Отчет по теме №8 Алгоритмы поиска

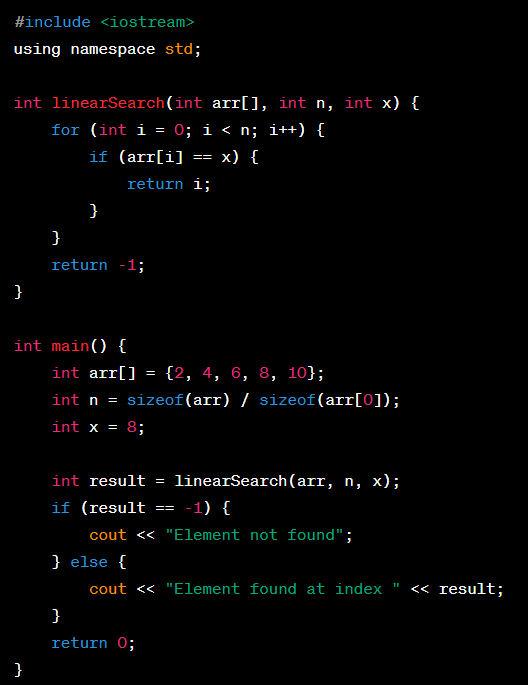
Работу выполнил:

Студент группы ИВТ(ВМК)-21

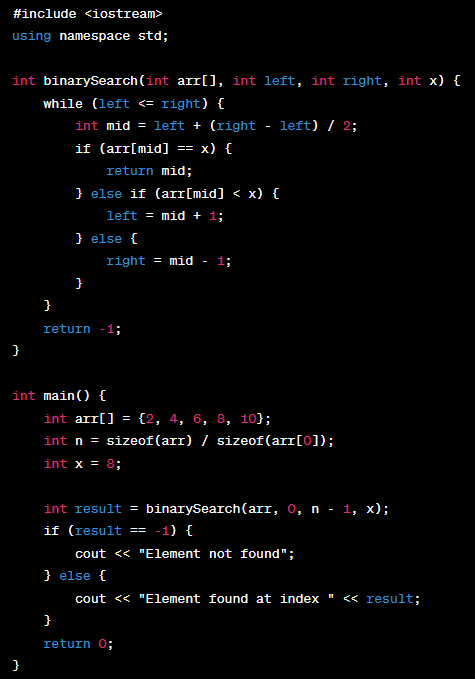
Рычков Родион Викторович

Алгоритмы поиска - это набор методов и техник, которые используются для поиска определенного элемента в наборе данных. Они могут быть применены в различных областях, таких как информатика, математика, биология и другие. Существует несколько различных типов алгоритмов поиска, которые различаются по способу работы и эффективности.

1. Линейный поиск Линейный поиск - это простой алгоритм поиска, который последовательно проверяет каждый элемент набора данных до тех пор, пока не будет найден искомый элемент. Он имеет линейную сложность времени O(n), где n - количество элементов в наборе данных.



1. Бинарный поиск Бинарный поиск - это алгоритм поиска, который работает с отсортированными наборами данных. Он делит набор данных на две части и сравнивает искомый элемент с элементом в середине набора. Если искомый элемент меньше, то бинарный поиск продолжает работу в левой половине набора данных, в противном случае - в правой половине. Бинарный поиск имеет логарифмическую сложность времени O(log n), где n - количество элементов в наборе данных.



1. Алгоритм прыжкового поиска Алгоритм прыжкового поиска - это улучшенный вариант линейного поиска, который работает с отсортированными наборами данных. Он делит набор данных на блоки фиксированного размера и выполняет линейный поиск в блоках, пока не будет найден блок, содержащий искомый элемент. Затем он выполняет линейный поиск внутри найденного блока. Алгоритм прыжкового поиска имеет сложность времени O(√n), где n - количество элементов в наборе данных.
2. Алгоритм интерполяционного поиска Алгоритм интерполяционного поиска - это алгоритм поиска, который работает с отсортированными наборами данных. Он использует линейную интерполяцию для оценки расположения искомого элемента в наборе данных и находит его с помощью бинарного поиска. Алгоритм интерполяционного поиска имеет сложность времени O(log log n) в среднем случае и O(n) в худшем случае, где n - количество элементов в отсортированном наборе данных.
3. Алгоритм Фибоначчиевого поиска Алгоритм Фибоначчиевого поиска - это алгоритм поиска, который также работает с отсортированными наборами данных. Он использует ряд Фибоначчи для выбора определенных индексов в наборе данных и выполняет бинарный поиск в выбранных подмассивах. Алгоритм Фибоначчиевого поиска имеет сложность времени O(log n), где n - количество элементов в наборе данных.
4. Алгоритм поиска в ширину (BFS) Алгоритм поиска в ширину - это алгоритм поиска, который работает с графами и деревьями. Он начинает с указанной вершины и ищет все соседние вершины на одном уровне, затем переходит к следующему уровню и повторяет процесс. Алгоритм поиска в ширину имеет сложность времени O(V+E), где V - количество вершин в графе, а E - количество ребер.
5. Алгоритм поиска в глубину (DFS) Алгоритм поиска в глубину - это алгоритм поиска, который также работает с графами и деревьями. Он начинает с указанной вершины и переходит к ее соседним вершинам, затем переходит к соседней вершине и продолжает до тех пор, пока не будет достигнут конец пути или найден искомый элемент. Алгоритм поиска в глубину имеет сложность времени O(V+E), где V - количество вершин в графе, а E - количество ребер.
6. Алгоритм A\* Алгоритм A\* - это алгоритм поиска пути в графах и деревьях, который используется в задачах планирования пути и искусственного интеллекта. Он использует эвристику для оценки стоимости перехода от одной вершины к другой и оптимизирует путь, выбирая вершины с наименьшей оценкой. Алгоритм A\* имеет сложность времени в худшем случае O(b^d), где b - фактор ветвления, а d - глубина решения.

Каждый из этих алгоритмов имеет свои преимущества и недостатки, и выбор конкретного алгоритма зависит от конкретной задачи и характеристик набора данных. Например, если требуется найти минимальный элемент в отсортированном массиве, то оптимальным будет использование бинарного поиска. Если необходимо найти кратчайший путь между двумя точками в графе, то алгоритм A\* может оказаться наиболее эффективным. Если же требуется найти все возможные пути между двумя точками в графе, то лучше использовать алгоритм поиска в глубину или в ширину.

Также важно учитывать время работы алгоритма, потребляемую память и другие ограничения на ресурсы. Например, алгоритмы поиска в ширину и в глубину могут потребовать большого количества памяти для хранения информации о вершинах, особенно при работе с большими графами.

В общем, выбор алгоритма поиска зависит от конкретной задачи, характеристик набора данных и ограничений на ресурсы, и может потребовать экспериментального анализа нескольких алгоритмов для определения наилучшего варианта.

///Добавить код и расчет вычислительной мощности для первых 2 .

Используемые источники:

https://works.doklad.ru/view/Xlwqp3pytds.html