САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №4

по курсу «Алгоритмы и структуры данных»

Тема: Деревья. Пирамида, пирамидальная сортировка. Очередь с приоритетами.

Вариант 20

Выполнил:

Смирнов Георгий Валерьевич

К3139

Проверил:

Афанасьев А.В.

Санкт-Петербург

2024 г.

# Содержание отчета

[**Содержание отчета 3**](#_gjdgxs)

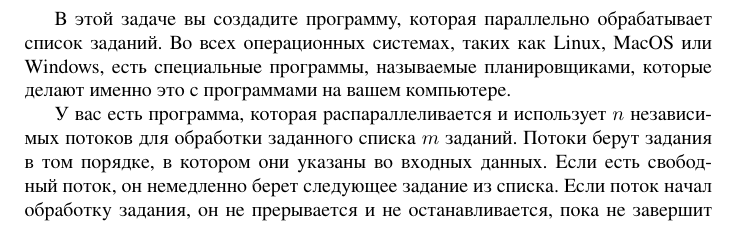
[**Задачи по варианту 4**](#_30j0zll)

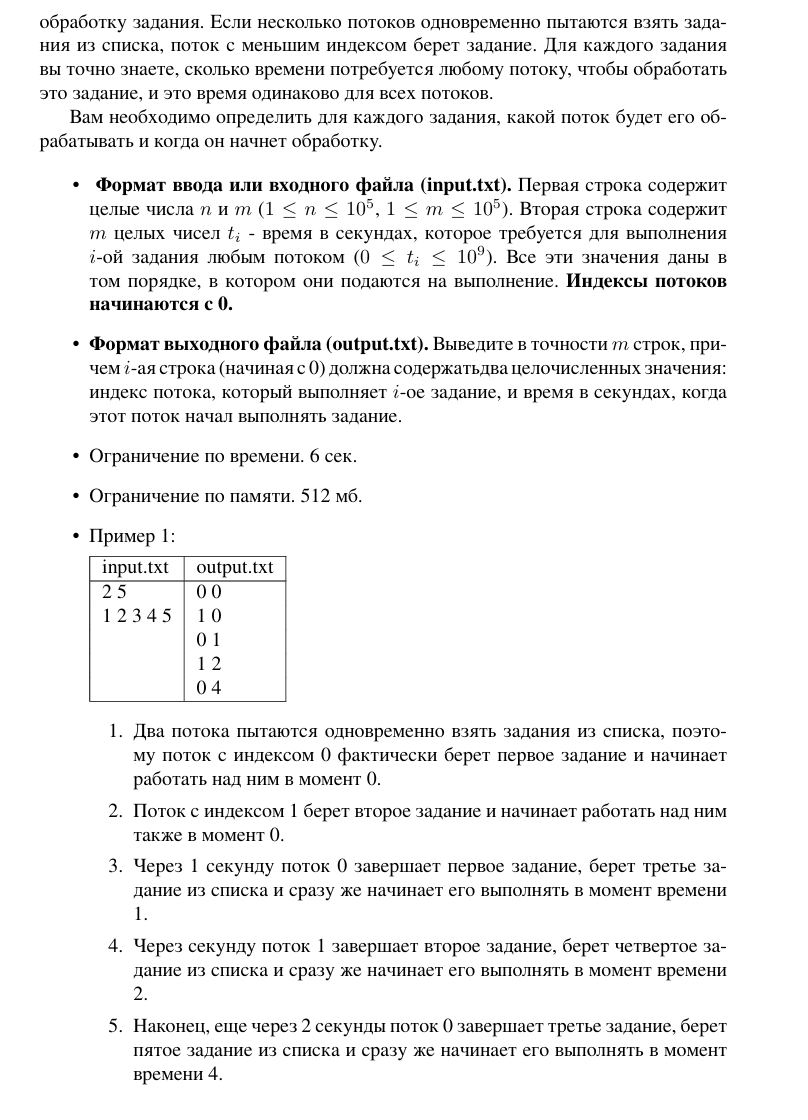
[Задача №5. Планировщик заданий. 4](#_1fob9te)

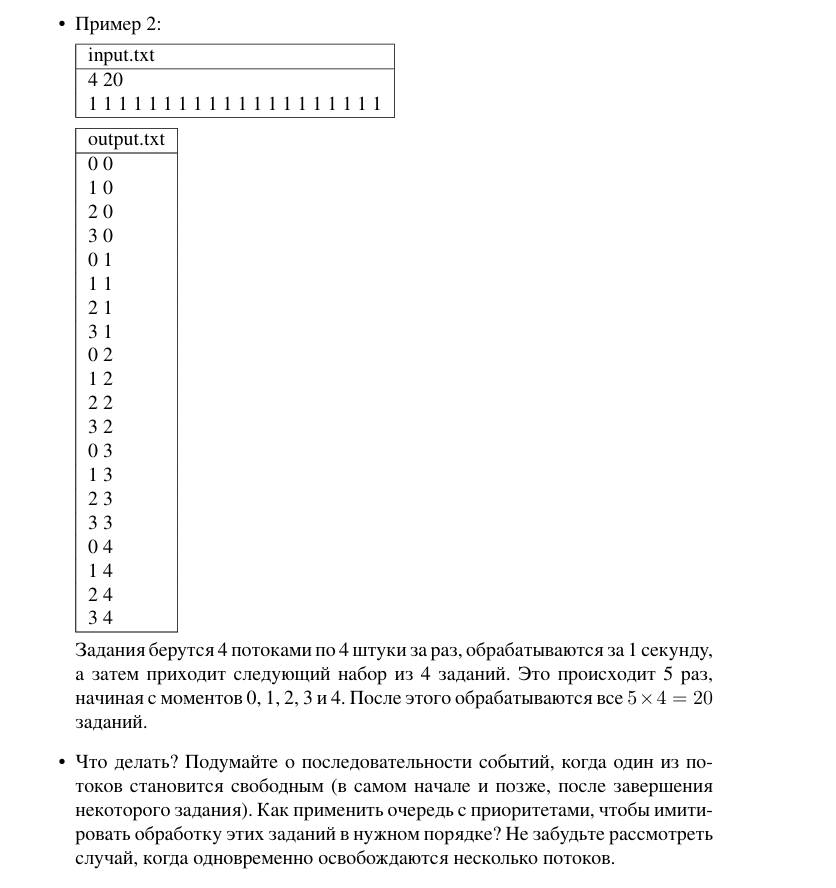
[Задача №7. Снова сортировка. 6](#_3znysh7)

# Задачи по варианту

## Задача №5. Планировщик заданий.







**Код программы**

import heapq

def process\_tasks(n, tasks):

# Очередь с приоритетом для потоков (время завершения, индекс потока)

threads = [(0, i) for i in range(n)]

heapq.heapify(threads)

results = []

for task\_time in tasks:

# Извлекаем поток с наименьшим временем завершения

finish\_time, thread\_index = heapq.heappop(threads)

# Записываем результат: индекс потока и время начала задачи

results.append((thread\_index, finish\_time))

# Обновляем время завершения для потока и возвращаем в очередь

heapq.heappush(threads, (finish\_time + task\_time, thread\_index))

return results

# Чтение данных из файла input.txt

with open('input.txt', 'r') as file:

n, m = map(int, file.readline().strip().split())

tasks = list(map(int, file.readline().strip().split()))

# Обработка задач

output = process\_tasks(n, tasks)

# Запись результатов в файл output.txt

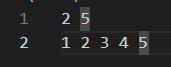
with open('output.txt', 'w') as file:

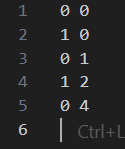
for thread\_index, start\_time in output:

file.write(f"{thread\_index} {start\_time}\n")

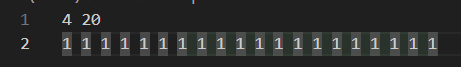
Результат работы кода на примерах из текста задачи:

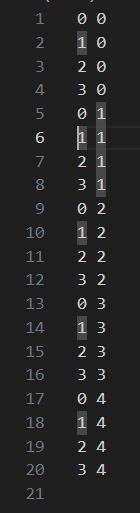
1)



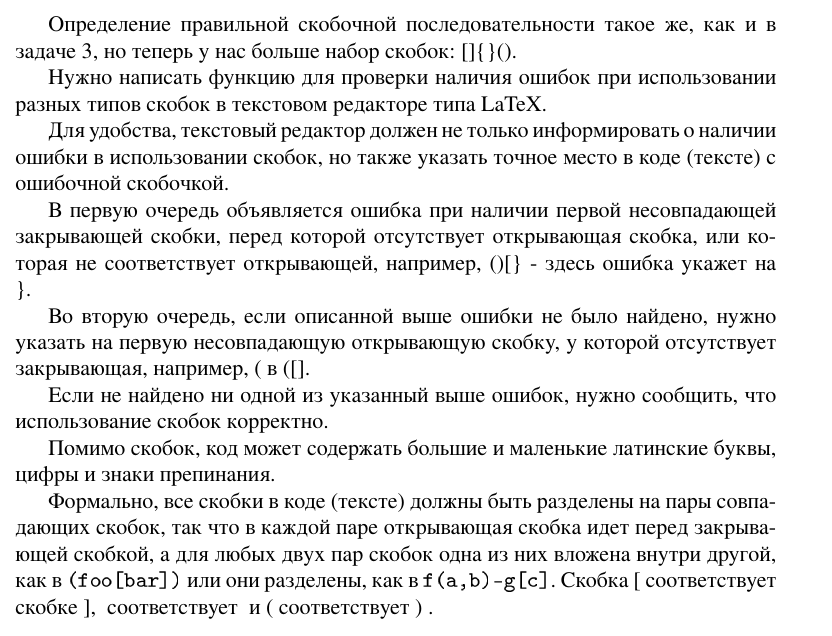


2)





## Задача №7. Снова сортировка.



**Код программы**

def check\_brackets(s):

stack = []

bracket\_map = {')': '(', ']': '[', '}': '{'}

opening\_brackets = set(bracket\_map.values())

for i, char in enumerate(s):

if char in opening\_brackets:

stack.append((char, i + 1))

elif char in bracket\_map:

if stack and stack[-1][0] == bracket\_map[char]:

stack.pop()

else:

return i + 1

if stack:

return stack[0][1]

return "Success"

# Чтение данных из файла input.txt

with open('input.txt', 'r') as file:

s = file.readline().strip()

# Проверка скобочной последовательности

result = check\_brackets(s)

# Запись результата в файл output.txt

with open('output.txt', 'w') as file:

file.write(f"{result}\n")

Результат работы кода на примерах из текста задачи:

1)

