Algorithms and data structures

lecture #4. Divide and Conquer Algorithms. Examples

Mentor: Rustam Khakov

Divide and Conquer Algorithms. Example

- Техника Разделяй и властвуй
- Алгоритмы «разделяй и властвуй»
- Преимущества и недостатки
- Example
  - Get max and min element
  - Count Inversions in an array
  - Binary Search
  - Closest Pair of Points
  - Merge Sort
  - Quick Sort

- Master Theorem (Additional)
  - Описание
  - Общая форма
  - Применение

Техника Разделяй и властвуй

- Divide: включает в себя разделение проблемы на более мелкие подзадачи
- Conquer: рекурсивно вызываем подзадачи до тех пор, пока они не будут решены
- Combine: объединить подзадачи, чтобы получить окончательное решение всей проблемы

Алгоритмы

- **Quick Sort** алгоритм сортировки. Алгоритм выбирает опорный элемент и переупорядочивает элементы массива таким образом, чтобы все элементы, меньшие, чем выбранный опорный элемент, перемещались в левую часть опорного элемента, а все большие элементы перемещались в правую сторону.
- **Merge Sort** также является алгоритмом сортировки. Алгоритм делит массив на две половины, рекурсивно сортирует их и, наконец, объединяет две отсортированные половины.
- **Closest Pair of Points** Задача состоит в том, чтобы найти ближайшую пару точек в наборе точек на плоскости xy. Задача может быть решена за время O(n^2) путем вычисления расстояний каждой пары точек и сравнения расстояний для поиска минимума. Алгоритм «разделяй и властвуй» решает проблему за время O(N log N).
- **Strassen’s Algorithm** — эффективный алгоритм умножения двух матриц. Простой метод умножения двух матриц требует 3 вложенных цикла и составляет O (n ^ 3). Алгоритм Штрассена умножает две матрицы за время O(n^2,8974).

Преимущества и недостатки

Преимущества алгоритма «разделяй и властвуй»:

- Сложная проблема решается легко.
- Делит задачу на подзадачи, поэтому ее можно решать параллельно, обеспечивая многопроцессорность.
- Эффективно использует кэш-память, не занимая много места
- Снижает временную сложность задачи

Недостатки алгоритма «разделяй и властвуй»:

- Включает в решение рекурсию, которая иногда медленная
- Эффективность зависит от реализации логики
- Это может привести к сбою системы, если в рекурсии есть ошибки

Examples

Get max element in array

найти максимальный элемент в заданном массиве.

Ввод: {40, 250, 80, 88, 240, 12, 148}
Вывод:
Минимальное число в данном массиве: 12
Максимальное число в данном массиве: 250

Examples

Binary Search

Дан отсортированный массив arr[] из n элементов.
Напишите функцию для поиска заданного элемента x в arr[] и возврата индекса x в массиве.

Примеры:
Ввод: arr[] = {11, 22, 44, 50, 60, 86, 114, 140, 145, 190}, x = 114
Вывод: 6
Объяснение: Элемент x присутствует в индексе 6.

Ввод: arr[] = {1, 24, 30, 46, 60, 100, 120, 133, 270}, x = 114
Вывод: -1
Объяснение: Элемент x отсутствует в arr[].

QuickSort - быстрая сортировка

Алгоритм быстрой сортировки является рекурсивным, метод на вход будет принимать границы участка массива от l включительно и до r не включительно. Работает следующим образом:

1) Процедура partition.
   a) Выбирается pivot элемент
   b) переставляются элементы участка массива таким образом, чтобы массив разбился на 2 части: левая часть содержит элементы, которые меньше pivot, а правая часть содержит элементы, которые больше или равны pivot.
2) Повторяется алгоритм на левой и правой частях

QuickSort - быстрая сортировка

```
partition(l, r):
    pivot = //выбираем опорный
    m = l
    for i = l ... r - 1:
        if a[i] < pivot:
            swap(a[i], a[m])
            m++
    return m
```

```
algorithm quicksort(A, low, high) is
    if low < high then
        p:= partition(A, low, high)
        quicksort(A, low, p)
        quicksort(A, p + 1, high)
```

QuickSort - быстрая сортировка

Достоинства:

- Один из самых быстродействующих (на практике) из алгоритмов внутренней сортировки общего назначения.
- Алгоритм очень короткий: запомнив основные моменты, его легко написать «из головы»
- Хорошо сочетается с механизмами кэширования и виртуальной памяти.
- Допускает естественное распараллеливание (сортировка выделенных подмассивов в параллельно выполняющихся подпроцессах).
- Работает на связных списках и других структурах с последовательным доступом, допускающих эффективный проход как от начала к концу, так и от конца к началу.

Недостатки:

- Сильно деградирует по скорости (до O(n^2) в худшем или близком к нему случае, что может случиться при неудачных входных данных.
- Прямая реализация в виде функции с двумя рекурсивными вызовами может привести к ошибке переполнения стека,
- Неустойчив.

Examples

Count Inversions in an array

Счетчик инверсии для массива указывает, насколько далек (или близок) массив от сортировки. Если массив уже отсортирован, то счетчик инверсии равен 0, а если массив отсортирован в обратном порядке, то счетчик инверсии будет максимальным.

Пример:

Ввод: arr[] = {8, 4, 2, 1}

Вывод: 6

Объяснение: Данный массив имеет шесть инверсий:

(8, 4), (4, 2), (8, 2), (8, 1), (4, 1), (2, 1).

Ввод: arr[] = {3, 1, 2}

Вывод: 2

Объяснение: Данный массив имеет две инверсии:

(3, 1), (3, 2)