Министерство образования Республики Беларусь

Учреждения образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №9

По дисциплине МОИС за 1 семестр

«ВЫБОРКИ (ПЕРЕСТАНОВКИ, СОЧЕТАНИЯ, РАЗМЕЩЕНИЯ)»

Выполнил

Студент группы ИИ-21

Кирилович А.А.

Проверил

Козинский А.А.

Брест 2023

Задача 1, 3

Реализовать алгоритм построения ВСЕХ перестановок N натуральных чисел. Выполнить все пункты указаний к тестированию. Оценить максимальное N, для которого ваш личный ноутбук выполняет такую реализацию за время меньшее (120 секунд).

Сколько существует перестановок из n элементов, в которых только один элемент находится на своем месте. Перестановки вывести.

#include <iostream>

#include <chrono>

#include <vector>

#include <algorithm>

void print\_array(int arr[], int n) {

    for (int i = 0; i < n; i++)

        std::cout << arr[i] << " ";

    std::cout << std::endl;

}

void generate\_permutations(int arr[], int n) {

    std::sort(arr, arr + n);

    while (true) {

        print\_array(arr, n);

        int i = n - 2;

        while (i >= 0 && arr[i] >= arr[i+1])

            i--;

        if (i < 0)

            break;

        int j = n - 1;

        while (arr[i] >= arr[j])

            j--;

        std::swap(arr[i], arr[j]);

        std::reverse(arr + i + 1, arr + n);

    }

}

std::vector<std::vector<int>> generate\_permutations\_from\_N\_to\_M(int n, int m){

    std::vector<std::vector<int>> res;

    std::vector<int> perm(n);

    for (int i = 0; i < n; i++)

        perm[i] = i;

    int count\_perm = 0;

    do {

        int count = 0;

        for (int i = 0; i < n; i++)

            if (perm[i] == i)

                count++;

        if (count == m) {

            res.push\_back(perm);

            count\_perm+=1;

        }

    } while (next\_permutation(perm.begin(), perm.end()));

    return res;

}

int main(int argc, char\*\* argv) {

    int choice = atoi(argv[1]);

    int num = atoi(argv[2]);

    std::vector<std::pair<int, double>> time;

    if (choice == 1) {

        for(int j = 2; j <= num; j++){

            auto T1 = std::chrono::high\_resolution\_clock::now();

            int arr[j];

            for(int i = 1; i <=j; i++)

                arr[i - 1] = i;

            generate\_permutations(arr,j);

            auto T2 = std::chrono::high\_resolution\_clock::now();

            auto duration1 = std::chrono::duration\_cast<std::chrono::microseconds>(T2 - T1).count();

            std::cout << "Lead time: " << (double)duration1/1000000 << " seconds" << std::endl;

            time.push\_back({j, (double)duration1 / 1000000});

        }

        for(int i = 0; i < num - 1; i++)

            std::cout << "amount of elements: "<< time[i].first <<

            " program running time: "<< time[i].second << std::endl;

    }

    else if (choice == 2) {

        for(int j = 3; j <= num; j++) {

            auto T1 = std::chrono::high\_resolution\_clock::now();

            int  m = 1;

            std::vector<std::vector<int>> perms = generate\_permutations\_from\_N\_to\_M(j, m);

            for (auto perm : perms) {

                for (auto elem : perm)

                    std::cout << elem + 1 << " ";

                std::cout << std::endl;

            }

            auto T2 = std::chrono::high\_resolution\_clock::now();

            auto duration1 = std::chrono::duration\_cast<std::chrono::microseconds>(T2 - T1).count();

            std::cout << "Lead time: " << (double)duration1/1000000 << " seconds" << std::endl;

            time.push\_back({j, (double)duration1 / 1000000});

        }

        for(int i = 0; i < num - 2; i++)

            std::cout << "amount of elements: "<< time[i].first <<

            " program running time: "<< time[i].second << std::endl;

    }

    else std::cout << "Invalid Argument." << std::endl;

}

Задача 2

Подсчитать количество способов, которыми можно поставить на доску (размером n x n) две шашки — белую и черную, так, чтобы они не могли бить друг друга.

#include <iostream>

using namespace std;

int main(int argc, char\*\* argv) {

    int n = atoi(argv[1]);

    char board[n][n];

    for (int row = 0; row < n; row++)

        for (int col = 0; col < n; col++)

            if ((row + col) % 2 == 0)

                board[row][col] = 'W';

            else board[row][col] = 'B';

    int i = 0;

    int move = 0;

    int num1 = 0, num2 = 0, num3 = 0, num4 = 0;

    bool check1, check2, check3, check4 = false;

    char Black\_checker, White\_checker;

    while(1) {

        if (num1 + num2 + num3 + num4 == (n - 1) \* 4)

            break;

        if(num3 + num4 == (n - 1) \* 2) {

            num3 = 0;

            num4 = 0;

            if (num1 < (n - 1))

                num1 += 1;

            else {

                num2 += 1;

                num1 = 0;

            }

        }

        if (num3 <= (n-1))

            if (num3 == 0 & num4 == 0) {

                i += 1;

                if (i == 2) {

                    num3 += 1;

                    i = 0;

                }

            }

            else if (num3 < (n-1))

                num3 += 1;

            else {

                num4 += 1;

                num3 = 0;

            }

        Black\_checker = board[num1][num2];

        White\_checker = board[num3][num4];

        if (Black\_checker != 'B' & White\_checker != 'B' &  (num1 != num3 || num2 != num4)) {

            if((num1 + 1 != num3) || (num2 + 1 != num4))

                check1 = true;

            if((num1 + 1 != num3) || (num2 - 1 != num4))

                check2 = true;

            if((num1 - 1 != num3) || (num2 + 1 != num4))

                check3 = true;

            if((num1 - 1 != num3) || (num2 - 1 != num4))

                check4 = true;

            if(check1 == true & check2 == true & check3 == true & check4 == true)

                move += 1;

            check1 = false;

            check2 = false;

            check3 = false;

            check4 = false;

        };

    }

    cout << "Moves: " << move << endl;

    return 0;

}