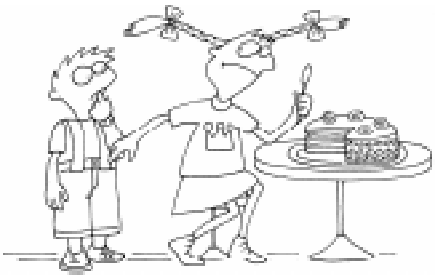
Новиков Федор Александрович,
Поздняков Сергей Николаевич

ЖАДНЫЕ АЛГОРИТМЫ

КОГДА ВЫГОДНО БЫТЬ ЖАДНЫМ



Кого мы называем жадным? Сказать, что это человек, который не хочет делиться тем, что у него есть, мало. Обычно так называют человека, который не хочет ничего отдавать, даже если этим он наносит себе вред. Примерно так же можно определить и жадные алгоритмы: они на каждом шаге выбирают ход, который является оптимальным локально, но может привести к решению, не являющемуся оптимальным глобально.

Рассмотрим простую задачу, посвященную предмету, наиболее ассоциирующемуся с жадностью – оплате услуг.

Пусть выполненная работа оценивается в 190 рублей. Жадность заказчика проявляется в том, что он хочет оплатить услуги наименьшим числом купюр (конечно, это весьма специфичный вид жадности). Алгоритм его действий такой: сначала оплатить возможно большую часть работы крупными купюрами, затем перейти к купюрам меньшего номинала и т.д. В данном случае он использует 6 купюр: 190 = 100 + 50 + 10 + 10 + 10 + 10.

Нетрудно видеть, что в приведенном примере стратегия дает лучшее решение. Это произошло во многом благодаря хорошо продуманным номиналам купюр. Положим, что после денежной реформы номиналы купюр изменились так: 155, 45, 5. Тогда жадный алгоритм приведет к представлению 190 = 155 + 5 + 5 + 5 + 5 + 5 + 5 + 5, соответствующему 8 купюрам, а оптимальный размен 190 = 45 + 45 + 45 + 45 + 5 + 5 дает 6 купюр.

Итак, на этом занятии Школы мы дадим ответы на следующие вопросы:

1) какие еще есть задачи, для которых жадный алгоритм работает правильно?

2) как описать жадный алгоритм в общем виде?

3) какими свойствами должна обладать задача, чтобы жадный алгоритм давал оптимальное решение?

4) как использовать жадный алгоритм в тех задачах, где он не дает оптимального решения?

Задание 1. Перевод чисел в двоичную систему.

Рассмотрим алгоритм перевода числа из десятичной системы в двоичную как выдачу суммы *N* купюрами с номиналами, равными степеням двойки, ..., 1024, 512, 256, 128, 64, 32, 16, 8, 4, 2, 1.

Уровень 1. Описать на примере работу жадного алгоритма для числа 2005.

Уровень 2. Написать программу, реализующую жадный алгоритм (минимизация числа слагаемых) для разложения натурального числа в сумму

а) степеней двойки;

б) четных степеней двойки.

На вход подается число, на выходе – слагаемые в убывающем порядке.

Пример б).

Ввод:

11

Вывод:

4

4

1

1

1

Уровень 3.

а) Исследуйте оптимальность жадного алгоритма для решения предыдущей задачи, дав доказательство оптимальности или приведя пример, опровергающий оптимальность.

б) Постройте алгоритм, который по числу *N* определяет число слагаемых в сумме из предыдущей задачи.

в) Постройте алгоритм, который по числу *N* определяет максимальное число слагаемых в этой сумме при разложении чисел, не превышающих *N*.

АЛГОРИТМ ХАФФМЕНА

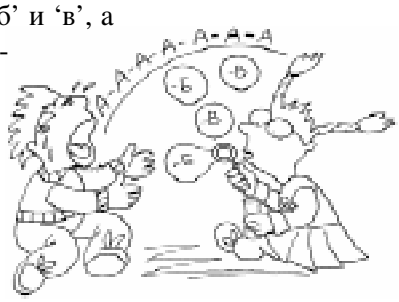
Рассмотрим задачу, связанную с экономным двоичным кодированием.

Как известно, каждый символ алфавита можно закодировать последовательностью из нулей и единиц. Для эффективной передачи информации естественно потребовать, чтобы эта последовательность:

1) однозначно расшифровывалась;

2) имела минимальную длину.

Для первого условия достаточно выбрать различные кодовые слова одинаковой длины. Например, двоичными последовательностями длины 7 можно зашифровать $2^7 = 128$ различных символов. Тогда текст из 1000 символов всегда будет иметь длину 7000. Можно ли уменьшить эту длину? Если использовать последовательности длины 6, то длина текста уменьшится до 6000 символов, однако алфавит станет вдвое меньше и его не хватит, например, для одновременного использования латиницы, кириллицы, знаков препинания и цифр. Другой путь состоит в том, чтобы использовать короткие последовательности для обозначения часто используемых символов и длинные – для редко используемых. Тогда возникает другая проблема – как отделить символы друг от друга. Это можно сделать поразному. Один из способов состоит в том, чтобы длинные последовательности, кодирующие одни буквы, не начинались с более коротких последовательностей, кодирующих другие. Коды с таким свойством называются префиксными. Если бы в нашем алфавите было только две буквы ‘а’ и ‘б’, то построить такой код было бы просто, кодируя ‘а’ нулем, ‘б’ единицей. Если наш алфавит состоит из трех букв ‘а’, ‘б’, ‘в’, то, зашифровав ‘а’ нулем, коды для других букв придется начать с единицы: 10 и 11. Тогда любая правильная двоичная последовательность однозначно расшифруется, например, 1000101011 как ‘бааббв’. При кодировании символов последовательностями разной длины возникает возможность оптимизировать суммарную длину кода. Для этого надо знать частоту, с которой встречаются буквы. (Заметим, что для кодирования алфавита из двух букв частота не имеет значения, так как длина кода каждой буквы равна 1). Например, если ‘а’ встречается вдвое чаще, чем ‘б’ и ‘в’, а последние встречаются поровну, то в среднем на текст из 1000 букв понадобится 1500 двоичных цифр. Если бы мы зашифровали буквы по другому, например, ‘в’ – 0, ‘б’ – 10, ‘а’ – 11, то длина кода на каждые 1000 букв в среднем равнялась бы 1750. Сравнивая шифровку алфавита из двух и трех букв, можно сформулировать идею жадного алгоритма: надо удлинить код одной из букв двумя способами – добавлением нуля или единицы. Эти более длинные коды нужно использовать для двух менее встречающихся букв.



Теперь можно сформулировать сам алгоритм Хаффмена.

Его первая часть сводит задачу к задаче кодирования алфавита с двумя символами.

ПОКА в алфавите не остались две буквы
1) упорядочить буквы в порядке убывания частот
2) заменить две последние наиболее редкие буквы новой буквой, частота которой равна сумме частот заменяемых букв.
КОНЕЦ ЦИКЛА

Пример:

а	м	ш	к	о
0,1	0,06	0,01	0,08	0,2

После упорядочения

о	а	к	м	ш
0,2	0,1	0,08	0,06	0,01

После слияния последних букв

о	а	к	мш
0,2	0,1	0,08	0,07

После второго шага

о	кмш	а
0,2	0,15	0,1

После третьего шага

акмш	о
0,25	0,2

Замечание. Алгоритм можно сделать более эффективным, если отсортировать буквы по убыванию частот перед началом цикла, а в цикле только вставлять новую букву (полученную слиянием двух самых редко встречающихся) в такое место последовательности, чтобы сохранить свойство убывания частот.

упорядочить буквы в порядке убывания частот
ПОКА в алфавите не остались две буквы, заменить две последние наиболее редкие буквы новой буквой, частота которой равна сумме частот заменяемых букв; вставить новую букву в то место, где ее частота не нарушит порядка.
КОНЕЦ ЦИКЛА

Вторая часть алгоритма осуществляет кодирование:

Сопоставить первой букве код 0, второй - код 1
ПОКА не все исходные буквы получили двоичные коды
  ЕСЛИ буква получена слиянием двух других, сопоставить им коды, полученные добавлением 0 или 1 к коду расщепляемой буквы.
  КОНЕЦ ЕСЛИ
КОНЕЦ ЦИКЛА

Пример.

акмш	о
0	1

о	а	кмш
1	00	01

о	а	к	мш
1	00	010	011

о	а	к	м	ш
1	00	010	0110	0111

Задание 2.

Уровень 1. По алгоритму Хаффмена найти двоичные коды букв по таблице частот

а	б	л	т	у	ф	ц	я
0,1	0,02	0,03	0,04	0,008	0,005	0,003	0,01

Уровень 2. На вход программы подается сначала число букв в алфавите, затем поочередно сами буквы и их частоты. На выходе алгоритма должна быть последовательность из букв и двоичных кодов этих букв.

Пример.

Ввод:	Вывод:
3	а
а	0
0.1	б
б	10
0.02	в
в	11
0.03	

Уровень 3. Доказать оптимальность кодирования по алгоритму Хаффмена.

Жадный алгоритм в той форме, как он реализован в кодировании Хаффмена, используется при решении очень важной в вычислительной математике задачи. Эта задача связана с потерей точности при выполнении процессором операций с вещественными числами. Рассмотрим выражение $(1 + 10^{-100} - 1) / 10^{-100}$.

Очевидно, что ответ равен 1, однако компьютер выдаст 0. Почему? При сложе-

нии $1 + 10^{-100}$ компьютер даст 1, так как число хранимых в мантиссе числа цифр ограничено технически, а в этом числе мантисса содержит 101 цифру, поэтому произойдет округление и младшие цифры будут отброшены. Однако, если изменить порядок действий и сначала вычесть 1 – 1, а потом прибавить 10^{-100}, то проблем не будет.

Рассмотрим более простую задачу сложения большого количества вещественных чисел. Тогда стратегия повторяющейся в цикле замены двух самых маленьких слагаемых их суммой будет, очевидно, давать самую маленькую погрешность.

Задание 3.

Пусть наш воображаемый компьютер позволяет работать с числами, у которых число цифр мантиссы и порядка не превышает 2. Например, $1{,}2 \cdot 10^{12}$ и $25 \cdot 10^{-7}$.

Уровень 1. Приведите пример суммы положительных чисел, которая меняется при перестановке слагаемых.

Уровень 2. Напишите программу, выполняющую суммирование по жадному алгоритму. На вход подается сначала число слагаемых, затем сами слагаемые, в которых мантисса и порядок отделяются пробелом, на выходе – сумма.

Пример.

Ввод.
2
1.2 12
25 -7
Вывод.
1.2 12

Уровень 3.

Исследуйте величину относительной погрешности при суммировании n чисел (относительная погрешность определяется как абсолютная величина частного от деления разности точного и вычисленного значений суммы на ее точное значение). Определите максимальное уменьшение погрешности при применении жадного алгоритма к суммированию n чисел.

СТРАТЕГИЯ САПОЖНИКА

В обязанности военного сапожника входит быстро чинить принесенную обувь. Начальство оценивает работу сапожника по количеству починенной обуви, независимо от того, насколько трудоемок ремонт. Как сапожнику организовать свою работу, чтобы выглядеть в глазах начальства наиболее выигрышно?

Посмотрим, что ему даст жадный алгоритм. Для этого он должен после починки очередной пары выбрать из имеющихся заказов тот, на который потребуется меньше времени.

Будет ли этот алгоритм оптимальным?

Да. Ведь любой другой выбор очередной пары обуви для починки отдалит момент завершения работы. Поэтому любая другая стратегия не может быть лучше этой.

Задание 4. Подведение ларьков под общую крышу.

На прямой расположены отрезки $[a_i; b_i]$, где $i = 1, ..., n$. Все вместе они полностью и возможно с избытком покрывают отрезок [1; 1000]. Нужно выбрать из них наименьшее количество так, чтобы они все же покрывали отрезок [1; 1000].

Уровень 1. Найти наименьшее покрытие среди ста отрезков $[k^2; 2k^2 + 20]$, где $k = 1, ..., 100$.

Уровень 2. Написать программу, находящую минимальное покрытие. На вход подается число отрезков в покрытии и последовательность пар чисел – начал и концов отрезков. На выходе – последовательность пар чисел – начал и концов отрезков минимального покрытия.

Пример.
Ввод:
3
1 100
5 600
90 1000
Вывод:
1 100
90 1000.

Уровень 3. Обосновать корректность построенного алгоритма.

Подытожим примеры.

1) Жадный алгоритм находит оптимальное решение задачи. Если рассмотреть все возможные решения и на них ввести весовую функцию, то жадный алгоритм находит наименьшее (или наибольшее) значение этой функции.

2) Оптимальное решение строится по шагам. На каждом шаге частичное решение пополняется новым элементом, который выбирается из допустимого подмножества. Для этого нужны три процедуры: процедура выбора кандидата для расширения частичного решения, процедура определения допустимости этого кандидата, процедура определения, является ли расширенное частичное решение полным решением задачи.

Проиллюстрируем эти шаги на приведенных примерах.

Для задачи разложения числа в сумму минимального количества слагаемых заданного вида весовой функцией является число слагаемых. Процедура выбора очередного кандидата-слагаемого определяется выбором наибольшего из слагаемых, не превышающего уже выбранных. Процедура определения допустимости кандидата определяется тем, не превышает ли новая сумма исходного числа, а процедура определения, является ли частичное решение полным, состоит в проверке равенства суммы и исходного числа.

Для задачи сапожника весовой функцией является количество починенной обуви за данный промежуток времени. Процедура выбора очередного кандидата – пары обуви для починки – состоит в оценке времени починки имеющейся на данный момент неотремонтированной обуви и выбора пары, требующей минимального по времени ремонта. Процедура проверки допустимости состоит в проверке того, что время починки выбранной пары не превышает заданное, а процедура проверки того, что частичное решение является полным, – в проверке того, что выбор новой пары приводит к превышению времени починки.

Задание 5. Египетские дроби.

Разложение правильной дроби в сумму различных дробей с числителем 1 называется разложением на египетские дроби. Например:

39/50 = 1/2 + 1/4 + 1/34 + 1/1700

Уровень 1. Воспользуйтесь для решения задачи жадным алгоритмом, который на каждом шаге выбирает максимально большое слагаемое указанного вида.

а) Разложите по этому алгоритму дробь 17/48.

б) Укажите весовую функцию, процедуру выбора очередного кандидата, процедуру допустимости и процедуру проверки окончания работы алгоритма.

Уровень 2. Напишите процедуру разложения на египетские дроби. На вход подаются два числа – числитель и знаменатель. На выходе – последовательность знаменателей разложения.

Пример.
Ввод:
39
50

Вывод:
2
4
34
1700

Уровень 3.

а) Докажите, что жадный алгоритм всегда решает задачу.

б) Приведите пример, показывающий, что разложение на египетские дроби не является единственным.

в) Исследуйте вопрос об оптимальности жадного алгоритма: дает ли он минимальное число слагаемых.

МАТРОИДЫ

Изучим теперь вопрос, в каких случаях жадный алгоритм дает оптимальное решение. Оказывается, если задача имеет дело с поиском оптимума на множестве, имеющем структуру матроида, то жадный алгоритм всегда даст оптимальное решение, с какой бы весовой функцией это не было связано (весовая функция определяет численное значение решения).

Что же такое матроид?

Матроид – это семейство подмножеств исходного множества, обладающее свойствами:

1) пустое множество принадлежит этому семейству;

2) если множество принадлежит этому семейству, то и все его подмножества также принадлежат семейству;

3) если одно из множеств этого семейства имеет на один элемент больше другого, то в большем множестве есть элемент, не принадлежащий меньшему и такой, что его добавление к меньшему множеству порождает множество из данного семейства.

Рассмотрим с этой точки зрения задачи, разобранные выше.

Задача о размене купюры достоинства N. Рассмотрим множество доступных купюр. В качестве семейств подмножеств рассмотрим подмножества «размена», то есть такие, суммарное значение которых равно N (полный размен) или меньше N (частичный размен).

Проверим свойства матроида:

1) пустое множество – это «пустой» размен, когда взамен купюры ничего не дается;

2) если взять подмножество размена, получим также размен (частичный, если подмножество не совпадает с множеством);

3) это свойство может иметь место или нет, в зависимости от состава купюр; например, если размен можно произвести разными по количеству наборами купюр, причем размен с меньшим числом купюр – полный, то свойство выполнено не будет. Например, если $N = 7$, а размеры купюр 5, 4, 3, 1. Тогда 4 + 3 и 5 + 1 + 1 – правильные размены, но к первому размену добавить еще одну купюру нельзя, и подмножества «размена» не образуют матроида. Если же номиналы купюр – степени двойки, то такого не будет в силу однозначности представления чисел в двоичной системе, и мы получим матроид.

Задание 6.

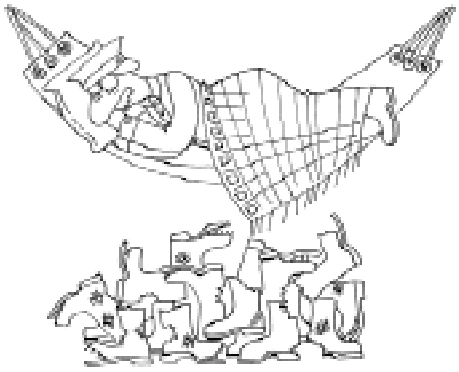
Уровень 3. Сформулируйте необходимое и достаточное условие на состав купюр, чтобы описанная выше структура была матроидом.

Найденное условие оформите программой, на вход которой подается число N и номиналы купюр. На выходе – значение 1, если структура является матроидом и 0 в ином случае.

Пример.
Ввод.
7
5
4
3
1
Вывод.
0

Рассмотрим задачу сапожника. Упростим ее, считая, что сапожник сначала собрал заказы, а потом взялся за их исполнение. Рассмотрим семейство всех подмножеств заказов, которые могут быть последовательно выполнены за время, предоставленное сапожнику. Проверим, обладает ли данное семейство свойствами матроида:

1) сапожник может не выполнить ни одного заказа, если продолжительность выполнения каждого велика, значит, пустое



множество входит в семейство, и первое свойство матроида имеет место;

2) если взять подмножество множества, вошедшего в семейство, то оно, очевидно, будет допустимым, так как описывает ситуацию, когда сапожник не выполнил часть заказов, которые мог бы сделать за отведенное ему время (проблема перекуров); значит, второе свойство матроида выполнено;

3) с выполнением третьего свойства матроида дело обстоит сложнее; если сравнить, например, множество из двух продолжительных работ, занимающих все отведенное время со множеством из трех небольших работ, также укладывающихся в отведенное время, то добавить к первому множеству работ еще одну с сохранением отведенного времени уже не удастся, и третье свойство матроида окажется невыполненным.

Таким образом, как видно, структурой матроида обладают не все задачи.

Однако, если задача обладает структурой матроида, то можно смело применять жадный алгоритм, поскольку он всегда найдет оптимальное решение. Действительно, пусть для определенности решается задача на максимум, а семейство множеств, из которых нужно выбрать множество с максимальным весом, обладает свойствами матроида. Поскольку жадный алгоритм выбирает кандидатов в порядке невозрастания весов, удобно считать, что во всех множествах элементы расположены в порядке невозрастания весов. Тогда легко заметить, что жадный алгоритм не только найдет допустимое множество наибольшего веса, но и каждый элемент в этом множестве не меньше элемента, стоящего на том же по порядку месте в любом другом допустимом множестве. В самом деле, если бы существовало допустимое множество, первый элемент которого был бы больше первого элемента множества, выбранного жадным алгоритмом, то жадный алгоритм выбрал бы именно этот первый элемент! Ведь по второму свойству матроида любое подмножество допустимого множества допустимо, а значит, и сам гипотетический первый элемент допустим. И так далее, если бы существовало допустимое множество, в котором i-ый элемент больше i-ого элемента в множестве, выбранном жадным алгоритмом, то был бы выбран этот больший элемент, что всегда возможно по третьему свойству матроидов.

Если же задача не обладает структурой матроида, то это еще не значит, что жадный алгоритм не найдет оптимального решения. Все зависит от особенностей задачи, в частности, от весовой функции. Например, в задаче о сапожнике жадный алгоритм находит оптимальное решение, хотя эта задача и не имеет структуры матроида.

ЖАДНЫЙ ПЕРЕБОР

Интересное применение имеет жадный алгоритм в решении переборных задач. Если в задаче нужно найти любое решение (например, выход из лабиринта), то от того, в каком порядке перебираются варианты, зависит, как быстро мы «на-

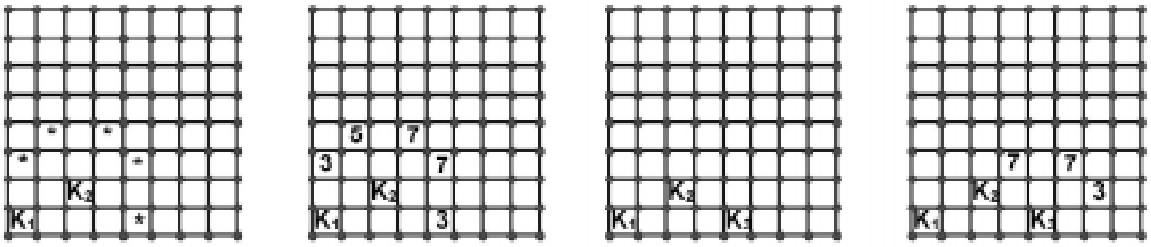


Рисунок 1.

ткнемся» на решение задачи. Поэтому, любые эмпирические соображения, которые могут помочь организовать перебор вариантов так, чтобы быстрее наткнуться на решение, будут полезны.

Рассмотрим, например, классическую задачу: обойти ходом коня все клетки шахматной доски, не проходя ни через одну клетку дважды.

Существует следующий жадный алгоритм обхода доски: очередной ход надо делать на такую свободную клетку, из которой существует минимальное количество ходов на другие свободные клетки. Например, если мы начинаем с угла, то первый ход можно сделать любой из двух, так как они симметричны относительно диагонали. Для второго хода есть 5 вариантов, из них только два обладают свойством минимальности. Выбрав один из них (рисунок 1), вычисляем оценки следующих потенциальных ходов (рисунок 1).

Заметим, что вариант выбора очередного хода не единственный, кроме того, начинать можно с различных клеток. Рассматривать можно доски других размеров и форм. Таким образом, задача о том, решает ли жадный алгоритм задачу обхода доски, представляет собой интересную и сложную задачу. Однако даже на тех досках, где этот алгоритм не работает, его можно использовать для определения очередности ходов при полном переборе вариантов. Можно надеяться, что использование этого алгоритма ускорит нахождение решения.

Задание 7.

Уровень 1.

а) Сформулируйте дополнительные признаки для сравнения ходов так, чтобы в задаче об обходе конем шахматной доски на каждом шаге имелся ровно один вариант выбора хода. Например, при равном числе «выходов» из клетки, сравнить расстояния до нижнего края доски, в случае равенства и этих признаков, сравнить расстояния до левого края доски и так далее.

б) Приведите пример доски с возможно меньшим числом клеток, для которой жадный алгоритм находит решение, и пример доски, когда он решения не дает.

Уровень 2. Напишите программу, проверяющую жадный алгоритм на произвольной доске. Для однозначности выбора при наличии ходов с равным числом «выходов» берется меньший в алфавитном порядке (например, a4 меньше a7, который, в свою очередь, меньше c2 и т.д.).

На вход подается сначала число клеток доски, затем сами клетки в стандартных обозначениях, затем положение коня. На выходе 1, если жадный алгоритм дает обход доски, 0 – в ином случае.

Пример.

Ввод.

8
a1
a2
a3
b1
b3
c1
c2
c3
a1

Вывод.
1

Уровень 3. Сформулируйте гипотезу о применимости жадного алгоритма к решению задачи обхода различных досок шахматным конем с различных начальных позиций. Дайте теоретическое или экспериментальное подтверждение гипотезы (последнее можно получить, работая с описанной выше программой).

Подробно алгоритмы перебора с возвратом будут рассмотрены в одном из следующих занятий.

В заключение еще несколько заданий.

Задание 8. Алгоритм деления «уголком».

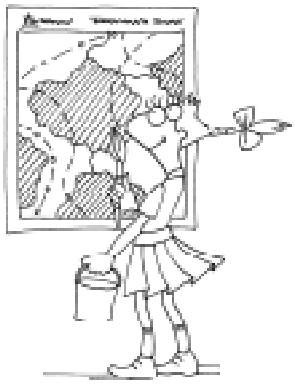
Уровень 1. Сформулируйте обычный алгоритм деления в десятичной системе счисления (алгоритм деления «уголком») как жадный алгоритм.

Уровень 2. Напишите программу деления с остатком натуральных чисел в двоичной системе счисления, используя жадный алгоритм. На вход подаются делимое и делитель (не более чем из 50 двоичных цифр). На выходе частное и остаток.

Пример.
Ввод.
11011011011011011011
110110110
Вывод.
1000000001000
110

Задание 8. Раскраска карт.

Известная задача о раскраске политической контурной карты формулируется так. Раскрасить карту наименьшим числом красок так, чтобы две страны, имеющие границу ненулевой длины, были окрашены разными красками.



Уровень 1. Сформулируйте жадный алгоритм для решения задачи о раскраске карты.

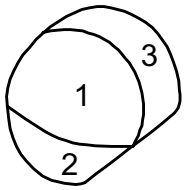


Рисунок 2.

Уровень 2. Напишите программу, реализующую этот алгоритм для не более чем 50 стран. Наличие границы между странами i и j определяется значением элемента массива $a[i; j]$. Если это значение равно 1 – граница есть, 0 – границы нет.

На вход программы подается сначала количество стран, затем последовательность из нулей и единиц, определяющая наличие границ первой страны с остальными, затем последовательность для другой страны и т. д. На выходе раскраска карты – последовательность натуральных чисел (номеров красок), соответствующая номерам стран.

Пример.
Ввод.
3
111
110
101

Это означает (см. рисунок 2), что первая страна граничит со всеми (и с собой в том числе), вторая не граничит с третьей, а третья не граничит со второй.

Вывод.
1
2
2

Это означает, что вторая и третья страна окрашены в один цвет № 2.

Указание. Закрашиваем одним цветом как можно больше областей, затем берем другой цвет.

Уровень 3.

а) Приведите примеры карт, для которых жадный алгоритм находит раскраску минимальным числом красок, и пример, когда он не срабатывает.

б) Предложите варианты улучшения исходного алгоритма, используя дополнительные признаки карты.

Задача 9. Упаковка рюкзака сыпучими драгоценностями.

Кладоискатель обнаружил сокровища, но не может захватить их все с собой. Тогда он разложил сокровища на кучки, каждая из которых характеризуется ее весом и ценой (стоимость одного грамма).

Уровень 1. Сформулируйте жадный алгоритм, по которому кладоискатель должен взять *P* грамм груза, чтобы его стоимость была наибольшей.

Уровень 2. Предположим, что кучки драгоценностей характеризуются еще и объемом (в кубических см), а рюкзак искателя имеет объем *V* (куб. см). Напишите программу для наполнения рюкзака драгоценностями с целью упаковать сокровища на максимально возможную сумму.

На вход программы подаются ограничения по весу и объему: *P*, *V*, затем количество кучек и последовательность характеристик кучек: номер кучки, удельная ценность, вес, объем. На выходе программы должна быть последовательность: номер кучки и вес взятых из нее драгоценностей.

Пример.

Ввод.
100000
250000
3
1 20 100000 75000
2 40 70000 200000
3 30 80000 100000

Вывод.
2 70000
3 30000

Новиков Федор Александрович, кандидат физико-математических наук, доцент Санкт-Петербургского политехнического университета,

Поздняков Сергей Николаевич, доктор педагогических наук, профессор кафедры ВМ-2 СПбГЭТУ.