www.inatel.br

Problemas de otimização

Problemas de otimização

Maximizar ou minimizar algum valor ou variável

- Método guloso;
- Método exaustivo (ou força-bruta);
- Divisão e conquista;
- Programação dinâmica.

Problema do caixeiro viajante

Problema do caixeiro viajante

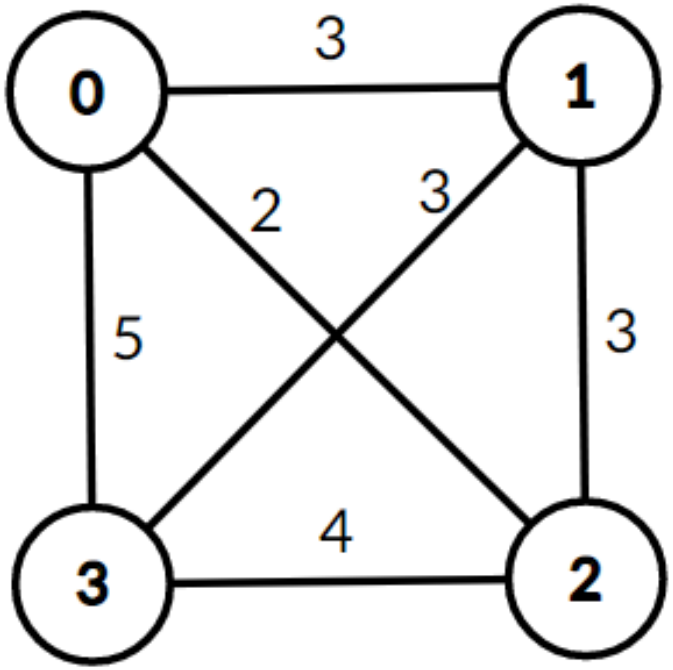
- Consiste em:

1. Sair da cidade inicial;
2. Passar por **todas** as outras cidades **apenas uma vez;**
3. Voltar para a cidade inicial;

Qual o menor percurso para completar essa viagem?

www.inatel.br

Problema do caixeiro viajante



custo[n][n];

	0	1	2	3
0		3	2	5
1	3		3	3
2	2	3		4
3	5	3	4	

www.inatel.br

Problema do caixeiro viajante

- Método guloso;
- Método exaustivo (força bruta);
- Programação dinâmica.

www.inatel.br

Método guloso

- Observa os ótimos locais;
- Rápido (Complexidade $O(n^2)$);
- Não garante resultado ótimo.

www.inatel.br

Força bruta (busca exaustiva)

- Testa TODAS as possibilidades;
- EXTREMAMENTE ULTRA LENTO(O(∞)O(n!));
- Garante resultado ótimo.


i have a question

```cpp
#include <iostream> // Entrada e saï¿½da
#include <climits> // INT_MAX
#include <algorithm> // next_permutation
using namespace std;

int main()
{
    int custo[100][100]; // 100 -> numero maximo de cidades
                         // custo[i][j] = custo de ir da cidade 'i' para a cidade 'j'
    int n; // numero de cidades
    int cidade_inicial;

    cin >> n;
    for(int i = 0;i < n;i++)
        for(int j = i+1;j < n;j++)
        {
            cout << "Entre com o custo de ir da cidade " << i << " para a cidade " << j << endl;
            cin >> custo[i][j];
            custo[j][i] = custo[i][j];
        }
    cin >> cidade_inicial;


    int v[100]; // vetor com as cidades a serem permutadas
    int p; // var. auxiliar
    int menor_custo; // menor custo da viagem
    int custo_caminho; // custo do caminho

    // colocando as cidades que serao permutadas no vetor (todas exceto a inicial)
    p = 0;
    for(int i = 0;i < n;i++)
        if(i != cidade_inicial)
        {
            v[p] = i;
            p++;
        }
```

```cpp
    // forca bruta
    menor_custo = INT_MAX; // inicializo o custo com infinito
    do
    {
        custo_caminho = custo[cidade_inicial][v[0]]; // custo da cidade inicial -> v[0]

        for(int i = 0;i < n-2;i++) // somar custo de v[0]->v[1]->v[2]...v[n-3]->v[n-2]
            custo_caminho += custo[v[i]][v[i+1]];

        custo_caminho += custo[v[n-2]][cidade_inicial]; // custo v[n-2] -> cidade inicial

        menor_custo = min(menor_custo,custo_caminho);
    }while(next_permutation(v,v+(n-1))); // n-1 = numero de cidades a serem permutadas
    // obs.: para fazer todas as permutacoes, o vetor deve estar ordenado no comeco

    cout << "Menor custo = " << menor_custo << endl;
    return 0;
}
```

```cpp
#include <iostream> // Entrada e saï¿½da
#include <climits> // INT_MAX

using namespace std;

int main()
{
    int custo[100][100]; // 100 -> numero maximo de cidades
                         // custo[i][j] = custo de ir da cidade 'i' para a cidade 'j'
    int cidade_inicial; // Primeira e ultima cidade
    int n; // numero de cidades

    cout << "Entre com o numero de cidades" << endl;
    cin >> n;
    cout << "Entre com a cidade inicial" << endl;
    cin >> cidade_inicial;

    for(int i = 0;i < n;i++)
        for(int j = i+1;j < n;j++)
        {
            cout << "Entre com o custo de ir da cidade " << i << " para a cidade " << j << endl;
            cin >> custo[i][j];
            custo[j][i] = custo[i][j]; // Custo de i->j = custo de j->i
        }

    bool vis[100]; // marca se ja visitou ou nao uma cidade
    int menor_custo; // menor custo da viagem
    int custo_at; // menor custo atual
    int cidade_atual; // cidade atual
    int proxima_cidade; // proxima cidade

    // inicializando vetor vis (nenhuma cidade foi visitada no inicio)
    for(int i = 0;i < n;i++)
        vis[i] = false;
```

```cpp
    // guloso
    menor_custo = 0;
    cidade_atual = cidade_inicial;
    for(int i = 0;i < n-1;i++) // n-1 vezes = qnt de caminhos (exceto a volta para a cidade inicial)
    {
        vis[cidade_atual] = true; // marco que ja visitei a cidade atual
        // encontrar qual a proxima cidade mais proxima
        custo_at = INT_MAX;
        for(int j = 0;j < n;j++)
        {
            if(!vis[j] && custo_at > custo[cidade_atual][j])
            {
                proxima_cidade = j;
                custo_at = custo[cidade_atual][j];
            }
        }
        menor_custo += custo_at;
        cidade_atual = proxima_cidade;
    }
    menor_custo += custo[cidade_atual][cidade_inicial];
    cout << "Menor custo = " << menor_custo << endl;

    return 0;
}
```