Java Basic

lecture #11. Greedy Algorithms

Mentor: <….>

lecture #11. Greedy Algorithms

- Introduction
- Greedy efficiency
- What is the difficulty
- Standard Greedy Algorithms
  - Egyptian Fraction
  - Huffman Coding
  - Fitting Shelves Problem
  - Minimum Swaps for Bracket Balancing
- Activity Selection problem – set 1 and set 2
- Cash exchange task
  - greedy algorithm works
  - greedy algorithm not working
- When the greedy algorithm fails

Введение в жадность

- Жадность — это алгоритмическая парадигма
- На каждом локальном шаге делает наилучший выбор в надежде, что итоговое решение будет оптимальным
- Жадные алгоритмы используются для задач оптимизации

Принцип жадного выбора

- К задаче применим принцип жадного выбора, если последовательность локально оптимальных выборов даёт глобально оптимальное решение

- Доказывается, что жадный выбор на первом шаге не закрывает пути к оптимальному решению

- Подзадача, возникающая после жадного выбора на первом шаге, аналогична исходной

Egyptian Fraction

Жадный алгоритм для египетских дробей — жадный алгоритм, который преобразует рациональные числа в египетские дроби, на каждом шаге выбирая наибольшую из возможных дробей, которая может быть использована в остаточной дроби.

Он же алгоритм Фибоначчи.

Алгоритм Фибоначчи осуществляет разложение $\frac{a}{b}$ путём последовательного проведения замены:

$$\frac{a}{b} = \frac{1}{[b/a]} + \frac{(-b) mod\ a}{b[b/a]}$$

Иначе говоря, на каждом шаге мы выбираем максимальную дробь вида $\frac{1}{n}$, не превосходящую $\frac{a}{b}$.

А на следующем шаге переходим к дроби $\frac{a}{b} - \frac{1}{n}$.

Поскольку каждый шаг разложения уменьшает числитель остаточной дроби, этот метод завершится за конечное число шагов. (И тем самым мы показали, что любую обыкновенную дробь можно разложить в египетскую).

Huffman Coding
Код Хаффмана — это особый тип кода оптимального префикса, который обычно используется для сжатия данных без потерь

https://www.techiedelight.com/huffman-coding/

Fitting Shelves Problem

Для заданной длины стены w и полок двух длин m и n найдите количество используемых полок каждого типа и оставшееся пустое пространство в оптимальном решении, чтобы пустое пространство было минимальным.
Большая из двух полок дешевле, поэтому она предпочтительнее.
Однако стоимость второстепенна, и первоочередной задачей является минимизация пустого пространства на стене.

Activiti selection set 1 - explanation

Задача. Даны заявки на проведение активити в некоторой аудитории. В каждой заявке указаны начало и конец активити. Нужно из всех заявок оставить как можно больше таким образом, чтобы они не пересекались.

Алгоритм решения сводится к тому, что бы выбрать заявки с минимальным временем окончания.

сортируем все заявки по времени окончания, на первом шаге выбираем активити которое закончится раньше всех (т.е. первый элемент в отсортированном по времени окончания списке), далее по очереди просматриваем остальные заявки и удовлетворяем т.е. в которых время начала активити больше или равно времени освобождения аудитории.

Activity Selection problem – set 2

- Дано n заданий, где каждое задание имеет крайний срок и стоимость.
- Также известно, что каждое задание занимает одну единицу времени, поэтому минимально возможный крайний срок для любого задания равен 1.
- Как максимизировать общую прибыль, если единовременно может быть запланировано только одно задание.

Input: Four Jobs with following deadlines and profits

JobID	Deadline	Profit
a	4	20
b	1	10
c	1	40
d	1	30

Output: Following is maximum profit sequence of job -> c, a

Cash exchange task

Задача.

- Монетная система некоторого государства состоит из монет достоинством 1, 2, 5, 10
- Требуется выдать сумму 68 наименьшим возможным количеством монет.

Жадный алгоритм решения этой задачи таков.

- Берётся наибольшее возможное количество монет достоинства.
- Таким же образом получаем, сколько нужно монет меньшего номинала, и т. д.
- **Для данной задачи жадный алгоритм не всегда даёт оптимальное решение**

When the greedy algorithm fails

Задача

- Найти выход из лабиринта при оптимальном движении: вправо и вниз

Жадный алгоритм решения этой задачи таков.

- Жадный алгоритм не делает лишних шагов
- Всегда движемся вправо и вниз, и т. д.
- **Для данной задачи жадный алгоритм не всегда даёт оптимальное решение или не даст его никогда**

Задача

- Классическая задача о рюкзаке
- Учитывая вес и стоимость n предметов, нам нужно положить эти предметы в рюкзак вместимостью W, чтобы получить максимальную общую стоимость в рюкзаке.
- **Для данной задачи жадный алгоритм не всегда даёт оптимальное решение**

Задачи относящихся к классу NP*(non-deterministic polynomial)****, жадные алгоритмы не дают оптимального решения.***

- задача коммивояжера
- задача минимальной раскраски графа
- задача выделения максимальной клики

SOLID

S

Принцип единственной ответственности (single responsibility principle)

Для каждого класса должно быть определено единственное назначение. Все ресурсы, необходимые для его осуществления, должны быть инкапсулированы в этот класс и подчинены только этой задаче.

O

Принцип открытости/закрытости (open-closed principle)

«программные сущности … должны быть открыты для расширения, но закрыты для модификации».

L

Принцип подстановки Лисков (Liskov substitution principle)

«функции, которые используют базовый тип, должны иметь возможность использовать подтипы базового типа не зная об этом».

I

Принцип разделения интерфейса (interface segregation principle)

«много интерфейсов, специально предназначенных для клиентов, лучше, чем один интерфейс общего назначения»

D

Принцип инверсии зависимостей (dependency inversion principle)

«Зависимость на Абстракциях. Нет зависимости на что-то конкретное»