DISTRIBUIÇÃO DE TAREFAS

CAIXEIRO VIAJANTE

AQUI ESTÁ UM ESBOÇO DE COMO VOCÊ PODE ABORDAR A TAREFA, INCLUINDO A IMPLEMENTAÇÃO EM C PARA UM EXEMPLO DE APLICAÇÃO DA TEORIA DOS GRAFOS. VOU SUGERIR O USO DO PROBLEMA DO "CAIXEIRO VIAJANTE", UM DOS PROBLEMAS MAIS CLÁSSICOS NA TEORIA DOS GRAFOS.

INTRODUÇÃO

CONCEITO DE GRAFOS:
UM GRAFO É UMA ESTRUTURA DE DADOS QUE CONSISTE EM UM CONJUNTO DE VÉRTICES (OU NÓS) E ARESTAS (OU ARCOS) QUE CONECTAM PARES DE VÉRTICES. ELE É USADO PARA MODELAR RELAÇÕES BINÁRIAS ENTRE OBJETOS.

ÁREA ABORDADA

O PROBLEMA DO CAIXEIRO VIAJANTE (TSP – TRAVELLING SALESMAN PROBLEM) É UMA APLICAÇÃO CLÁSSICA DA TEORIA DOS GRAFOS. ELE É UTILIZADO EM ÁREAS COMO LOGÍSTICA, PLANEJAMENTO DE ROTAS, E ATÉ MESMO EM CIRCUITOS ELETRÔNICOS.

EXEMPLO ESCOLHIDO

UMA SOLUÇÃO SIMPLES PARA O PROBLEMA DO CAIXEIRO VIAJANTE USANDO A ABORDAGEM DE FORÇA BRUTA.

DESENVOLVIMENTO

CONCEITOS DA ÁREA ESCOLHIDA:
O TSP CONSISTE EM ENCONTRAR O MENOR CAMINHO POSSÍVEL QUE PERMITA A UM VENDEDOR PASSAR POR VÁRIAS CIDADES, PARTINDO DE UMA CIDADE INICIAL, VISITANDO TODAS AS OUTRAS EXATAMENTE UMA VEZ, E RETORNANDO À CIDADE DE PARTIDA.

DADOS E INFORMAÇÕES:

O PROBLEMA É NP-DIFÍCIL, O QUE SIGNIFICA QUE NÃO EXISTE UM ALGORITMO CONHECIDO QUE POSSA RESOLVER TODOS OS CASOS DO TSP EM TEMPO POLINOMIAL. NO ENTANTO, SOLUÇÕES APROXIMADAS OU DE FORÇA BRUTA PODEM SER IMPLEMENTADAS PARA PROBLEMAS MENORES.

IMPLEMENTAÇÃO EM C:
AQUI ESTÁ UM EXEMPLO DE IMPLEMENTAÇÃO EM C USANDO FORÇA BRUTA PARA RESOLVER O PROBLEMA DO CAIXEIRO VIAJANTE:

main.c

```c
#include <stdio.h>
#include <limits.h>
#include <stdbool.h>

#define V 4  // Número de vértices no grafo

// Função auxiliar para trocar dois valores
void swap(int* a, int* b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

// Função para reverter um segmento de array
void reverse(int array[], int start, int end) {
    while (start < end) {
        swap(&array[start], &array[end]);
        start++;
        end--;
    }
}

// Função para gerar a próxima permutação lexicográfica
bool next_permutation(int array[], int size) {
    int i = size - 2;
    while (i >= 0 && array[i] >= array[i + 1]) {
        i--;
    }
    if (i < 0) {
        return false;
    }
    int j = size - 1;
    while (array[j] <= array[i]) {
        j--;
    }
    swap(&array[i], &array[j]);
    reverse(array, i + 1, size - 1);
    return true;
}

int travellingSalesmanProblem(int graph[][V], int s) {
    int vertex[V - 1];
    int j = 0;
    for (int i = 0; i < V; i++) {
        if (i != s) {
            vertex[j++] = i;
        }
    }

    int min_path = INT_MAX;
    do {
        int current_pathweight = 0;
        int k = s;
        for (int i = 0; i < V - 1; i++) {
            current_pathweight += graph[k][vertex[i]];
            k = vertex[i];
        }
        current_pathweight += graph[k][s];
        if (current_pathweight < min_path) {
            min_path = current_pathweight;
        }

    } while (next_permutation(vertex, V - 1));

    return min_path;
}

int main() {
    int graph[V][V] = { {0, 10, 15, 20},
                        {10, 0, 35, 25},
                        {15, 35, 0, 30},
                        {20, 25, 30, 0} };
    int s = 0;
    printf("O menor caminho tem o custo %d\n", travellingSalesmanProblem(graph, s));
    return 0;
}
```

IMPLEMENTAÇÃO EM C:
POSSÍVEIS RESULTADOS:

ESTE PROGRAMA CALCULARÁ A MENOR ROTA QUE O CAIXEIRO VIAJANTE PODE TOMAR PARA VISITAR TODAS AS CIDADES E RETORNAR À CIDADE DE ORIGEM.

CONCLUSÃO

CONSIDERAÇÕES:
O PROBLEMA DO CAIXEIRO VIAJANTE ILUSTRA A COMPLEXIDADE DE PROBLEMAS NP-DIFÍCEIS E A IMPORTÂNCIA DE MÉTODOS EFICIENTES PARA RESOLVÊ-LOS. APESAR DE SER INEFICIENTE PARA GRANDES ENTRADAS, O MÉTODO DE FORÇA BRUTA SERVE PARA ILUSTRÁR O CONCEITO.