Domácí úkol k cvičení číslo 6

5. dubna 2024

1 Assignment Problem

Mějme n lidí, které je potřeba přiřadit k vykonání n úkolů, jeden člověk na jeden úkol. Náklady, které by vznikly, pokud by i-tý člověk byl přiřazen k j-tému úkolu, označme jako $C[i, j]$ pro každý pár $i, j = 1, 2, ..., n$. Úkolem je najít přiřazení s minimální celkovou cenou.

Zadání je převzaté ze strany 119 z knihy *Introduction to the design & analysis of algorithms*. Problém si zasloužil i vlastní stránku na Wikipedii. Tam lze také najít mnoho zajímavých algoritmů k řešení tohoto problému.

Příklad Malá instance tohoto problému je následující, s tabulkou reprezentujícími náklady na přiřazení $C[i, j]$:

	Job 1	Job 2	Job 3	Job 4
Person 1	9	2	7	8
Person 2	6	4	3	7
Person 3	5	8	1	8
Person 4	7	6	9	4

1.1 Bruteforce řešení

Pro naivní řešení pomocí bruteforce prohledávání použijte metodu procházení všech možných permutací, které odpovídají jednotlivých přiřazením, a výpočet celkové ceny pro každou permutaci. Nejprve je vytvořen vektor přiřazení, který obsahuje počáteční přiřazení, kde každá osoba je přiřazena k odpovídajícímu indexu úkolu. Pak postupně generujte všechny možné permutace tohoto vektoru přiřazení, napočítejte cenu jednotlivých přiřazení a udělejte update minima. Permutace generujte pomocí funkce `std::next_permutation`, popřípadě po vlastní ose.

Pro jaké největší n Váš výsledný program doběhne pod 10s?

Příkládek Uvažujme permutaci přiřazení: (2, 3, 4, 1) v příkladu výše. To znamená, že Person 1 je přiřazena k Job 2, Person 2 k Job 3, Person 3 k Job 4 a Person 4 k Job 1. Vypočtěme celkovou cenu této permutace:

$$\text{Total cost} = C[1, 2] + C[2, 3] + C[3, 4] + C[4, 1] = 2 + 3 + 8 + 7 = 20.$$

Porovnáme s minimem a jdeme na další permutaci, dokud neprojdeme všechny.

2 Návrh kostry řešení

Komentované iniciativě se meze nekladou.

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <limits>

using std::vector, std::next_permutation, std::cout, std::numeric_limits;

int assignmentProblem(const vector<vector<int>>& costMatrix) {
  int n = costMatrix.size();
  int minCost = numeric_limits<int>::max();
  vector<int> assignment(n);


  // TODO: loop over all assignments==permutations and find minimum

  return minCost;
}

int main() {
  vector<vector<int>> testCostMatrix = {
    {9, 2, 7, 8},
    {6, 4, 3, 7},
    {5, 8, 1, 8},
    {7, 6, 9, 4}
  };

  vector<vector<int>> bigCostMatrix = // something random

  return 0;
}

```