Introduction

О проекте

Данный проект представляет собой учебный набор библиотек для изучения алгоритмов и структур данных. Основная цель проекта — академическая, собрать в себе как можно больше алгоритмов и структур данных для исследований. Большинство из написанного я уже знаю, но добавлю для освежения памяти и полноты картины.

Основные алгоритмы

Проект включает в себя следующие алгоритмы:

- **Бинарный поиск**: тип поискового алгоритма, который последовательно делит пополам заранее отсортированный массив данных, чтобы обнаружить нужный элемент.

В данном проекте используются библиотеки, включая, но не ограничиваясь:

- **Algorithms.Library**: содрежит в себе классы для реализации различных алгоритмов.
- **Algorithms.Library.Tests**: содержит в себе тесты алгоритмов.

Цели разработки проекта

Данный проект разрабатывается исключительно в учебных целях. Он предназначен для изучения алгоритмов и структур данных. В процессе работы над проектом я стремлюсь:

- **Практика оценки сложности алгоритмов**.
- **Практика применения новых алгоритмов и структур данных**.
- **Развивать навыки программирования**: углубить знания и навыки в C# и проектировании информационных систем.
- **Улучшить подходы к тестированию**: научиться тестировать алгоритмы и обеспечивать их надежность.

Проект является лабораторной работой для получения навыков, необходимых для будущей карьеры в разработке, и не предназначен для коммерческого использования.

Quick Start

Установка и запуск

Для быстрого старта и локальной разработки, выполните следующие шаги:

1. Клонируйте репозиторий: `git clone git@github.com:xventrux/study-algorithms.git`
2. Перейдите в каталог проекта: `cd study-algorithms`

Binary search

Описание

Бинарный поиск гораздо более эффективный в сравнении с линейным поиском.
Бинарный поиск основан на идее деления данных на половины и последующем поиске в одной из них с последующим делением.

Визуализация

Код на c#

```
public class BinarySearcher<TParam> : ISearcher<TParam>
    where TParam : IComparable<TParam>
{
    private const int _notFoundValue = -1;

    public int Search(TParam[] sortedArray, TParam target)
    {
        ValidateInput(sortedArray, target);

        int left = 0;
        int right = sortedArray.Length - 1;

        while (left <= right)
        {
            int mid = GetMiddleNumber(left, right);
            int comparison = sortedArray[mid].CompareTo(target);
            if (comparison == 0)
            {
                return mid;
            }
            else if (comparison < 0)
            {
                left = mid + 1;
            }
```

```
            else
            {
                right = mid - 1;
            }
        }

        return _notFoundValue;
    }

    private void ValidateInput(TParam[] sortedArray, TParam target)
    {
        if (sortedArray is null)
            throw new ArgumentNullException(nameof(sortedArray));

        if (target is null)
            throw new ArgumentNullException(nameof(target));
    }

    private int GetMiddleNumber(int left, int right)
        => left + (right - left) / 2;
}
```