Trabalho Prático 3 - Estações de Recarga da BiUAIDi

1 Estruturação do Trabalho Prático

O trabalho prático será dividido em duas partes, a primeira obrigatória e a segunda opcional e geradora de pontos extras. Como é apresentado a seguir, o trabalho como um todo é organizado em 5 partes:

1. Flexibilização do aplicativo
2. Uso de QuadTree como estrutura de dados
3. Avaliação experimental comparativa das versões do aplicativo
4. Extensão do aplicativo para uma arquitetura de dois níveis
5. Avaliação experimental do aplicativo estendido.

A parte obrigatória compreende as 3 primeiras partes enquanto a parte opcional está associada às duas últimas. As entregas são detalhadas na Seção 3.

2 Descrição do problema

Você foi contratado pela fabricante de carros elétricos **BiUaiDi** para modernizar o aplicativo de identificação de estações de recarga para os carros da fabricante. O aplicativo é multi-plataforma e a sua função fundamental é, dada a posição atual do carro, identificar as 10 estações de recarga mais próximas. Belo horizonte será a primeira cidade onde o aplicativo será implantado, tendo em vista as frequentes necessidades de recarga em virtude do relevo da cidade.
Tendo em vista os prazos exíguos de lançamento do carro e o pequeno número de estações de recarga (em torno de 100 na cidade de Belo Horizonte), a versão atual do aplicativo calcula a distância entre o ponto de origem e todos as estações de recarga, o que é bastante ineficiente, mas efetivo para o pequeno número de estações de recarga disponíveis. Adicionalmente, estações de recarga podem ser ativadas ou desativadas temporariamente, seja por estarem sendo utilizadas por algum veículo, seja por outra questão técnica, além de novas estações de recarga que são habilitadas a todo momento. Assim, o aplicativo deve sugerir estações de recarga a partir da configuração instantânea das estações de recarga.
O aplicativo atual é bastante simples e não satisfaz todos os requisitos. Ele é invocado informando uma coordenada x, y e retorna as dez estações de recarga mais próximas. Por exemplo, o comando:

```
recargaBiUaiDi 610167.846027439 7795602.43183451
```

gera a saída apresentada na Figura 2 e ilustrada pelo mapa da Figura 3.
O aplicativo atual é disponibilizado como ponto de partida no minha.ufmg. Ele tem uma série de limitações, como descrito a seguir. Primeiro, ele considera um conjunto fixo de aproximadamente 100 estações de recarga. Segundo, ele não permite ativações e desativações das estações. Terceiro, ele sempre retorna as dez estações de recarga mais próximas. Finalmente, as estações de recarga são mantidas em um vetor, e cada consulta demanda o cálculo das distâncias da localização atual do veículo a todos os pontos de recarga ($O(n)$), que precisam ser ordenados para identificar os dez pontos mais próximos ($O(nlogn)$).
Diante do cenário, você propôs uma estratégia de cinco estágios para construir uma nova solução, descrita a seguir:

2.1 Flexibilização do aplicativo

O primeiro estágio da estratégia é evoluir o aplicativo para que ele trabalhe com um conjunto variável de pontos de recarga e também passe a funcionar como um serviço que receba variados pontos de origem, por exemplo representado pela movimentação de um veículo. Essa situação será simulada pela leitura e processamento de dois arquivos:

```
00446106627;4461;AVE;PRESIDENTE ANTONIO CARLOS;6627;Campus UFMG;PAMPULHA;31270901;609242.979899325;7802765.0723672
00125901537;1259;AVE;AFONSO PENA;1537;Centro;CENTRO-SUL;30130004;611566.586666557;7796363.53379632
```

Figura 1: Exemplo de arquivo de entrada

```
RUA TOMAZ GONZAGA, 388, Lourdes, CENTRO-SUL, 30180140 (492.143)
AVE BARBACENA, 691, Barro Preto, CENTRO-SUL, 30190134 (1126.117)
RUA GUAICUI, 510, Luxemburgo, CENTRO-SUL, 30380342 (1696.438)
RUA CAMPO BELO, 28, São Pedro, CENTRO-SUL, 30330330 (2004.310)
RUA ORENOCO, 58, Carmo, CENTRO-SUL, 30310060 (2004.932)
RUA DOIS MIL SEISCENTOS E TRINTA E DOIS, 105, Carlos Prates, NOROESTE, 30000 (2060.761)
AVE DOM PEDRO II, 388, Bonfim, NOROESTE, 31210242 (2234.568)
RUA URUCUIA, 110, Floresta, CENTRO-SUL, 30150060 (2324.576)
BEC BRILHANTINA, 96, Vila Dias, LESTE, 31010172 (3169.933)
RUA SANTOS, 494, Jardim América, OESTE, 30421386 (3178.736)
```

Figura 2: Exemplo de saída textual do aplicativo

Estações de recarga: A BiUaiDi fez um convênio com a Prefeitura de BH e conseguiu uma lista, na forma de um arquivo, de todas as potenciais estações de recarga que podem ser instaladas na cidade. Esse arquivo, uma vez carregado no aplicativo, vai permitir variar as estações de recarga ativas num dado momento. Os atributos sobre estações de recarga são apresentados na Tabela 1.

A Figura 1 apresenta exemplos dos dados de duas estações de recarga.

Atualizações e Consultas: O segundo arquivo simula o funcionamento do aplicativo em um carro em movimento e o dinamismo em termos de ativação e desativação de pontos de recarga. Assim, cada linha do arquivo contem um de três comandos possíveis:

C x y n Consulta dos n pontos de recarga mais próximos da posição x, y. Um exemplo da consulta é apresentado a seguir:

```
C 610167.846027439 7795602.43183451 10
```

A resposta para a consulta pode ser vista na Figura 2. A Figura 3 mostra um mapa contendo as estações de recarga ativas naquele momento e as estações sugeridas. O mapa foi gerado pelo programa `biuaidinaive`.

Atributo	Tipo	Descrição
idend	char *	Identificador do endereço.
id_logrado	long	Identificador do logradouro.
sigla_tipo	char *	Tipo do logradouro (p.ex., rua e avenida).
nome_logra	char *	Nome do logradouro.
numero_imo	int	Número do imóvel no logradouro.
nome_bairr	char *	Nome do bairro.
nome_regio	char *	Nome da região (p.ex., centro-sul, Pampulha).
cep	int	CEP (código de endereçamento postal).
x	double	Coordenada X.
y	double	Coordenada Y.

Tabela 1: Informações sobre potenciais estações de recarga

A id Ativar o ponto de recarga associado ao identificador id.

O exemplo a seguir mostra o comando da ativação de uma mesma estação de recarga, mostrando as duas mensagens de saída possíveis.

Entrada	Saída
`A 00446106627`	Ponto de recarga 00446106627 ativado.
`A 00446106627`	Ponto de recarga 00446106627 já estava ativo.

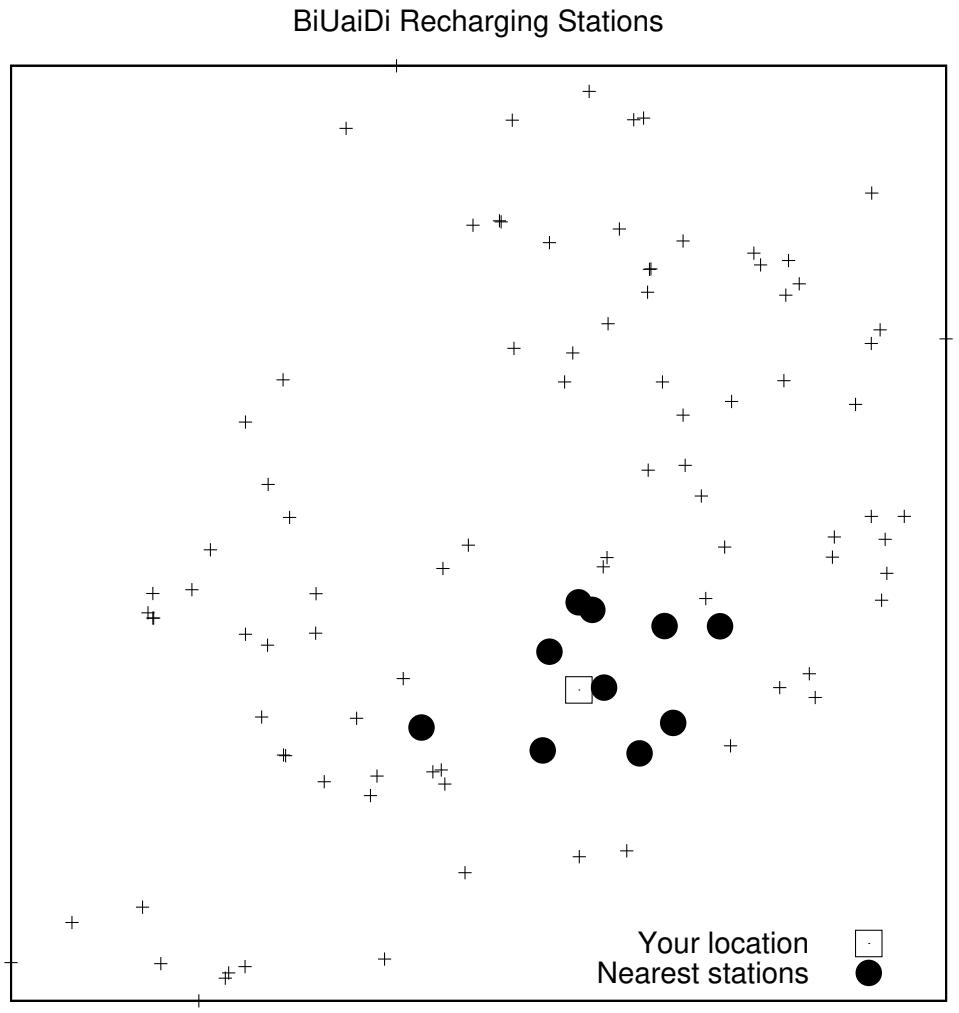

Figura 3: Exemplo de saída gráfica do aplicativo

```
A 01855100150
D 03986100116
A 03587300106A
D 01981600122A
D 07512700842
C 604476.327080534 7793936.51282176 10
C 605262.481794150 7793257.07737953 5
D 02813300418
C 616071.596501685 7799448.81579611 7
C 607550.931138684 7798701.16804509 10
A 01668501159A
A 08098800057B
C 606279.331855856 7793399.39903000 5
C 605104.903517628 7798056.92607213 5
C 607516.231712159 7793552.93842478 10
A 08777200075
C 602712.314503365 7798159.40825128 3
C 615804.983781242 7800034.04266877 5
D 06751206455A
C 616439.872399631 7800034.51657842 10
```

Figura 4: Exemplo de arquivo de atualizações e consultas

D id Desativar o ponto de recarga associado ao identificador *id*.

O exemplo a seguir mostra o comando de desativação de uma mesma estação de recarga, mostrando as duas mensagens de saída possíveis.

Entrada	Saída
`D 00446106627`	Ponto de recarga 00446106627 desativado
`D 00446106627`	Ponto de recarga 00446106627 já estava desativado.

2.2 Uso de QuadTree como estrutura de Dados

A implementação original do aplicativo **BiUaiDi** adota como estrutura de dados um vetor das estações de recarga, que é utilizado para calcular as distâncias de um dado ponto de origem (posição instantânea de um carro) a cada uma das estações de recarga. As distâncias calculadas são colocadas em um segundo vetor, que é ordenado, e as estações associadas às distâncias mais curtas (na versão original dez) compõem a saída do aplicativo (ilustrado nas Figuras 2 e 3). É notório que essa complexidade de $O(nlogn)$ é indesejável e demanda uma reavaliação da estrutura de dados utilizada.

O segundo estágio da estratégia proposta (e do TP) é substituir a estrutura de dados original por uma mais eficiente. A proposta é utilizar uma QuadTree, estrutura de dados apropriada para dados georeferenciados, para armazenar as estações de recarga e identificar as estações mais próximas de um dada localização de origem. Assim, deve ser implementado um tipo abstrato de dados QuadTree que execute as operações de construção, inserção, as funções mencionadas (consultas e atualizações) e destruição da estrutura de dados. Embora essa implementação possa ser realizada utilizando alocação dinâmica, iremos implementá-la utilizando um vetor, onde os apontadores entre as células são índices do vetor de nodos. Um exemplo de implementação de árvore binária vetorizada é provida no minha.ufmg da disciplina.

Importante ressaltar a necessidade de reimplementar as funções de consulta de estações de recarga próximas, assim como a ativação e desativação de pontos de recarga. A expectativa é que a implementação dessas três funções tenha complexidade sub-linear, em particular logarítmica.

O aplicativo, à semelhança da versão original (*naive*), se inicia com a leitura de um arquivo e armazenamento em uma QuadTree de todos os dados das estações de recarga, seguido da leitura e processamento de um segundo arquivo contendo as várias funções de consulta, ativação e

desativação a serem executadas.

2.3 Avaliação experimental comparativa das versões do aplicativo

O terceiro estágio do projeto (TP) é a avaliação experimental dos aplicativos, tanto em termos de desempenho quanto à sua localidade de referência. O aspecto desempenho deve ser medido através do tempo execução do aplicativo como um todo e de cada uma das funções (construção, consultas e atualizações). O aspecto localidade de referência deve ser avaliado por métricas como distância de pilha, além de incluir discussões sobre a eficiência de cada estrutura de dados em executar as várias funções.
Para facilitar os experimentos, é disponibilizado um arquivo contendo todos os endereços da cidade de Belo Horizonte (725917 endereços). Como mencionado, os atributos que caracterizam uma estação de recarga são mostrados na Tabela 1.
Em termos da carga de trabalho a ser utilizada, devem ser consideradas as seguintes dimensões:

Número e distribuição das estações de recarga: A dimensão estações de recarga deve ser avaliada tanto em relação ao seu número quanto à distribuição espacial das estações. Por exemplo, a partir do elenco de estações possíveis pode-se selecionar as estações a serem consideradas de forma aleatória ou mesmo variando as densidades entre as regiões da cidade.

Número e natureza das consultas: A dimensão consultas deve ser avaliada tanto em relação ao seu número absoluto e relativo comparado às ativações e desativações, assim como às suas características, como concentração temporal e espacial. Por exemplo, as consultas podem estar associadas a uma trajetória real a ser realizada por um veículo.

Número e natureza das ativações e desativações: A dimensão ativações e desativações deve ser avaliada tanto em relação ao seu número, quanto ao seu impacto em termos de densidade espacial e frequência temporal (indicada pela taxa de intercalação de consultas e ativações/-desativações).

2.4 Extensão do aplicativo para uma arquitetura de dois níveis

Como mencionado, o aplicativo deve equipar os veículos da **BiUaiDi**, cujo sistema multimídia tem uma memória limitada. Para viabilizar a sua efetiva implementação, é necessário criar uma arquitetura de dois níveis que permita a replicação transparente da parte dos dados necessária para calcular as distâncias para as estações de recarga mais próximas. Essa foi a motivação de utilizar uma estrutura de dados que possua boa localidade de referência para a carga de trabalho do aplicativo.
Desta forma, além de adaptar a estrutura de dados QuadTree implementada, é necessário definir e implementar uma política de reposição de páginas.
Disponibilizamos no minha.ufmg a extensão do algoritmo de árvore binária implementado como vetor.

2.5 Avaliação experimental do aplicativo estendido

A avaliação experimental do aplicativo estendido deve avaliar tanto medidas de desempenho como a latência para determinar as estações de recarga mais próximas, quanto o funcionamento da arquitetura de dois níveis, em termos do número e volume de dados transferidos. Note que as ativações e desativações não devem ser contabilizadas como tráfego do carro.
Desta forma a avaliação deve considerar as seguintes dimensões:

Razão entre memória primária e secundária: É o aspecto principal do desempenho da arquitetura de dois níveis. Quanto maior for essa razão, mais dados podem ser mantidos na memória primária.

Número de estações considerado nas consultas: O número de estações considerado nas consultas indica a amplitude da varredura pelos vizinhos mais próximos na estrutura de dados.

Deve ser elaborado um plano experimental que exercite variações nessas dimensões e permita observar os impactos das variações nos valores. A avaliação deve incluir uma discussão dos resultados.

3 Como será feita a entrega

Como mencionado, o trabalho tem uma parte obrigatória e uma parte opcional. A seguir detalhamos essas submissões.

3.1 TP3 Obrigatório

A parte obrigatória do TP3 compreende duas submissões:

VPL TP3: Submissão do código que contemple os passos 1 e 2, a ser submetido até **19/08, 23:59** (o sistema vai ficar aberto madrugada adentro, mais para evitar problemas transientes de infraestrutura). Não vão ser aceitas submissões em atraso. Detalhes sobre a submissão do código são apresentados na Seção 3.4.

Relatório TP3: arquivo PDF contendo a documentação dos passos 1 e 2, assim como a avaliação experimental comparativa das versões do aplicativo (passo 3), conforme instruções, a ser submetido até **19/08, 23:59** (o sistema vai ficar aberto madrugada adentro, mais para evitar problemas transientes de infraestrutura). Não vão ser aceitas submissão em atraso. Detalhes sobre a submissão de relatório são apresentados na Seção 3.3.

3.2 TP3 Opcional

A parte opcional do TP3 compreende uma submissão:

TP3 Pontos extra: Submissão de um arquivo compactado (zip) contendo o código do passo 4 (arquitetura de dois níveis) e relatório PDF previsto no passo 5 até **21/08, 23:59** (o sistema vai ficar aberto madrugada adentro, mais para evitar problemas transientes de infraestrutura). O código deve estender o VPL TP3 (ou seja, não pode partir do código do colega) e o relatório pode replicar, quando pertinente, seções do Relatório TP3 submetido pelo próprio aluno. Ou seja, o aluno que não entregar o TP3 não pode submeter para a parte de pontos extras. Não vão ser aceitas submissões em atraso. Detalhes sobre a submissão opcional são apresentados nas Seções 3.3 e 3.4.

3.3 Documentação

A documentação do trabalho deve ser entregue em formato **PDF** e também **DEVE** seguir o modelo de relatório que será postado no Moodle. Além disso, a documentação deve conter **TODOS** os itens descritos a seguir **NA ORDEM** em que são apresentados:

1. **Capa:** Título, nome, e matrícula.
2. **Introdução:** Contém a apresentação do contexto, problema, e qual solução será empregada.
3. **Método:** Descrição da implementação, detalhando as estruturas de dados, tipos abstratos de dados (ou classes) e funções (ou métodos) implementados.
4. **Análise de Complexidade:** Contém a análise da complexidade de tempo e espaço dos procedimentos implementados, formalizada pela notação assintótica.
5. **Estratégias de Robustez:** Contém a descrição, justificativa e implementação dos mecanismos de programação defensiva e tolerância a falhas implementados.
6. **Análise Experimental:** Apresenta os experimentos realizados em termos de desempenho computacional e localidade de referência, assim como as análises dos resultados.

7. **Conclusões:** A Conclusão deve conter uma frase inicial sobre o que foi feito no trabalho. Posteriormente deve-se sumarizar o que foi aprendido.
8. **Bibliografia:** Contém fontes utilizadas para realização do trabalho. A citação deve estar em formato científico apropriado que deve ser escolhido por você.
9. Número máximo de páginas incluindo a capa: 10

Não se esqueça de descrever de forma concisa sua implementação da Quadtree e sua organização em dois níveis. Analise o impacto das sua escolhas na complexidade do algoritmo, compare com outras possíveis estruturas que também resolveriam o problema.

A documentação deve conter a descrição do seu trabalho em termos funcionais, dando foco nos algoritmos, estruturas de dados e decisões de implementação importantes durante o desenvolvimento.

Evite a descrição literal do código-fonte na documentação do trabalho.

Dica: Sua documentação deve ser clara o suficiente para que uma pessoa (da área de Computação ou não) consiga ler, entender o problema tratado e como foi feita a solução.

A documentação deverá ser entregue como uma atividade separada designada para tal no minha.ufmg. A entrega deve ser um arquivo `.pdf`, nomeado `nome_sobrenome_matricula.pdf`, onde nome, sobrenome e matrícula devem ser substituídos por suas informações pessoais.

3.4 Código

Você deve utilizar a linguagem C ou C++ para o desenvolvimento do seu sistema. O uso de estruturas pré-implementadas pelas bibliotecas-padrão da linguagem ou terceiros é terminantemente vetado. Você DEVE utilizar a estrutura de projeto abaixo junto ao Makefile:

```
– TP
    |– src
    |– bin
    |– obj
    |– include
    Makefile
```

A pasta **TP** é a raiz do projeto; **src** deve armazenar arquivos de código (*.c, *.cpp, ou *.cc); a pasta **include**, os cabeçalhos (headers) do projeto, com extensão *.h, por fim as pastas **bin** e **obj** devem estar vazias. O Makefile deve estar na raiz do projeto. A execução do Makefile deve gerar os códigos objeto *.o no diretório **obj** e o executável do TP no diretório **bin**. O arquivo executável **DEVE** se chamar **tp3.out** e deve estar localizado na pasta **bin**. O código será compilado com o comando:

```
make all
```

O seu código será avaliado através de uma **VPL** que será disponibilizada no moodle. Você também terá à disposição uma VPL de testes para verificar se a formatação da sua saída está de acordo com a requisitada. A VPL de testes não vale pontos e não conta como trabalho entregue. Um pdf com instruções de como enviar seu trabalho para que ele seja compilado corretamente estará disponível no Moodle.

4 Avaliação

- Corretude na execução dos casos de teste - (30% da nota total)
- Indentação, comentários do código fonte e uso de boas práticas - (10% da nota total)

- Conteúdo segundo modelo proposto na seção **Documentação** - (20% da nota total)
- Definição e implementação das estruturas de dados e funções - (15% da nota total)
- Apresentação da análise de complexidade das implementações - (5% da nota total)
- Análise experimental - (15% da nota total)
- Seu trabalho segue todas as instruções de entrega - (5% da nota total)

Se o programa submetido **não compilar**[1], seu trabalho não será avaliado e sua nota será **0**. Não será possível entregar os trabalhos com atraso.

5 Considerações finais

1. Comece a fazer esse trabalho prático o quanto antes, enquanto o prazo de entrega está tão distante quanto jamais estará.
2. Leia atentamente o documento de especificação, pois o descumprimento de quaisquer requisitos obrigatórios aqui descritos causará penalizações na nota final.
3. Certifique-se de garantir que seu arquivo foi submetido corretamente no sistema.
4. Plágio é crime. Trabalhos onde o plágio for identificado serão **automaticamente anulados** e as medidas administrativas cabíveis serão tomadas (em relação a todos os envolvidos). Discussões a respeito do trabalho entre colegas são permitidas. É permitido consultar fontes externas, desde que exclusivamente para fins didáticos e devidamente registradas na seção de bibliografia da documentação. **Cópia e compartilhamento de código não são permitidos.**

6 FAQ (Frequently asked Questions)

1. Posso utilizar qualquer versão do C++? NÃO, o corretor da VPL utiliza C++11.
2. Posso fazer o trabalho no Windows, Linux, ou MacOS? SIM, porém lembre-se que a correção é feita sob o sistema Linux, então certifique-se que seu trabalho está funcional em Linux.
3. Posso utilizar alguma estrutura de dados do tipo Queue, Stack, Vector, List, e etc..., do C++? NÃO.
4. Posso utilizar smart pointers? NÃO.
5. Posso utilizar o tipo String? SIM.
6. Posso utilizar o tipo String para simular minhas estruturas de dados? NÃO.
7. Posso utilizar alguma biblioteca para tratar exceções? SIM.
8. Posso utilizar alguma biblioteca para gerenciar memória? SIM.
9. As análises e apresentação dos resultados são importantes na documentação? SIM.
10. Os meus princípios de programação ligados a C++ e relacionados a engenharia de software serão avaliados? NÃO.
11. Posso fazer o trabalho em dupla ou em grupo? NÃO.
12. Posso trocar informações com os colegas sobre a teoria? SIM.
13. Posso utilizar IDEs, Visual Studio, Code Blocks, Visual Code, Eclipse? SIM.

[1] Entende-se por compilar aquele programa que, independente de erros no Makefile ou relacionados a problemas na configuração do ambiente, funcione e atenda aos requisitos especificados neste documento em um ambiente Linux.