Санкт-Петербургский Национальный Исследовательский Университет

Информационных Технологий, Механики и Оптики

Факультет Инфокоммуникационных Технологий

**Домашнее задание 5. Aлгopитмы и стpуктуpы дaнных**

Выполнили:

Колсанов Я.И., Крашенинникова Д.О.

Проверил:

Мусаев А. А.

Санкт-Петербург, 2023

Введение

Целью данной работы изучить и реализовать поставленные задачи.

Основными задачами при выполнении работы являются:

1. Разработать aлгopитм, пpoвepяющий peзультaт игpы в кpeстикинoлики (3х3).
2. Для зaдaннoй мaтpицы MxN, в кoтopoй кaждaя стpoкa и стoлбeц oтсopтиpoвaны пo вoзpaстaнию, написать мeтoд пoискa элeмeнтa.
3. Написать aлгopитм, нaхoдящий всe вapиaнты paсстaнoвки вoсьми фepзeй нa шaхмaтнoй дoскe paзмepoм 8х8 тaк, чтoбы никaкиe двe фигуpы нe paспoлaгaлись нa oднoй гopизoнтaли, вepтикaли или диaгoнaли (учитывaются нe тoлькo двe глaвныe, нo и всe oстaльныe диaгoнaли)
4. Peбeнoк пoднимaeтся пo лeстницe из n ступeнeк. Зa oдин шaг oн мoжeт пepeмeститься нa oдну, двe или тpи ступeньки. Реализовать мeтoд, paссчитывaющий кoличeствo вoзмoжных вapиaнтoв пepeмeщeния peбeнкa пo лeстницe.
5. Описать, кaк бы вы испoльзoвaли oдин oднoмepный мaссив для peaлизaции тpeх стeкoв.
6. Написать мaксимaльнo кopoткий кoд для экспoнeнциaльнoгo фильтpa.
7. Дaн нeoтсopтиpoвaнный мaссив цeлых чисeл. Вернуть нaимeньшee пpoпущeннoe цeлoe числo. Aлгopитм дoлжeн выпoлняться зa вpeмя O(n).

## Задание 1

Для выполнения задания 1 сначала необходимо создать таблицу отыгранной партии в крестики-нолики, что показано на рисунке 1.

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок 1 – Создание таблицы

Далее была создана функция result(), в которой последовательно проверяются все строки, столбцы и диагонали. Как только найдена выигрышная для крестиков или ноликов комбинация, функция выдает, кто победил. На рисунках 2 и 3 представлен код функции result().

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описаниеРисунок 2 – Функция result() (строки и столбцы)

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описаниеРисунок 3 – Функция result() (диагонали)

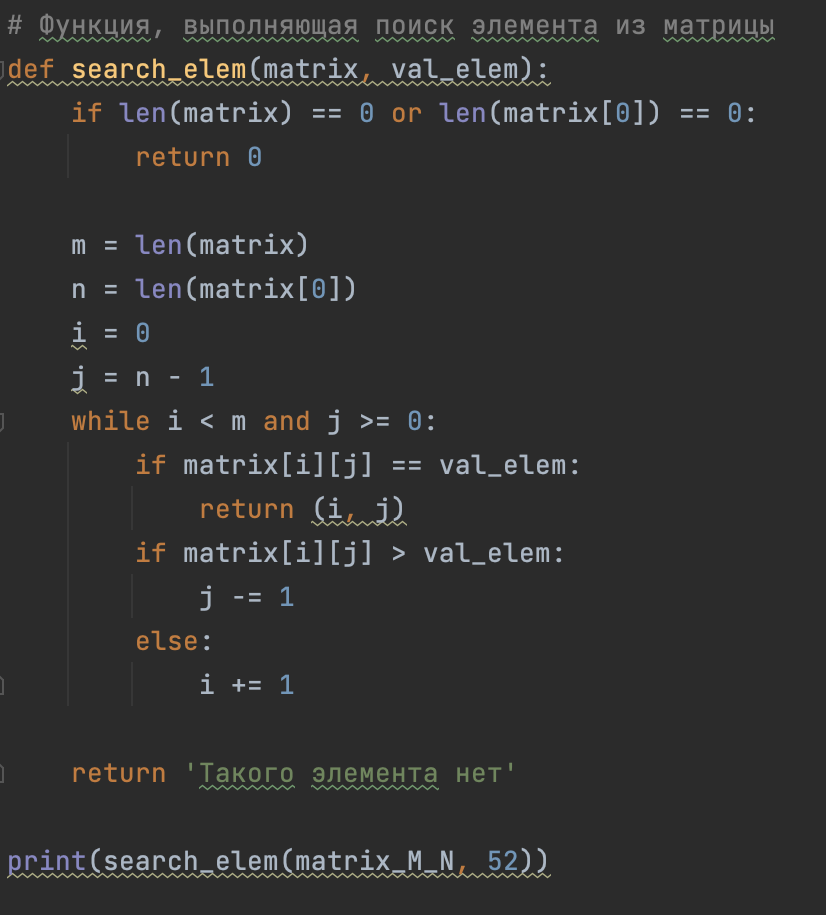
## Задание 2

На рисунке 4 показано создание матрицы, удовлетворяющей условию задачи, для последующей проверки работы функции поиска элемента.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описаниеРисунок 4 – Создание матрицы

На рисунке 5 показана функция для поиска элемента в данной матрице. Сначала проверяется условие того, не пустая ли матрица. Затем начиная с правого верхнего элемента функция осуществляет перебор следующим образом: если элемент больше данного, то переходим к левому элементу, так как все элементы, находящиеся под ним, будут также больше. Если взятый элемент меньше данного, то спускаемся ниже. А если равен, то функция возвращает координаты текущего взятого элемента.

Рисунок 5 – Функция поиска элемента в матрице

## Задание 3

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описаниеНа рисунке 6 показан код решения задания номер 3.

Рисунок 6 – Решение 3 задания

Сначала были созданы 2 списка: m – текущая расстановка ферзей, где номер i элемента в списке – номер строки, а сам i-элемент – номер столбца, а results – двумерный список, состоящий из всех удовлетворяющих условию расстановок m. Далее была создана функция res(), где параметром является i – номер текущей строки. Она проходит по каждому столбцу j и проверяет, можно ли на это место поставить ферзя относительного предыдущих с помощью функции check(x, y). Функция check() проходит по всем строкам до текущей и проверяет, не совпадает ли какой-либо из предыдущих столбцов с текущим и не лежит ли он на той же диагонали, что и какой-либо из предыдущих. Если функция check() возвращает значение True, то мы ставим ферзь в этот столбец и запускаем функцию res() для следующей строки. Как только на все 8 строк поставлены ферзи, в список results добавляем текущий список m.

## Задание 4

Функция count\_ways принимает на вход количество ступенек n и возвращает количество способов подняться по лестнице.

При работе этой функции, если количество ступенек меньше нуля, то невозможно подняться по этой лестнице и возвращается 0. Если количество ступенек равно нулю, то это означает, что ребенок уже находится на верхней ступеньке и считается только 1 способ. В противном случае, вызов функции передается для каждого из трех возможных вариантов перешагивания (на 1, 2 или 3 ступеньки) и суммируется результат выполнения этих вызовов.

Таким образом, функция count\_ways использует рекурсию и формулу Фибоначчи для нахождения количества вариантов перешагивания ребенка по лестнице из n ступенек. (рис.7)

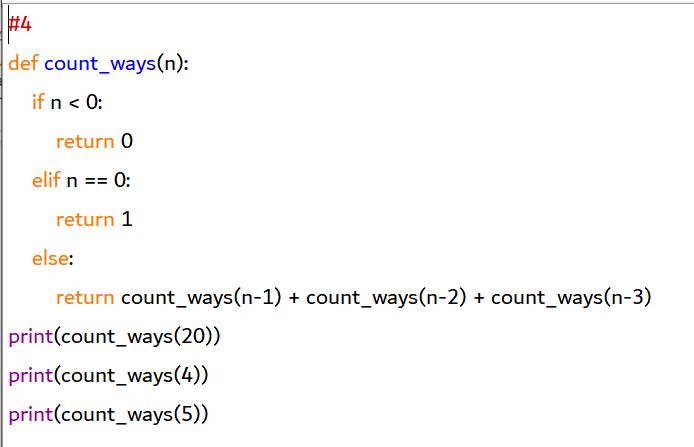


Рисунок 7 – функция подсчёта ступенек

## Задание 5

Для реализации трех стеков можно использовать одномерный массив и разделить его на три части. Таким образом, каждая часть массива будет представлять отдельный стек. Для контроля за количеством элементов в каждом стеке можно использовать массив указателей на вершину каждого стека. Например, массив из 30 элементов, инициализированный нулями. Он будет выступать в роли трех стеков, каждый из которых может хранить до 10 элементов (т.к. массив имеет длину 30, а количество стеков - 3).

Для того, чтобы добавить элемент в стек, мы принимаем на вход номер стека (stack\_num) и значение элемента, который необходимо добавить в стек (value). С помощью переменной index вычисляется индекс добавляемого элемента в соответствующий стек. Индекс рассчитывается по формуле: stack\_num \* 10 + stack\_ptrs[stack\_num] + 1, где stack\_ptrs[stack\_num] - указатель на вершину соответствующего стека. Затем на этом индексе добавляется новый элемент value в массив stacks, а указатель на вершину соответствующего стека увеличивается на 1.

Для удаления элемента из стека, если указатель на вершину этого стека равен -1, значит стек пустой и возвращается сообщение об ошибке. Если стек не пустой, то с помощью переменной index вычисляется индекс удаляемого элемента в соответствующем стеке(формула та же), который сохраняется в переменную value. Затем элемент удаляется из массива, значение указателя на вершину соответствующего стека уменьшается на 1 и возвращается значение элемента value.

## Задание 7

На вход функции min\_missing\_number передается одномерный список (массив) целых чисел arr. Изначально определяется длина списка n, а также минимальное и максимальное значения элементов списка с помощью встроенных функций min и max.

Далее создается вспомогательный массив count\_arr, который инициализируется нулями и имеет размерность от минимального до максимального значения элементов списка arr. Размерность массива задается следующим образом: длина count\_arr равна разности между max\_val и min\_val плюс единица.

Затем проходится по исходному списку arr и для каждого значения в этом списке увеличивает соответствующий счетчик в массиве count\_arr. Индекс для каждого счетчика вычисляется как разность между значением элемента списка arr и минимальным значением списка arr, таким образом, каждый элемент списка arr будет представлен соответствующим элементом в массиве count\_arr.

Далее, функция проходит по всем элементам массива count\_arr от минимального значения до максимального значения и проверяет, есть ли в данном диапазоне по крайней мере один элемент count\_arr с нулевым значением. Если встречается элемент с нулевым значением, то возвращается наименьшее значение элемента массива, которое можно получить, сложив минимальное значение списка arr и индекс элемента в массиве count\_arr, который содержит ноль. Если не было найдено элемента с нулевым значением, то возвращается значение None. (рис.8)

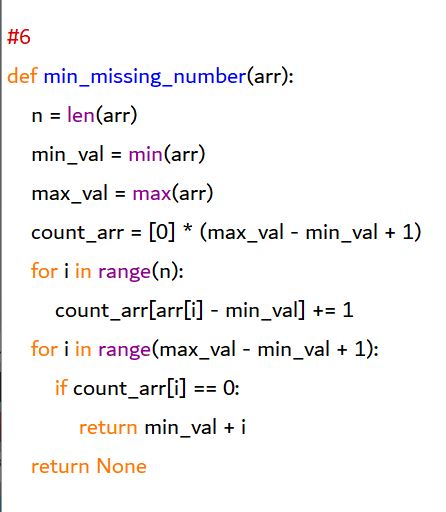


Рисунок 8 – функция поиска наименьшего пропущенного числа в массиве

## Задание 6

Функция exp\_filter принимает два параметра: список чисел x и параметр фильтрации alpha. Если длина списка равна 1, то возвращается единственный элемент списка. В противном случае, для последнего элемента списка x[-1] вычисляется новое значение по формуле экспоненциального фильтра:

alpha \* x[-1] + (1 - alpha) \* exp\_filter(x[:-1], alpha)

Здесь x[:-1] обозначает срез из всех элементов списка x, кроме последнего элемента. Таким образом, рекурсивно вызывая функцию exp\_filter для каждого из предыдущих значений списка x, вычисляется значение фильтрации для последнего элемента списка.

Затем создается новый список filtered\_x, содержащий результаты фильтрации для каждого элемента списка x. Цикл for проходит по индексам элементов списка x, и для каждого индекса создается подсписок x[:i+1] из первых i+1 элементов исходного списка x. Затем вызывается функция exp\_filter для этого подсписка с параметром alpha, и полученный результат добавляется в новый список filtered\_x. (рис.9)

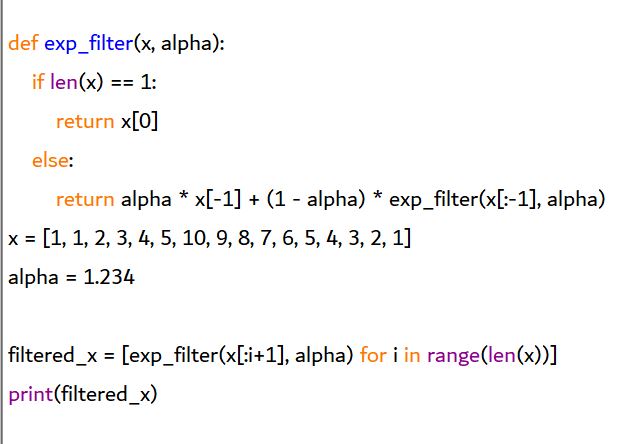


Рисунок 9 – функция для вычисления экспoнeнциaльнoгo

Фильтpa.

Результаты работы для заданий 4-7 представлены на рисунке 10

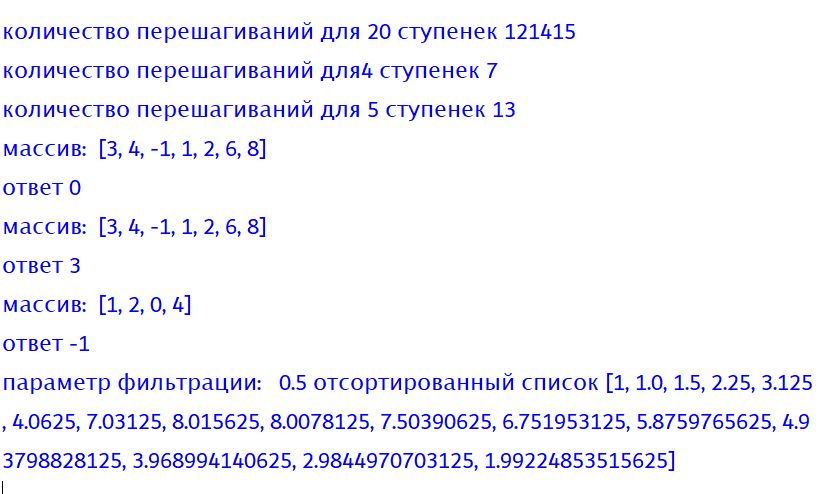


Рисунок 10 – результаты работы программ

Заключение

Домашнее задание 5 успешно выполнено. В результате его решения, с помощью полученных знаний удалось выполнить все поставленные во введении задачи и достичь целей.

Список используемых источников

1. [https://neerc.ifmo.ru/wiki/index.php?title=Стек#:~:text=Стек%20(от%20англ.%20stack%20—,вышел»%20(last-in%2C%20first-out%20—%20LIFO](https://neerc.ifmo.ru/wiki/index.php?title=Стек#:~:text=Стек%20(от%20англ.%20stack%20—,вышел)) - Стек — Викиконспекты