Алгоритмы и структуры данных

2024 — 2025

Содержание

1 Медленные сортировки

1.1 Устойчивость сортировки

Сохранение порядка равных элементов

1.2 Варианты сортировок

Название	Сложность	Память	Присваивания	Устойчивость
Сортировка пузырьком	$O(n^2)$	$O(1)$	$O(n^2)$	1
Сортировка выбором	$O(n^2)$	$O(1)$	$O(n)$	0
Сортировка вставкой	$O(n^2)$	$O(1)$	$O(n^2)$	1

2 Быстрые сортировки

2.1 Оценка времени снизу

Исходя из принципа количества информации, с помощью бинарных вопросов можно угадать обратную перестановку не быстрее, чем за $O(n \log n) \sim O(n!)$

2.2 Сортировка слиянием

Название	Сложность	Память	Присваивания	Устойчивость
Сортировка слиянием	$O(n \log n)$	$O(n)$	$O(n \log n)$	1

1. Функция $merge(seq1, seq2)$
2. Вызываем рекурсивно, разделяя массив пополам, вплоть до очевидного случая
3. Когда мы проваливаемся на самый низкий уровень сравнения, то потребляется $\log n$
4. У рекурсивных вызовов большая константа

2.3 Подсчет количества инверсий

1. Сортируем массив
2. Считаем разницу в индексах и однозначно определяем количесвто инверсий с этим элементом

2.4 Сортировка кучей (HeapSort)

Название	Сложность	Память	Присваивания	Устойчивость
Быстрая сортировка	$O(n \log n)$	$O(\log n)$	–	0

2.5 Быстрая сортировка

Название	Сложность	Память	Присваивания	Устойчивость
Быстрая сортировка	$O(n \log n)$	$O(\log n)$	–	0

1. Выбираем случайный элемент
2. Разбиваем массив на 3 части
   (a) меньше x
   (b) больше x
   (c) равно x
3. Рекурсивно вызываем функцию в меньшей части, а в основной, в случае задачи оптимизации памяти, продолжаем работать в том же вызове функции
4. Худший случай - $O(n^2)$

2.6 Задача о поиске k-ой статистики

Название	Сложность	Память	Присваивания
Задача	$O(n)$	$O(1)$	–

1. Запускаем $partision$
2. Получаем 2 части, выбираем нужную
3. Работаем в одном вызове функции

2.7 Сортировка подсчетом

Название	Сложность	Память	Присваивания	Устойчивость
Сортировка подсчетом	$O(n+k)$	$O(k)$	–	$0/1$

1. Находим минимальный и максимальный элемент (k возможных значений)
2. Создаем, содержащий все возможные элементы

2.8 Сортировка по разрядам

Название	Сложность	Память	Присваивания	Устойчивость
Сортировка по разрядам	$O\big((n+k)\cdot p\big)$	$O(k)$	–	1

1. Применяем устойчивую версию сортировки подсчетом для каждого p разряда (которые состоят из k штук)
2. Очень сложно писать

3 Динамическое программирование - 1

3.1 Примеры задач

1. Задача о ступеньках (в том числе с запрещенными ступеньками)
2. Задача о поиске n - го числа фиббоначи
3. Задача о минимизации затрат
4. Задача о столбац и сварщике (нельзя выпиливать 2 столба подряд)
5. Задача о наибольшей возрастающей подпоследовательности
6. Задача о поиске наибольшей общей подпоследовательности в строках
7. Поиск кратчайшего представления числа

3.2 Инструменты

1. Добавление фиктивных элементов
2. Внедрение матриц при вычислении реккурентно-заданных последовательностей
3. Придумывать смысл формулы

3.3 Способы восстановления сертификата

1. В динамике определяем нужный вариант проходя в обратную сторону
2. Используем дополнительную память

3.4 Задача о поиске наибольшей общей подпоследовательности в строках (строим таблицу)

Строим таблицу, где каждая ячейка отображает длину подстроки на данный момент

3.5 Поиск кратчайшего представления числа

Находим оптимальную по длине запись слагаемого $k < n$ и представляем как сумму или произведение

4 Динамическое программирование - 2

4.1 Задача о обеде в столовой

4.1.1 Условие:

1. Поесть можно либо за деньги, либо за купон, который выдается за дорогой обед
2. Нужно минимизировать затраты

4.1.2 Решение:

1. Ввод дополнительного параметра, строим таблицу вместо строки
2. $dp[i][j]$ - количесвто купонов в i день с сегодняшней ценой обеда j

$$dp[i][j] = \min \left\{ \begin{array}{cc} dp[i-1][j+1] & \\ dp[i-1][j-1] + a_i & \text{если } a_i \geqslant 1000 \\ dp[i-1][j] + a_i & \text{иначе} \end{array} \right\}$$

4.2 Задача о покупке ткани

4.2.1 Условие:

1. T - требуемый размер ткани, N - количество магазинов
2. a_i - цена, b_i - оптовый барьер
3. c_i - оптовая цена, d_i - всего метров

4.2.2 Решение:

1. Пишем функцию f, которая возвращает минимальные затраты на покупку x метров ткани
2. Решить для 2-х магазинов
3. Решить для (1, 2) - го и 3 - го магазина
4. Продолжить для следующих
5. $dp[i][j]$ - затраты, первая группа магазинов i, второй магазин j

$$dp[i][j] = \min_{0,\dots,j} \{dp[i-1][k] + f(i, j-k)\}$$

4.3 Задача о елках

4.3.1 Условие:

1. N, T - количество елок и клу
2. w_i, e_i - отбрасываемая тень i - ой елки
3. Елку нельзя сажать, если на нее отбрасывается тень
4. Максимизировать количество елок

4.3.2 Решение:

1. $dp[i][j]$ - минимальная тень на восток (последний посаженный сорт), где последняя занятая клетка -i, j - количество елок