Bartłomiej Barszczak

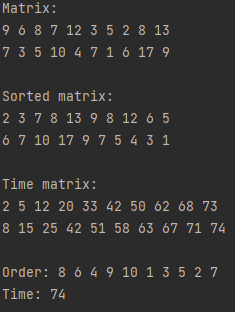
WEAIiIB Automatyka i Robotyka

Rok II semestr IV grupa 3

# Zadanie 1

#include <iostream>  
#include <vector>  
#include <limits>  
#include <map>  
  
  
int INF = std::numeric\_limits<int>::max();  
  
/\*\*  
 \* Funckja printuje podana macierz w konsoli  
 \* @param matrix macierz  
 \* @param title tytul macierzy  
 \*/  
void print\_matrix(const std::vector<std::vector<int>> &matrix, const std::string& title="Matrix:") {  
 std::cout << title << std::endl;  
 for (auto &row : matrix) {  
 for (auto item : row) {  
 std::cout << item << " ";  
 }  
 std::cout << std::endl;  
 }  
 std::cout << std::endl;  
}  
  
bool compere\_elements(std::vector<std::vector<int>> &matrix, std::vector<int> &marked, int x, int y, int b) {  
 // sprawdzanie czy kolumna jest juz wykreslona  
 for (auto item: marked)  
 if (item == y)  
 return false;  
  
 // sprawdzanie czy element jest mniejszy od minimalnego  
 if (matrix[x][y] < b)  
 return true;  
 else  
 return false;  
}  
  
std::vector<std::vector<int>> build\_matrix(std::vector<std::vector<int>> &matrix, std::vector<int> &order) {  
 std::vector<std::vector<int>> result = matrix;  
 int i = 0;  
  
 // budowanie posortowanej macierzy zadan  
 for (auto item: order) {  
 result[0][i] = matrix[0][item];  
 result[1][i] = matrix[1][item];  
 i++;  
 }  
 return result;  
}  
  
int count\_time(const std::vector<std::vector<int>> &matrix) {  
 std::vector<std::vector<int>> result = matrix;  
 result[1][0] = matrix[0][0] + matrix[1][0];  
  
 // tworzenie macierzy czasow  
 for (int i = 1; i < matrix[0].size(); i++) {  
 result[0][i] = result[0][i - 1] + result[0][i];  
 result[1][i] = std::max(result[1][i - 1], result[0][i]) + result[1][i];  
 }  
 print\_matrix(result, "Time matrix:"); // printowanie macierzy czasow  
 return result[1][result[0].size() - 1]; // zwracanie maksymalnego czasu  
}  
  
std::pair<std::vector<int>, int> johnson2m(std::vector<std::vector<int>> &matrix) {  
 int min\_element; // najmnieszy element macierzy  
 int indexes[2]; // indeks wykreslania kolumny  
 int beg = 0; // poczatekowa pozycja  
 int end = 0; // koncowa pozycja  
 std::vector<int> order = {}; // kolejnosc zadan  
 std::vector<std::vector<int>> sorted\_matrix = {}; // posortowane zadania  
  
 // glowna petla  
 while ((beg + end) < int(matrix[0].size())) {  
 min\_element = INF; // wybieranie najmniejszego elementu  
 for (int i = 0; i < 2; i++) {  
 for (int j = 0; j < matrix[0].size(); j++) {  
 if (compere\_elements(matrix, order, i, j, min\_element)) {  
 indexes[0] = i; // dodanie indeksu wiersza  
 indexes[1] = j; // dodanie indeksu kolumny  
 min\_element = matrix[i][j]; // zaktualizowanie najmniejszego elementu  
 }  
 }  
 }  
  
 // dodanie zadania w zalezonosci gdzie zostal znaleziony najmnieszy element  
 if (indexes[0] == 0) {  
 order.insert(std::next(order.cbegin(), beg), indexes[1]);  
 beg++;  
 } else {  
 order.insert(std::next(order.cend(), -end), indexes[1]);  
 end++;  
 }  
 }  
  
 sorted\_matrix = build\_matrix(matrix, order); // budowanie posortowanej macierzy z uporzadkoawnych zadan  
 print\_matrix(sorted\_matrix, "Sorted matrix:"); // printowanie posortowanej macierzy  
 return {order, count\_time(sorted\_matrix)}; // zwracanie kolejnosci zadan oraz maksymalnego czasu  
}  
  
  
int main() {  
 // macierz czasow  
 std::vector<std::vector<int>> m1 = **{** {9, 6, 8, 7, 12, 3, 5, 2, 8, 13},  
 {7, 3, 5, 10, 4, 7, 1, 6, 17, 9}  
 **}**;  
 print\_matrix(m1); // printowanie macierzy na konsolii  
  
 auto result = johnson2m(m1); // wykonanie algorytmu johnsona dla 2 maszyn  
  
 // prezentowanie wynikow  
 std::cout << "Order: ";  
 for (auto item: result.first) {  
 std::cout << item + 1 << " ";  
 }  
 std::cout << "\nTime: " << result.second << std::endl;  
  
 return 0;  
}

# Zadanie 2



# Zadanie 3

1. Jaki typ problemu rozwiązujemy (klasyfikacja Grahama)?

W klasyfikacji Grahama rozwiązujemy problem permutation flow, w tym przypadku dla 2 maszyn.

1. Jakie czasy uzyskamy przy alternatywnych sposobach uszeregowania (taki samo min)?
2. Jakie warunki są konieczne w realizacji algorytmu / co jeśli nie będzie spełniony?

W przypadku problemu permutation flow to algorytm nie wymaga specjalnych warunków.

1. Jaka jest złożoność obliczeniowa algorytmu?

Złożoność obliczeniowa: O(n2)