UNIVERSIDADE
FEDERAL DO CEARÁ
CAMPUS DE CRATEÚS

Compiladores e Linguagens de Programação • 2024.1

Trabalho 3

PROBLEMA DO CAIXEIRO VIAJANTE

PARTE I – INSTRUÇÕES PARA IMPLEMENTAÇÃO

DEFINIÇÃO

Dado um conjunto de cidades e as distâncias entre cada par de cidades, encontrar o caminho mais curto que visita cada cidade exatamente uma vez e retorna à cidade de origem. O objetivo é encontrar o caminho que minimiza a distância total percorrida.

IMPLEMENTAÇÃO EM TRÊS PARADIGMAS DE PROGRAMAÇÃO

O problema do caixeiro viajante, um clássico na ciência da computação, nos desafia a encontrar a rota mais eficiente para um vendedor que precisa visitar várias cidades e retornar ao ponto de partida. Este problema é uma excelente oportunidade para aplicarmos três paradigmas de programação: **imperativo**, **orientado a objetos** e **funcional**. Cada paradigma oferece uma maneira diferente de pensar e abordar problemas, além do desenvolvimento de habilidades práticas.

- **Imperativo**
    - No paradigma imperativo, você pode obter uma compreensão mais profunda do funcionamento interno de algoritmos. Ao trabalhar diretamente com estruturas de dados e manipulação de baixo nível, você desenvolve uma visão mais clara de como otimizar a utilização de recursos como memória e processamento.
    - Você deve implementar o programa em **C**.
- **Orientado a objetos**
    - Ao adotar o paradigma orientado a objetos, é possível modelar problemas do mundo real utilizando classes e objetos, o que lhe permite criar sistemas mais modulares e escaláveis.
    - Você deve implementar o programa em **Java.**
- **Funcional**
    - A programação funcional, com sua ênfase em funções puras, imutabilidade e composição de funções, os encorajará a pensar de maneira declarativa e a criar soluções elegantes e concisas.
    - Você deve implementar o programa em **Haskell.**

FUNCIONALIDADES A IMPLEMENTAR

1. Geração de um conjunto de cidades de forma aleatória.
   - *Cada cidade pode ser representada por uma coordenada em um plano bidimensional, o que significa que cada cidade terá uma posição específica em um gráfico com um eixo X e um eixo Y.*
   - *Você deve permitir que o usuário decida quantas cidades ele deseja gerar dentro de um intervalo definido por você.*
   - *Você também deve escolher um intervalo, como por exemplo, de 0 a 100, para representar as coordenadas x e y de cada cidade. Garantir que as coordenadas geradas sejam únicas para evitar sobreposição de cidades.*
   - *Utilize uma função para gerar números aleatórios dentro do intervalo escolhido.*
2. Cálculo da distância entre duas cidades quaisquer.
   - *Para calcular a distância entre duas cidades em um plano bidimensional, utilizamos a fórmula da distância euclidiana.*
3. Cálculo da distância total de um determinado caminho.
   - *No contexto do problema do caixeiro viajante, um caminho refere-se a uma sequência ordenada de cidades que o caixeiro viajante deve percorrer. Especificamente, o caminho começa em uma cidade inicial, passa por todas as outras cidades exatamente uma vez, e retorna à cidade de origem.*
4. Implementação de algoritmos para encontrar uma solução.
   - *Para resolver o problema do caixeiro viajante, você pode usar diferentes algoritmos. Aqui, vamos adotar dois métodos básicos: força bruta e vizinho mais próximo.*

     ***Força bruta**: testa todas as possíveis rotas (permutações) entre as cidades para encontrar a que tem a menor distância total.*

     ***Vizinho mais próximo**: começa em uma cidade e sempre vai para a cidade mais próxima que ainda não foi visitada, até que todas as cidades sejam visitadas.*
5. Visualização do melhor caminho encontrado.
   - Saída textual que represente a ordem das cidades no caminho que representa a solução, adotando o seguinte formato:
     - `Melhor caminho encontrado: A -> C -> B -> E -> D -> A.`
     - `Distância total: 120.5 (a unidade adotada pelo programa é Km).`
   - Um plano 2D que mostre os pontos representando as cidades e o caminho que representa a solução.

   Algumas linguagens não possuem uma biblioteca gráfica embutida; nesse caso, você pode utilizar ferramentas externas, bibliotecas específicas ou exportar os dados e usar uma ferramenta de visualização como o Matplotlib em Python.

PARTE II– Instruções para Elaboração do Relatório

Introdução

(*Limite de 250 palavras*)

- Apresente o problema do caixeiro viajante, explicando os desafios envolvidos em encontrar a rota mais curta que visita todas as cidades e retorna ao ponto de partida.

Procedimentos Metodológicos

(*Limite de 1200 palavras*)

- **Descrição dos paradigmas de programação**
  - **Imperativo**: explique o paradigma imperativo, focando em como ele lida com a manipulação de estado e controle de fluxo.
  - **Orientado a objetos:** descreva como o paradigma orientado a objetos encapsula dados e comportamentos em classes e objetos.
  - **Funcional:** apresente o paradigma funcional, enfatizando a imutabilidade, funções puras e a ausência de efeitos colaterais.
- **Algoritmos implementados**
  - Descreva o algoritmo de força bruta e o algoritmo de vizinho mais próximo explicando sua lógica e quando eles são mais eficazes.
  - Descreva como as cidades foram geradas em cada linguagem (C, Java, Haskell).

Resultados

(*Limite de 400 palavras*)

- **Análise comparativa de desempenho**
  - Como o tempo de execução foi medido em cada algoritmo/linguagem.
  - Como o consumo de memória foi avaliado em cada algoritmo/linguagem.
  - Insira uma tabela como a apresentada abaixo (Informe quantidade de cidade consideradas).

	Imperativo (C)		Orientação a objetos (Java)		Funcional (Haskell)	
Algoritmo	**Tempo**	**Memória**	**Tempo**	**Memória**	**Tempo**	**Memória**
Força bruta						
Vizinho mais próximo						

Conclusão

(*Limite de 400 palavras*)

- Principais lições aprendidas com a implementação dos diferentes paradigmas.

PARTE III– Instruções para a Apresentação

(*O tempo limite de apresentação é de 10 minutos*)

Para cada algoritmo, apresente os resultados obtidos (em ordem crescente de tempo de execução) em cada um dos paradigmas conforme descrito a seguir:

Informe a quantidade de cidades consideradas.

- **[Nome do algoritmo]**
    - **[Nome do paradigma]**

Caminho	Distância	Tempo
Caminho 1		
Caminho 2		
⋮		
Caminho n		

Finalize relatando com as principais lições aprendidas e os principais desafios enfrentados.

***

Instruções para a Entrega

- O relatório deverá ser entregue pelo SIGAA em formato PDF até dia **12/09/2024** às **14:59**.
- Deverá ser obrigatoriamente feito em dupla.
- A apresentação será de apenas 10 minutos para apresentar os resultados.