|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Big O** | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | o(1) | constant |  |  |  |  |
|  |  |  |  | o(log n) | logarithmic |  |  |  |  |
|  |  |  |  | o(n) | linear |  |  |  |  |
|  |  |  |  | o(n log n) | linearithmic |  |  |  |  |
|  |  |  |  | o(n^2) | quadratic |  |  |  |  |
|  |  |  |  | o(2^n) | exponential |  |  |  |  |
|  |  |  |  | o(n!) | factorial |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| **Common Data Structures** | **Time Complexity** | | | | | | | | **Space Complexity** |
|  | Average |  |  |  | Worst |  |  |  |  |
| *Data Structure* | *Access* | *Search* | *Insertion* | *Deletion* | *Access* | *Search* | *Insertion* | *Deletion* | *Overall* |
| Array | o(1) | o(n) | o(n) | o(n) | o(1) | o(n) | o(n) | o(n) | o(n) |
| Stack | o(n) | o(n) | o(1) | o(1) | o(n) | o(n) | o(1) | o(1) | o(n) |
| Queue | o(n) | o(n) | o(1) | o(1) | o(n) | o(n) | o(1) | o(1) | o(n) |
| Singly-Linked List | o(n) | o(n) | o(1) | o(1) | o(n) | o(n) | o(1) | o(1) | o(n) |
| Doubly-Linked List | o(n) | o(n) | o(1) | o(1) | o(n) | o(n) | o(1) | o(1) | o(n) |
| Skip List | o(log n) | o(log n) | o(log n) | o(log n) | o(n) | o(n) | o(n) | o(n) | o(n log n) |
| Hash Table | N/A | o(1) | o(1) | o(1) | N/A | o(n) | o(n) | o(n) | o(n) |
| Binary Search Tree | o(log n) | o(log n) | o(log n) | o(log n) | o(n) | o(n) | o(n) | o(n) | o(n) |
| Cartesian Tree | N/A | o(log n) | o(log n) | o(log n) | N/A | o(n) | o(n) | o(n) | o(n) |
| B-Tree | o(log n) | o(log n) | o(log n) | o(log n) | o(log n) | o(log n) | o(log n) | o(log n) | o(n) |
| Red-Black Tree | o(log n) | o(log n) | o(log n) | o(log n) | o(log n) | o(log n) | o(log n) | o(log n) | o(n) |
| Splay Tree | N/A | o(log n) | o(log n) | o(log n) | N/A | o(log n) | o(log n) | o(log n) | o(n) |
| AVL Tree | o(log n) | o(log n) | o(log n) | o(log n) | o(log n) | o(log n) | o(log n) | o(log n) | o(n) |
| KD Tree | o(log n) | o(log n) | o(log n) | o(log n) | o(n) | o(n) | o(n) | o(n) | o(n) |



# Coding Techniques

## How many digits a number has?

1. Use Math.Log (number):

int count = (int) Math.Log10(nums[i]) + 1;

1. Create variable with value 10 (decimal) and mod the number with the value:

public int NumberOfDigits (int val)

{

int count = 1;

int mul = 10;

while(val % mul != val)

{

count++;

mul = mul \* 10;

}

return count;

}

## Reverse an Integer number?

public static int Reverse(int x)

{

int RevNum = 0;

bool PosInt = x >= 0 ? true : false;

x = Math.Abs(x);

while (x > 0)

{

RevNum = (RevNum \* 10) + (x % 10);

x = x / 10;

}

return PosInt ? RevNum : -RevNum;

}

## Convert a number from binary to integer?

1. Use Convert.ToInt32(binary\_number,2)

int num = Convert.ToInt32(BinaryValue, 2);

1. Binary Shift:

int num = 0;

num <<= 1;

num |= BinaryValue;

OR

int num = 0;

num = (num << 1) | BinaryValue;

## Split a number into individual digits (example 123 into 1 and 2 and 3):

int[] GetIntArray(int num)

{

List<int> listOfInts = new List<int>();

while(num > 0)

{

listOfInts.Add(num % 10);

num = num / 10;

}

listOfInts.Reverse();

return listOfInts.ToArray();

}

## Transform string to lower case

public static string ToLower(string str)

{

StringBuilder lowerCase = new StringBuilder();

for (int i = 0; i < str.Length; i++)

{

if (str[i] >= 65 && str[i] <= 90)

{

lowerCase.Append((char)(str[i] + 32));

}

}

return lowerCase.ToString();

}