INF 213 - Roteiro da Aula Pratica 12

Atencao: esta aula pratica não valera nota!
O submitty não irá realizar testes nela

Objetivo: praticar o uso de C++11, de STL, etc

→ LEMBREM-SE DE USAR PAPEL E CANETA COMO RASCUNHO ANTES DE IMPLEMENTAR <<--

Material de apoio: https://drive.google.com/open?id=1hzeMqTM-YkXtAuD5qYTceo9O3gRdIlDA

Etapa 1

Crie um map<string,double> para armazenar o coeficiente de rendimento academico (CRA) de estudantes (a string representa o nome do estudante). Assuma que os nomes sao unicos.

Crie quatro funcoes para calcular o CRA medio de todos estudantes em um map. A primeira versao da funcao devera iterar pelo map usando iteradores tradicionais. A segunda versao devera utilizar o “range-based” for (use tambem o tipo “auto” para facilitar a implementacao). A terceira deverá utilizar structured binding para iterar nos pares. A quarta funcao devera utilizar a funcao accumulate (da STL) para calcular a soma das notas (a soma devera ser utilizada para calcular a media).

Dica: utilize uma funcao lambda na funcao accumulate (isso não foi ensinado em detalhes nesta disciplina → pesquise sobre isso na internet e peca ajuda ao professor).

Insira alguns estudantes no map e, entao, use as quatro funcoes para calcular as notas medias de tais estudantes.

Etapa 2

Adicione ao programa anterior uma funcao que, dado o map (descrito anteriormente), imprime o nome e o CRA de cada estudante. Tal impressao devera ser realizada na ordem lexicografica dos nomes.

A seguir, adicione uma nova funcao que faz o mesmo que a anterior, mas imprimindo as informacoes em uma ordem baseada nos CRAs (do maior para o menor).

Teste as duas funcoes.

Etapa 3

Considere novamente o map criado nos exercicios anteriores. Usando a funcao count_if e funcoes lambda (peca ajuda ao professor), conte quantos alunos obtiveram notas maiores ou iguais a 60%.

Teste sua implementacao.

Etapa 4 - Problema do caixeiro viajante

Dada uma matriz quadrada M, suponha que o elemento da posição i,j indica a distancia rodoviaria entre a cidade i e a cidade j.

Um problema classico da computacao (o Problema do Caixeiro Viajante) consiste em, dada a matriz M com n linhas/colunas, calcular a distancia minima total necessaria para se visitar todas as n cidades (sem repetir cidades) e voltar para a cidade inicial.

Por exemplo, considere a matriz M abaixo:

0	10	15
20	0	40
7	8	0

Uma opcao de rota consiste em visitar as cidades na ordem: 0, 2, 1,0 . O custo total sera: 15 (ir de 0 para 2) + 8 (ir de 2 para 1) + 20 (ir de 1 para 0).

Esse problema (que sera estudado mais a fundo nas disciplinas de PAA e Grafos) parece ser bastante simples (talvez deveriamos comecar em uma cidade e sempre viajar para a proxima cidade mais proxima?), mas, por incrivel que pareca, não se conhece nenhum algoritmo eficiente (polinomial) para resolve-lo! Os algoritmos eficientes conhecidos sao apenas aproximacoes (não ha garantias de que eles encontrarao sempre a melhor solucao).

Uma forma de resolve-lo (mas de forma ineficiente) e sempre achar a melhor solucao consiste em se testar todas as solucoes possiveis (todas permutacoes de cidades).

Crie uma funcao “custo” que, dada a matriz M (representada por um vector<vector<int> >) e um vector com a sequencia de cidades a serem visitadas, retorna o custo dessa solucao (ou seja, se a matriz for a apresentada acima e o vetor tiver os elementos (0,2,1,0) a saida devera ser 15+8+20 = 43).

A seguir, desenvolva um programa que gera todas as possibilidades de percurso e, entao, imprime na tela o custo do menor percurso. Se você utilziar a STL (utilize!) seu programa sera bastante simples e com poucas linhas de codigo! (dica: use a função next_permutation)

Use a funcao time para medir o tempo de execucao do seu programa considerando matrizes de diferentes tamanhos. O arquivo fornecido como material para a pratica contem matrizes de diferentes tamanhos como exemplo (o inicio de cada matriz contem o tamanho dela).

Dica para deixar seu programa um pouco mais rapido (tente fazer isso após resolver o problema): e’ necessario testar todas possibilidades para a primeira cidade do percurso? Ou a cidade na qual comecamos não importa?

Curiosidade: apesar desse problema ser extremamente dificil de ser resolvido, encontrar a melhor rota entre duas cidades e’ uma tarefa simples de ser realizada de forma eficiente! (há inclusive um algoritmo que faz isso utilizando apenas 4 linhas de codigo!). Isso será estudado em teoria de grafos e em PAA.