# ADSA exercises

# Chapter 1 - Introduction, Analysis

Find the runtime (algorithmic complexity) of the following pieces of code.

## EASY

1.

```
void foo(int[] array){
  int sum = 0;
  int product = 1;
  for (int i = 0; i < array.length; i++){
    sum += array[i];
  }
  for (int i = 0; i < array.length; i++){
    product *= array[i];
  }
  System.out.println(sum + ", " + product);
}
```

2.

```
void printPairs(int[] array){
  for (int i = 0; i < array.length; i++){
    for (int j = 0; j < array.length; j++){
      System.out.println(array[i] + ", " + array[j]);
    }
  }
}
```

3.

```
void printUnorderedPairs(int[] array){
  for (int i = 0; i < array.length; i++){
    for (int j = i + 1; j < array.length; j++){
      System.out.println(array[i] + ", " + array[j]);
    }
  }
}
```

4.

```
void printUnorderedPairs(int[] arrayA, int[] arrayB){
  for (int i = 0; i < arrayA.length; i++){
    for (int j = 0; j < arrayB.length; j++){
      if (arrayA[i] < arrayB[j]){
        System.out.println(arrayA[i] + ", " + arrayB);
      }
    }
  }
}
```

5.

```
void printUnorderedPairs(int[] arrayA, int[] arrayB){
  for (int i =0; i < arrayA.length; i++){
    for (int j = 0; j < arrayB.length; j++){
      for (int k = 0; k < 100000; k++){
        System.out.printlen(arrayA[i] + "," + arrayB[j]);
      }
    }
  }
}
```

6.

```
void reverse(int[] array){
  for (int i = 0; i < array.length / 2; i++){
    int other = array.length - i - 1;
    int temp = array[i];
    array[i] = array[other];
    array[other] = temp;
  }
}
```

7.

```
boolean isPrime(int n){
  for (int x = 2; x * x <= n; x++){
    if (n % x == 0){
      return false;
    }
  }
  return true;
}
```

8.

```
int factorial(int n){
  if (n < 0){
    return -1;
  } else if (n == 0){
    return 1;
  } else {
    return n * factorial(n - 1);
  }
}
```

9.

```
int product(int a, int b){
  int sum = 0;
  for (int i = 0; i < b; i++){
    sum += a;
  }
  return sum;
}
```

10.

```
int power(int a, int b){
  if (b < 0){
    return 0; // error
  } else if (b == 0){
    return 1;
  } else {
    return a * power(a, b-1);
  }
}
```

11.
The follow ing code computes a % b.

```
int mod(int a, int b){
  if (b <= 0){
    return -1;
  }
  int div = a / b;
  return a - div * b;
}
```

## MEDIUM

1.

```
int powersOf2(int n){
  if (n == 1){
    System.out.println(1);
    return 1;
  } else {
    int prev = powersOf2(n / 2);
    int curr = prev * 2;
    System.out.println(curr);
    return curr;
  }
}
```

2.

```
int sqrt(int n){
  return sqrt_helper(n, 1, n);
}

int sqrt_helper(int n, int min, int max){
  if (max < min) return -1; // no square root
  int guess = (min + max) / 2;
  if (guess * guess == n){ // found it!
    return guess;
  } else if (guess * guess < n ){ // too low
    return sqrt_helper(n, guess + 1, max); // try higher
  } else { // too high
    return sqrt_helper(n, 1, guess - 1); // try lower
  }
}
```

3.

```
int sqrt(int n){
  for (int guess = 1; guess * guess <= n; guess++){
    if (guess * guess == n){
      return guess;
      }
    }
    return -1;
  }
```

4.

```
int sumDigits(int n){
  int sum = 0;
  while (n > 0){
    sum += n % 10;
    n /= 10;
  }
  return sum;
}
```

## HARD

1.

```
void permutation(String str){
  permutation(str, "");
}

void permutation(String str, String prefix){
  if (str.length() == 0){
    System.out.println(prefix);
  } else {
    for (int = 0; i < str.length(); i++){
      String rem = str.substring(0, i) + str.subString(i + 1);
      permutation(rem, prefix + str.charAt(i));
    }
  }
}
```

2.

```
int fib(int n){
  if (n <= 0) return 0;
  else if (n == 1) return 1;
  return fib(n - 1) + fib(n - 2);
}
```

3.

```
int div(int a, int b){
  int count = 0;
  int sum = b;
  while (sum <= a){
    sum += b;
    count++;
  }
  return count;
}
```

4. The follow ing code computes the intersection (the number of elements in common) of tw o arrays. It assumes that neither are duplicates. It computes the intersection by sorting one array b and then iterating through array a checking (via binary

search) if each values is in b. What is its runtime?

```
int intersection(int[] a, int[] b){
  mergesort(b);
  int intersect = 0;
  for (int x : a){
    if (binarySearch(b, x) >= 0){
      intersect ++;
    }
  }
  return intersect;
}
```