin >> arr[i];
11     }
12
13     int maxtill = INT_MIN;
14     int pre = 1, suf = 1;
15
16     for (int i = 0; i < n; i++) {
17         pre = (pre == 0) ? arr[i] : pre * arr[i];
18         suf = (suf == 0) ? arr[n - i - 1] : suf * arr[n - i - 1];
19         maxtill = max(maxtill, max(pre, suf));
20     }
21
22     cout << maxtill << endl;
23     return 0;
24 }
```

$T_c = O(n)$   $sc = O(1)$

Output:



```
input.txt  X
1  4
2  2 3 -2 4

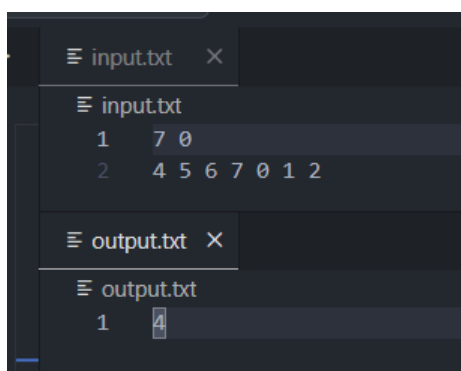
output.txt  X
1  6
2
```

### 3. search in rotated sorted array

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    int n, target;
    cin >> n >> target;
    vector<int> nums(n);
    for (int i = 0; i < n; i++) {
        cin >> nums[i];
    }
    int low = 0;
    int high = nums.size() - 1;
    while (low <= high) {
        int mid = (low + high) / 2;
        if (nums[mid] == target) {
            cout << mid;
            return 0;
        } else if (nums[low] <= nums[mid]) {
            if (nums[low] <= target && nums[mid] >= target)
                high = mid - 1;
            else
                low = mid + 1;
        } else {
            if (nums[mid] <= target && nums[high] >= target)
                low = mid + 1;
            else
                high = mid - 1;
        }
    }
    cout << -1;

    return 0;
}
```

$T_c = O(\log n)$     $sc = O(1)$



#### 4.Container with most water

```
practice.cpp > main()
1  #include <bits/stdc++.h>
2  using namespace std;
3  int main() {
4      int n, target;
5      cin >> n >> target;
6      vector<int> h(n);
7      for (int i = 0; i < n; i++) {
8          cin >> h[i];
9      }
10     int maxArea = 0;
11     int l = 0, r = h.size() - 1;
12     while (l < r) {
13         maxArea = max(maxArea, min(h[l], h[r]) * (r - l));
14         if (h[l] < h[r])
15             l++;
16         else
17             r--;
18     }
19     cout << maxArea;
20
21     return 0;
22 }
23
```

Tc =  $O(n)$  sc =  $O(1)$

Output:

```
..  input.txt  X
    input.txt
    1  9
    2  1 8 6 2 5 4 8 3 7
    output.txt X
    output.txt
    1  49
```