Złożoność

Przeanalizuj poniższe funkcje i oszacuj pesymistyczną złożoność każdej z nich względem przyjmowanych parametrów. Przedstaw złożoność w notacji wielkiego O.

Zadanie 1

```
int silnia(int n) {
    int wynik = 1;

    for(int i = 1; i <= n; i++) {
            wynik *= i;
    }

    return wynik;
}
```

Zadanie 2

```
bool czyPierwsza(int n) {
    for(int i = 2; i <= sqrt(n); i++) {
        if (n % i == 0) {
            return false;
        }
    }

    return true;
}
```

Zadanie 3

```
int potega(int a, int b) {
    int wynik = 1;
    for(int i = 1; i <= b; i++) {
        wynik *= a;
    }

    return wynik;
}
```

Zadanie 4

```
string naBinarny(int n) {
    int reszta;
    string wynik;
    while(n > 0) {
        reszta = n % 2;
        n /= 2;
        wynik = (char)(reszta + '0') + wynik;
    }

    return wynik;
}
```

Zadanie 5

```
int naDziesietny(string bin) {
    int wynik = 0, cyfra, potega = 1;
    for(int i = bin.length() - 1; i >= 0; i--) {
        cyfra = bin[i] - '0';
        wynik += cyfra * potega;
        potega *= 2;
    }

    return wynik;
}
```

Zadanie 6

```
int szukaj(int tab[], int n, int szukana) {
    for(int i = 0; i < n; i++) {
        if (szukana == tab[i]) {
            return i;
        }
    }

    return -1;
}
```

Zadanie 7

```
double suma(double tab[][4], int wys, int szer) {
    double wynik = 0;
    for(int i = 0; i < wys; i++) {
        for(int j = 0; j < szer; j++) {
            wynik += tab[i][j];
        }
    }

    return wynik;
}
```

Zadanie 8

```
void zliczLitery(string wyraz, int liczniki[]) {
    int numerLitery;
    for(int i = 0; i < wyraz.length(); i++) {
        numerLitery = (int)(wyraz[i] - 'a');
        liczniki[numerLitery]++;
    }
}
```

Zadanie 9

```
bool czyAnagramy(string wyraz1, string wyraz2) {
    int liczniki1[26] = {};
    int liczniki2[26] = {};
    zliczLitery(wyraz1, liczniki1);
    zliczLitery(wyraz2, liczniki2);

    for(int i = 0; i < 26; i++) {
        if(liczniki1[i] != liczniki2[i]) {
            return false;
        }
    }

    return true;
}
```

Zadanie 10

```
void insertionSort(int array[], int n) {
    for (int i = 1; i < n; i++) {
        int j = i;
        while (j > 0 && array[j] < array[j - 1]) {
            swap(array[j], array[j - 1]);
            j--;
        }
    }
}
```

Zadanie 11

```
void oddEvenSort(int array[], int n) {
    for (int i = 0; i < n; i++) {
        for (int j = i % 2 + 1; j < n; j += 2) {
            if (array[j] < array[j - 1]) {
                swap(array[j], array[j - 1]);
            }
        }
    }
}
```

Zadanie 12

```
int fastExp(int a, int n) {
    int w = 1;

    while (n > 0) {
        if (n % 2 == 1) {
            w *= a;
        }

        a *= a;
        n /= 2;
    }

    return w;
}
```

Zadanie 13

```
int levenshteinDistance(string a, string b) {
    int matrix[a.length() + 1][b.length() + 1];
    int cost;

    for (int i = 0; i < a.length(); i++) {
        matrix[i][0] = i;
    }

    for (int i = 0; i < b.length(); i++) {
        matrix[0][i] = i;
    }

    for (int i = 1; i <= a.length(); i++) {
        for (int j = 1; j <= b.length(); j++) {
            if (a[i - 1] == b[j - 1]) {
                cost = 0;
            } else {
                cost = 1;
            }

            matrix[i][j] = min(matrix[i - 1][j - 1] + cost, min(matrix[i - 1][j]
+ 1, matrix[i][j - 1] + 1));
        }
    }

    return matrix[a.length()][b.length()];
}
```