Data Structures & Algorithms

Lecture: Big O Notation

NUST
SCHOOL OF ELECTRICAL ENGINEERING &
COMPUTER SCIENCE (SEECS)

what is it?

simplified analysis of an algorithm's efficiency

1. complexity in terms of input size, N
2. machine–independent
3. basic computer steps

Big O Notation (cont’d)

types of measurement

worst–case

best–case

average–case

Big O Notation (cont'd)

general rules

1. ignore constants

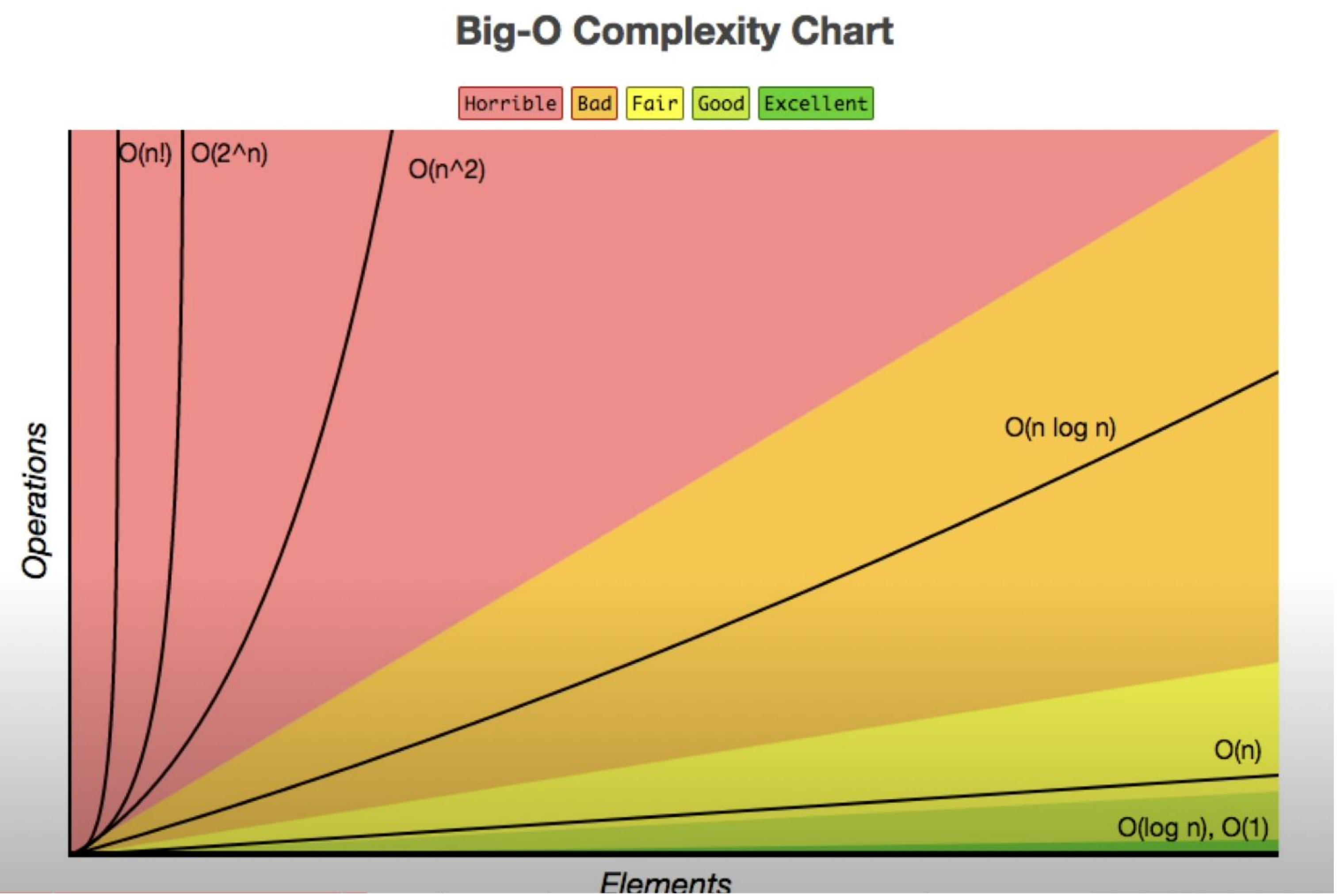
   $5n \rightarrow O(n)$

2. certain terms "dominate" others

   $O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2) < O(2^n) < O(n!)$

   i.e., ignore low-order terms

Big O Notation (cont'd)



Big O Notation (cont'd)

constant time

```
x = 5 + (15 * 20);
```

independent of input size, N

Big O Notation (cont'd)

NUST SEECS

constant time

O(1) "big oh of one"

```
x = 5 + (15 * 20);
```

independent of input size, N

Big O Notation (cont'd)

```
x = 5 + (15 * 20);
y = 15 - 2;
print x + y;
```

$$\text{total time} = O(1) + O(1) + O(1) = O(1)$$

$$3 * O(1)$$

linear time

$$N * O(1) = O(N)$$

```
for x in range (0, n):
  print x; // O(1)
```

Big O Notation (cont'd)

```
y = 5 + (15 * 20);
for x in range (0, n):
    print x;
```

$O(1)$

$O(N)$

total time = $O(1) + O(N) = O(N)$

quadratic time

$O(N^2)$

```
for x in range (0, n):
    for y in range (0, n):
        print x * y; // O(1)
```

Big O Notation (cont'd)

O(?)

```
x = 5 + (15 * 20);
for x in range (0, n):
    print x;
for x in range (0, n):
    for y in range (0, n):
        print x * y;
```

Big O Notation (cont'd)

O(?)

```
x = 5 + (15 * 20);
```

$O(1)$

```
for x in range (0, n):
    print x;
```

$O(N)$

```
for x in range (0, n):
    for y in range (0, n):
        print x * y;
```

$O(N^2)$

Big O Notation (cont'd)

NUST SEECS

$O(N^2)$

```
x = 5 + (15 * 20);                      O(1)
for x in range (0, n):
     print x;                           O(N)
for x in range (0, n):
     for y in range (0, n):             O(N²)
          print x * y;
```

NUST SEECS

O(?)

```
if x > 0:
    // O(1)
else if x < 0:
    // O(logn)
else:
    // O(n²)
```

Big O Notation (cont'd)

$O(N^2)$

```
if x > 0:
    // O(1)
else if x < 0:
    // O(logn)
else:
    // O(n²)
```

Big O Notation (cont'd)

NUST SEECS

in practice

1. constants matter
2. be cognizant of best-case and average-case

Big O Notation (cont'd)

- O(n/2)
- O(n*m)
- O(n^3)
- O(3^n)