ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчёт по лабораторной работе № 7

«Жадные алгоритмы»

Выполнила работу

Луценко Елена

Академическая группа J3112

Принято

Дунаев Максим Владимирович

Санкт-Петербург

2024

**Структура отчёта:**

1. Введение

Цель работы: разработать эффективное решение для задачи “Split Array Largest Sum” с использованием жадного подхода.

Задачи:

1. Проанализировать условие задачи.
2. Выбрать оптимальный алгоритмический подход для решения задачи.
3. Реализовать выбранный алгоритм на языке C++.
4. Оптимизировать решение с учетом ограничений по памяти и времени выполнения.
5. Провести анализ временной и пространственной сложности полученного алгоритма.
6. Проанализировать, почему в решении необходим жадный алгоритм.
7. Теоретическая подготовка
8. Класс:

Объединяет данные (поля, атрибуты) и функции (методы), работающие с этими данными, в одну структуру

Используемые уровни доступа:

* **public** — члены класса доступны извне (для других частей программы).
* **private** — члены класса доступны только внутри самого класса.

1. Функция:

Блок кода, который выполняет определенную задачу, принимает входные параметры (если они есть), выполняет вычисления и, возможно, возвращает результат; функции помогают организовать код, повышая его читаемость и повторное использование.

1. Векторы:

Использование std::vector для динамического хранения целых чисел

Типы данных:

int: применяется для представления целых значений элементов, используется для индексов массивов и счётчиков элементов;

bool: применяется для обозначения возможности разбития массива на k подмассивов с суммой, не превышающей максимально возможную.

1. Алгоритм:

Использование жадного подхода с применением бинпоиска для нахождения минимально возможной максимальной суммы для разделения массива nums на k подмассивов (на каждом шаге бинарного поиска принимается локально оптимальное решение (выбор текущего mid как потенциальной наименьшей максимальной суммы подмассива) с целью достижения глобально оптимального результата (минимизации максимальной суммы подмассива при разбиении на k подмассивов)).

1. Реализация (в заданной структуре класса Solution)

1) Реализация вспомогательного метода findMax для нахождения максимального элемента в массиве

int findMax(vector<int> nums) {  
 int maxVal = nums[0];  
 for (int num : nums) {  
 if (num > maxVal) {  
 maxVal = num;  
 }  
 }  
 return maxVal;  
}

1. Реализация вспомогательного метода findSum для нахождения суммы элементов в массиве

int findSum(vector<int> nums) {  
 int totalSum = 0;  
 for (int num : nums) {  
 totalSum += num;  
 }  
 return totalSum;  
}

1. Реализация метода canSplit (метод для проверки возможности разбиения массива на k подмассивов с суммой элементов, не превышающей максимально возможную)

bool canSplit(vector<int> nums, int k, int maxSum) {  
 int count = 1;  
 int currentSum = 0;   
 for (int num : nums) {  
 currentSum += num;   
  
 if (currentSum > maxSum) {

count++;   
 currentSum = num;

}  
  
 if (count > k) {

return false;  
 }  
 }  
  
 return true;   
}

1. Реализация метода splitArray для нахождения минимально возможной максимальной суммы элементов подмассива при разбиении

int splitArray(vector<int>& nums, int k) {  
 int left = findMax(nums);   
 int right = findSum(nums);

while (left < right) {  
 int mid = (left + right) / 2;   
  
 if (canSplit(nums, k, mid)) {  
 right = mid;   
 } else {  
 left = mid + 1;   
 }  
 }  
  
 return left;   
}

1. Экспериментальная часть

Временная сложность: O (nlog(sum)), где n - кол-во чисел в nums, sum - сумма всех элементов в массиве

Это связано с использованием бинарного поиска для нахождения минимально возможной максимальной суммы, которая позволит разбить массив nums на k подмассивов.

Пространственная сложность: O (1), так как дополнительных структур данных не используется.

(подробный подсчёт приведён в коде в приложении)

1. Заключение

В ходе выполнения работы был реализован алгоритм для решения задачи нахождения минимально возможной максимальной суммы для разделения массива nums на k подмассивов. Цель работы была достигнута путем применения жадного подхода.

В ходе решения данной задачи используется жадный подход с применением бинпоиска для нахождения минимально возможной максимальной суммы для разделения массива nums на k подмассивов (на каждом шаге бинарного поиска принимается локально оптимальное решение (выбор текущего mid как потенциальной наименьшей максимальной суммы подмассива) с целью достижения глобально оптимального результата (минимизации максимальной суммы подмассива при разбиении на k подмассивов)).

1. Приложения

ПРИЛОЖЕНИЕ А

Листинг кода min\_max\_sum.cpp

class Solution {  
public:  
 int splitArray(vector<int>& nums, int k) { // временная сложность - O(nlog(sum), пространственная сложность - O(1), где n - кол-во элементов в nums, sum — это сумма всех элементов в массиве  
 int left = findMax(nums); // временная сложность - O(n), пространственная сложность - O(1), где n - кол-во элементов в nums  
 int right = findSum(nums); // временная сложность - O(n), пространственная сложность - O(1), где n - кол-во элементов в nums  
  
 while (left < right) { // временная сложность - O(nlog(sum), пространственная сложность - O(1), где n - кол-во элементов в nums, sum — это сумма всех элементов в массиве  
 int mid = (left + right) / 2;  
  
 if (canSplit(nums, k, mid)) { // временная сложность - O(n), пространственная сложность - O(1), где n - кол-во элементов в nums  
 right = mid;  
 } else {  
 left = mid + 1;  
 }  
 }  
  
 return left;  
 }  
  
private:  
 int findMax(vector<int> nums) { // временная сложность - O(n), пространственная сложность - O(1), где n - кол-во элементов в nums  
 int maxVal = nums[0];  
 for (int num : nums) {  
 if (num > maxVal) {  
 maxVal = num;  
 }  
 }  
 return maxVal;  
 }  
  
 int findSum(vector<int> nums) { // временная сложность - O(n), пространственная сложность - O(1), где n - кол-во элементов в nums  
 int totalSum = 0;  
 for (int num : nums) {  
 totalSum += num;  
 }  
 return totalSum;  
 }  
  
 bool canSplit(vector<int> nums, int k, int maxSum) { // временная сложность - O(n), пространственная сложность - O(1), где n - кол-во элементов в nums  
 int count = 1;  
 int currentSum = 0;  
  
 for (int num : nums) { // временная сложность - O(n), пространственная сложность - O(1), где n - кол-во элементов в nums  
 currentSum += num;  
  
 if (currentSum > maxSum) {  
 count++;  
 currentSum = num;  
 }  
  
 if (count > k) {  
 return false;  
 }  
 }  
  
 return true;  
 }  
};

ПРИЛОЖЕНИЕ B

Изображение выглядит как снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рис 1. Задача прошла все тесты