Санкт-Петербургский Политехнический университет Петра Великого
Институт машиностроения, материалов и транспорта
Кафедра «Мехатроника и роботостроение (при ЦНИИ РТК)»

# Курсовая работа

Дисциплина: объектно-ориентированное программирование

Тема: алгоритм Кадана

Работу выполнил: Раков А.Б.
студент группы 3331506/80401

Преподаватель: Кузнецова Е. М.

«__»_____ 2021 г.

Санкт-Петербург
2021 г

## Содержание

# 1. Введение

Изучаемый в данной курсовой работе алгоритм был предложен Джеем Каданом (Jay Kadane) в 1984 году для поиска максимального подмассива одномерного массива (maximum subarray problem). Представленный алгоритм используется в компьютерном зрении (при определении наиболее ярких частей изображений), при анализе геномной последовательности (применяется для определения важных биологических частей белковых последовательностей).

## 2. Описание алгоритма

Для решения поставленной задачи в данном алгоритме используется только один цикл. Идея заключается в том, чтобы разбить большую задачу на несколько подзадач, то есть речь идет о динамическом программировании.

Как видно из кода, представленного в приложении 1, на каждом шаге цикла вычисляется максимальная сумма подмассива до определенной позиции. Например, при индексе 0 массива максимальная сумма - значение элемента, который находится в самом начале массива (нулевого элемента массива). При индексе 1 проверяется элемент под этим индексом и сумма, посчитанная до этого элемента, что означает, что сравниваются значения первого элемента и суммы нулевого и первого, и так далее. После окончания цикла выходное значение записывается как максимальная сумма подмассива.

Работа алгоритма Кадана представлена на рисунке ниже.

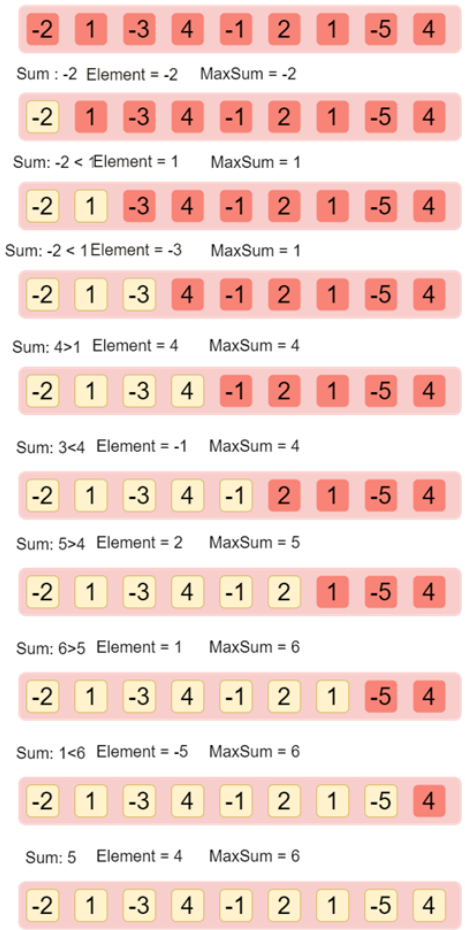


Рисунок 1 – Пример работы алгоритма Кадана

## 3. Оценка скорости

Благодаря тому, что алгоритм Кадана использует подмассивы (максимальная сумма подмассива, который заканчивается на каждой позиции, вычисляется простым способом из связанного, но меньшего подмассива, который закончился на предыдущей позиции), этот алгоритм можно рассматривать как пример динамического программирования. Временная сложность данного алгоритма – $\boldsymbol{O}(n)$. Поскольку он проходит по всему массиву лишь единожды, то его скорость работы зависит исключительно от размера входного массива.

## 4. Анализ алгоритма

Для измерения времени работы алгоритма была выбрана библиотека <*chrono*>. Измерения приводились десять раз при различных размерах входного массива. Каждый раз количество его элементов увеличивалось. Таблица с данными о времени работы алгоритма и количестве элементов массива представлена ниже.

Таблица 1 – Данные тестов

| Количество элементов массива | Время работы алгоритма, мкс |
|---|---|
| 5 | 0 |
| 10 | 0 |
| 50 | 1 |
| 100 | 3 |
| 200 | 6 |
| 500 | 15 |
| 1000 | 27 |
| 5000 | 101 |
| 10000 | 240 |
| 100000 | 1994 |

На рисунке 2 приведен график зависимости времени работы алгоритма от количества элементов входного массива.

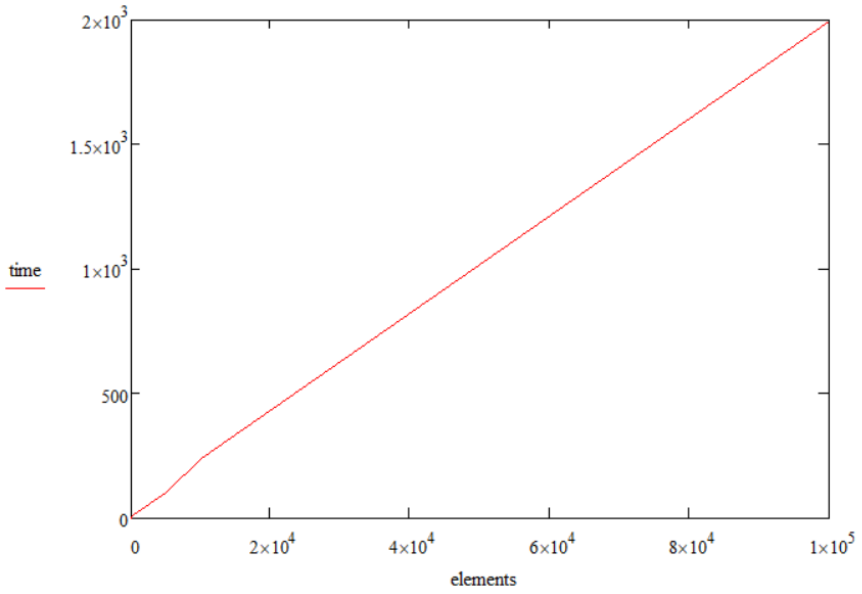


Рисунок 2 – Зависимость время от количества входных данных

## 5. Заключение

Решение задачи по поиску наибольшей суммы в подмассиве с помощью алгоритма Кадана крайне элегантно: собирается только требуемая информация (например, не считаются суммы всех возможных подмассивов). Благодаря использованию методов динамического программирования достигается лучшая временная сложность, а именно – линейное время.

## Список литературы


1. MAXimal :: algo :: Поиск подотрезка массива с максимальной/минимальной суммой http://e-maxx.ru/algo/maximum_average_segment
2. J. Bentley. Programming Pearls. Addison-Wesley, 1999 (2nd edition)
3. Maximum subarray problem – Wikipedia https://en.wikipedia.org/wiki/Maximum_subarray_problem#cite_note-FOOTNOTEBentley198974-4

# Приложение 1 – Код программы

```cpp
#include <iostream>
#include <chrono>
using namespace std;

#define ARRAY_SIZE 10
#define RANGE_OF_VALUES 10000

int kadanes(int array[],int length) {
    int highestMax = 0;
    int currentElementMax = 0;

    for(int i = 0; i < length; i++) {
        currentElementMax =max(array[i],currentElementMax + array[i]) ;
        highestMax = max(highestMax,currentElementMax);
    }

    return highestMax;
}
int main() {
    int elementsQuantity[ARRAY_SIZE] = {5, 10, 50, 100, 200, 500, 1000, 5000, 10000, 100000};

    for (int k = 0; k < 10; k++) {
        int arr[elementsQuantity[k]] = {0};

        for (int i = 0; i < elementsQuantity[k]; i++) {
            arr[i] = rand() % RANGE_OF_VALUES - RANGE_OF_VALUES/2;
        }

        auto start = chrono::steady_clock::now();
        cout << "The Maximum Sum is: "<<kadanes(arr,elementsQuantity[k]) << endl;
        auto end = chrono::steady_clock::now();
        auto duration = chrono::duration_cast<chrono::microseconds>(end - start);
        cout << "Duration is " << duration.count() << " ms"<< endl;
    }

    return 0;
}
```