Введение в теорию сложности

10 класс

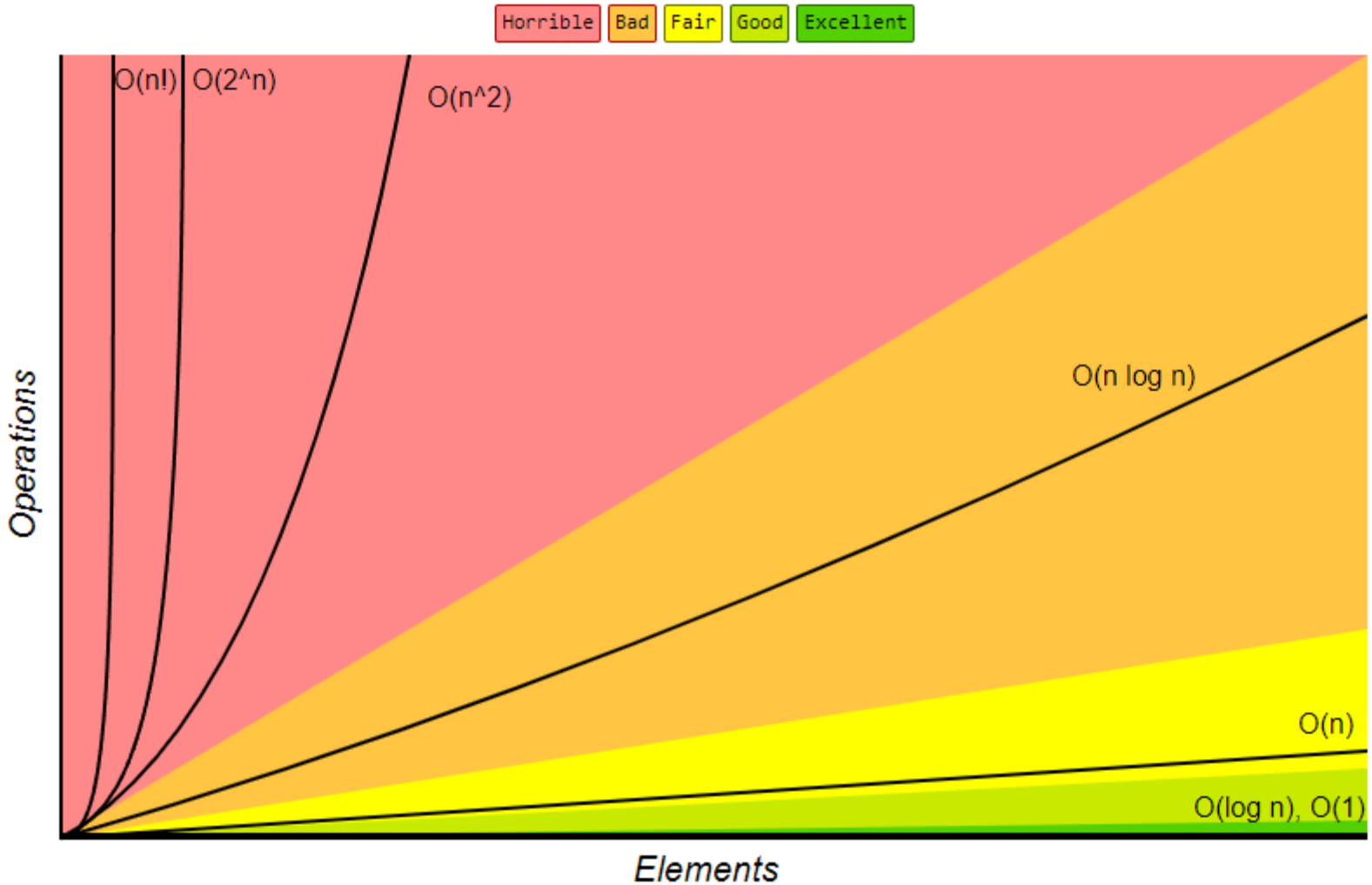
Вспомним основные теоретические аспекты

Почему нет смысла сравнивать программы по времени их работы?

Какие виды асимптотической сложности алгоритмов существуют?

Зачем нам знать асимптотическую сложность алгоритма?

Big-O Complexity Chart



Common Data Structure Operations

Data Structure	Time Complexity								Space Complexity
	Average				Worst				Worst
	Access	Search	Insertion	Deletion	Access	Search	Insertion	Deletion	
Array	Θ(1)	Θ(n)	Θ(n)	Θ(n)	O(1)	O(n)	O(n)	O(n)	O(n)
Stack	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Queue	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Singly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Doubly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Skip List	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n log(n))
Hash Table	N/A	Θ(1)	Θ(1)	Θ(1)	N/A	O(n)	O(n)	O(n)	O(n)
Binary Search Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)
Cartesian Tree	N/A	Θ(log(n))	Θ(log(n))	Θ(log(n))	N/A	O(n)	O(n)	O(n)	O(n)
B-Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Red-Black Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Splay Tree	N/A	Θ(log(n))	Θ(log(n))	Θ(log(n))	N/A	O(log(n))	O(log(n))	O(log(n))	O(n)
AVL Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
KD Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)

Array Sorting Algorithms

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
Quicksort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(log(n))
Mergesort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(n)
Timsort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)
Heapsort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(1)
Bubble Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Insertion Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Selection Sort	Ω(n^2)	Θ(n^2)	O(n^2)	O(1)
Tree Sort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(n)
Shell Sort	Ω(n log(n))	Θ(n(log(n))^2)	O(n(log(n))^2)	O(1)
Bucket Sort	Ω(n+k)	Θ(n+k)	O(n^2)	O(n)
Radix Sort	Ω(nk)	Θ(nk)	O(nk)	O(n+k)
Counting Sort	Ω(n+k)	Θ(n+k)	O(n+k)	O(k)
Cubesort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)

Задание 1. Оцените сложность алгоритмов в нотации Big O:

Выражение	**O(…)**
$5 + 0.001n^3 + 0.025n$	
$500n + 100n^{1.5} + 50n \log_{10} n$	
$0.3n + 5n^{1.5} + 2.5n^{1.75}$	
$100n + 0.01n^2$	
$0.01n + 100n^2$	
$2n + n^{0.5} + 0.5n^{1.25}$	
$100n \log_3 n + n^3 + 100n$	

Задание 2. Оцените сложность алгоритмов в нотации Big O:

$f(n)$	$O(\ldots)$	**Пример**
$3n + 4$		
$2n$		
$n^2 + 100n$		
$n + n^{0.5}$		

Задание 3 (Предпроф. экзамен). Зависимость времени выполнения алгоритма от количества операций для каждого из алгоритмов X и Y записана выражениями: $X(n) = n^2 + 12n - 10$, $Y(n) = n^2 + 11n + 10$, где n – количество операций. Чему должно быть равно n, чтобы время выполнения алгоритмов стало одинаково?

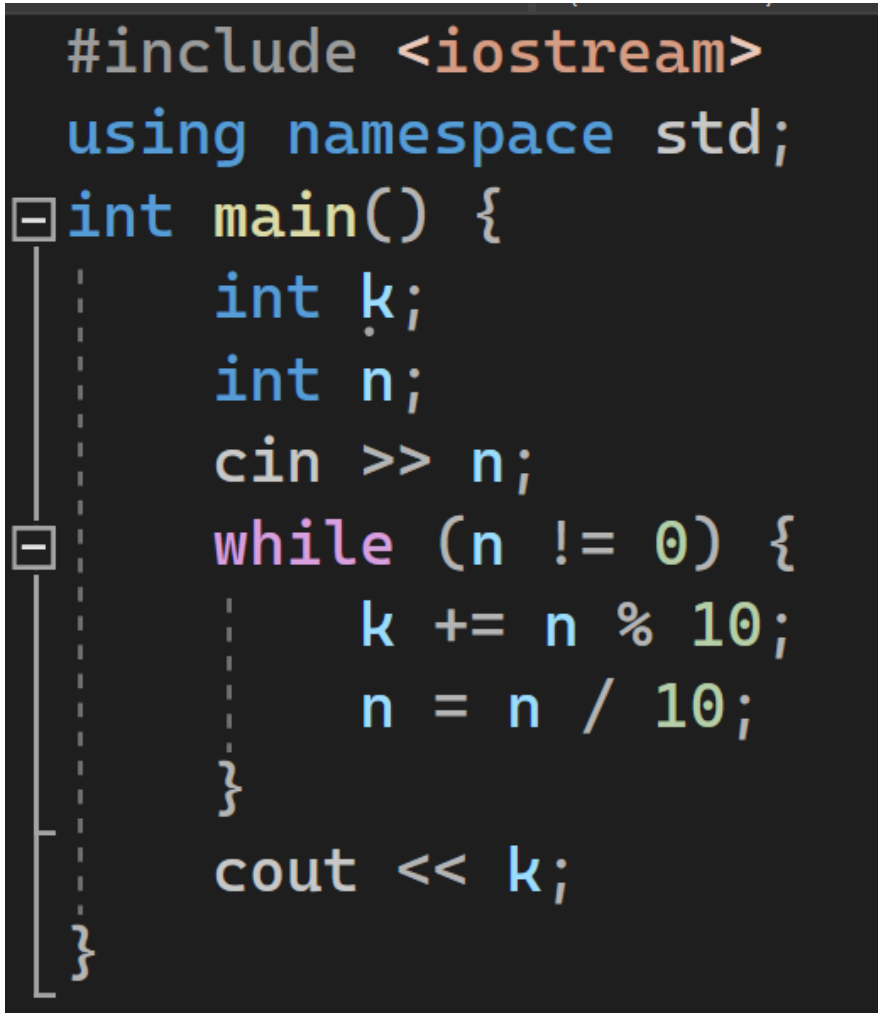
Задание 4. Оцените асимптотическую сложность алгоритма в нотации Big O.

```cpp
#include <iostream>
using namespace std;
int main() {
    int k = 0;
    int n;
    cin >> n;
    for (int i = 5; i < n; i++) {
        k++;
    }
}
```

Задание 5. Оцените асимптотическую сложность алгоритма в нотации Big O. Опишите цель данного алгоритма.

```cpp
#include <iostream>
using namespace std;
int main() {
    int k;
    long m = 1e10;
    int n;
    cin >> n;
    for (int i = 0; i < n; i++) {
        cin >> k;
        if (k < m) {
            m = k;
        }
    }
    cout << m;
}
```

Задание 6. Оцените асимптотическую сложность алгоритма в нотации Big O. Опишите цель данного алгоритма.



```cpp
#include <iostream>
using namespace std;
int main() {
    int k;
    int n;
    cin >> n;
    while (n != 0) {
        k += n % 10;
        n = n / 10;
    }
    cout << k;
}
```

Задание 7. Оцените асимптотическую сложность алгоритма в нотации Big O. Опишите цель данного алгоритма.

```
#include <iostream>
using namespace std;
int main() {
    int n = 0;
    cin >> n;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < i * i; j++) {
            cout << i * j;
        }
    }
}
```

Задание 8. Оцените асимптотическую сложность алгоритма в нотации Big O. Опишите цель данного алгоритма.

```cpp
#include <iostream>
using namespace std;
int main() {
    int n = 0;
    cin >> n;
    for (int i = 1; i < n; i++) {
        for (int j = i; j < n; j++) {
            cout << i * j;
        }
    }
}
```

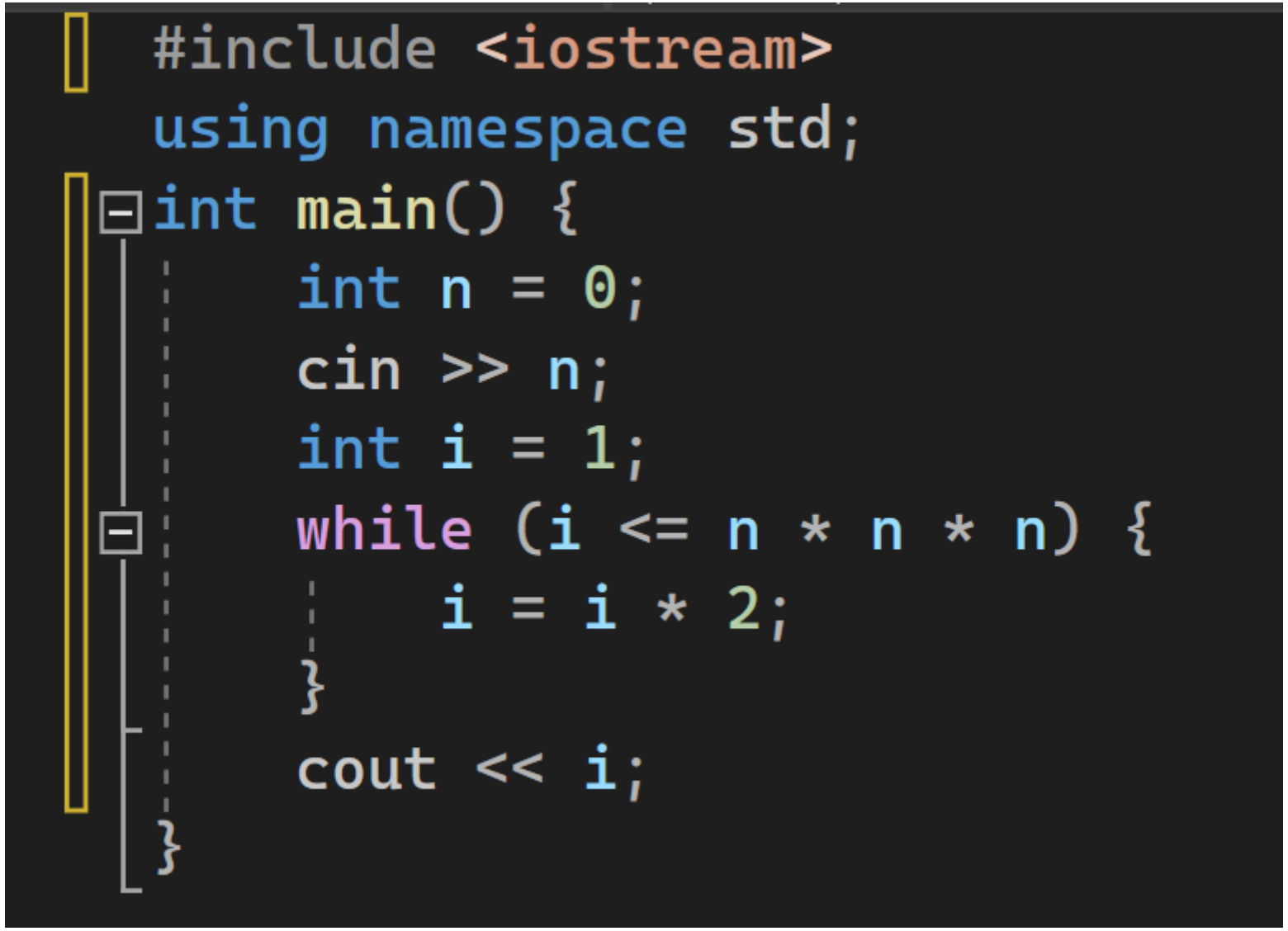
Задание 9. Оцените асимптотическую сложность алгоритма в нотации Big O. Опишите цель данного алгоритма.

```cpp
#include <iostream>
using namespace std;
int main() {
    int n = 0;
    cin >> n;
    for (int i = 0; i < n; i++) {
        cout << i << endl;
    }
    int k = 0;
    cin >> k;
    for (int i = 0; i * i < k; i++) {
        cout << i * i << endl;
    }
}
```

Задание 10. Оцените асимптотическую сложность алгоритма в нотации Big O. Опишите цель данного алгоритма.

```cpp
#include <iostream>
using namespace std;
int main() {
    int n = 0;
    cin >> n;
    int i = 1;
    while (i <= n * n * n) {
        i = i * i;
    }
    cout << i;
}
```

Задание 11. Оцените асимптотическую сложность алгоритма в нотации Big O. Опишите цель данного алгоритма.



```cpp
#include <iostream>
using namespace std;
int main() {
    int n = 0;
    cin >> n;
    int i = 1;
    while (i <= n * n * n) {
        i = i * 2;
    }
    cout << i;
}
```