

Санкт-Петербургский политехнический университет Петра Великого

Институт машиностроения, материалов и транспорта

Высшая школа автоматизации и робототехники

Курсовая работа

Дисциплина: Программирование на языках высокого уровня

Тема: Алгоритм Нарайаны

Выполнил

студент гр. 3331506/70401

Жернаков А. А.

Преподаватель

Ананьевский М. С.

« » _____ 2020 г.

Санкт-Петербург

2020 г.

Формулировка задачи, которую решает алгоритм

Алгоритм Нарайаны - нерекурсивный алгоритм, генерирующий по данной перестановке следующую за ней перестановку (в лексикографическом порядке).

Словесное описание алгоритма

Пусть дана n -элементная перестановка $a = \{a_1, a_2, a_3, \dots, a_n\}$.

Шаг 1: Найти такой наибольший j , для которого $a_j < a_{j+1}$. Если такого элемента не существует, то a – наибольшая n -элементная перестановка. Работа алгоритма завершена.

Шаг 2: Увеличить a_j . Для этого надо найти наибольшее $l > j$, для которого $a_l > a_j$. Затем поменять местами a_j и a_l .

Шаг 3: Записать последовательность a_{j+1}, \dots, a_n в обратном порядке.

Реализация алгоритма

Алгоритм был реализован на языке C++.

Функция, реализующая алгоритм Нарайаны, представлена на рисунке 1.

```
void narayana(int array[], const int n)
{
    // реализация первого шага
    int i = n - 2;
    while (i >= 0 && array[i] > array[i + 1])
        i--;
    // реализация второго шага
    int j = n - 1;
    while (array[j] < array[i])
        j--;
    swap(array[i], array[j]);
    // реализация третьего шага
    for (int t = i + 1, k = n - 1; t < k; t++, k--)
        swap(array[t], array[k]);
    // вывод новой перестановки в консоль
    for (i = 0; i < n; i++)
        cout << array[i];
}
```

Рисунок 1 – Алгоритм Нарайаны

Анализ алгоритма

Наилучшим является случай, когда элементы расположены по возрастанию. В этом случае произойдет 2 сравнения и 1 обмен.

Наихудшей является ситуация, при которой первый элемент меньше второго, а все последующие (с третьего - по последний) меньше первого и расположены в порядке убывания. Если всего в перестановке n элементов, то произойдет $2(n - 1)$ сравнений и $n/2$ обменов при четном n и $(n + 1)/2$ обменов при нечетном n .

В результате сложность алгоритма можно оценить как $O(n)$.

Применение алгоритма

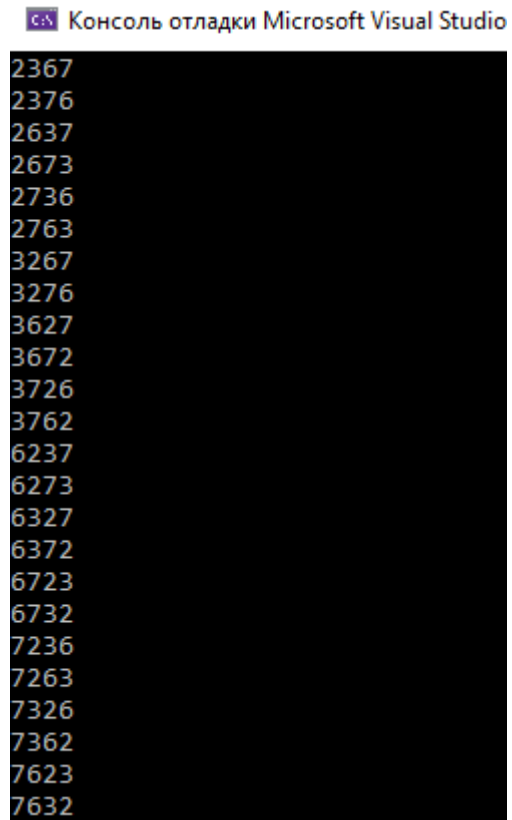
Основным применением алгоритма Нарайаны является вывод всех возможных перестановок из n элементов.

На рисунке 2 представлена реализация функции, которая реализует это применение.

```
// применение алгоритма Нарайаны
void application(int array[], int n)
{
    bubble_Sort(array, n); // сортировка массива по возрастанию
    for (int g = 0; g < n; g++)
        cout << array[g];
    int f = factorial(n); // нахождение количества возможных перестановок
    for (int r = 1; r < f; r++)
    {
        cout << "\n";
        narayana(array, n);
    }
}
```

Рисунок 2 – Применение алгоритма Нарайаны

Результат выполнения данной функции представлен на рисунке 3.



Консоль отладки Microsoft Visual Studio

```
2367
2376
2637
2673
2736
2763
3267
3276
3627
3672
3726
3762
6237
6273
6327
6372
6723
6732
7236
7263
7326
7362
7623
7632
```

Рисунок 3 – Результат