


| | |
|--|---|
|  <p>UNIVERSIDADE DE COIMBRA FACULDADE DE CIÊNCIAS E TECNOLOGIA</p> <p>Departamento de Engenharia Informática</p> | <p>Projeto #3 Algoritmos e Estruturas de Dados</p> <p>2020-2021 – 2º Semestre</p> <p>VERSÃO 2.0</p> <p>Submissão RELATÓRIO (Infoestudante) e MOOSHAK: R 3.1 12 Mar '21 23h59m R 3.2 19 Mar '21 23h59m R 3.3 26 Mar '21 23h59m R 3.4 9 Abr '21 23h59m</p> |
| <p>Anotações: este projeto foi preparado para ser resolvido parcialmente no espaço das quatro sessões práticas. Vão ser disponibilizados quatro formulários dos relatórios parciais para cada uma das semanas de duração do projeto.</p> <p>É incentivado que os alunos discutam ideias e questões relativas ao trabalho proposto, mas é entendido que quer a reflexão final sobre os resultados obtidos, quer o código desenvolvido, são da autoria de cada estudante. Procedimentos contrários ao que é dito acima, nomeadamente cópia de código desenvolvido por colegas ou obtido da net é entendido como fraude. Para além de a fraude denotar uma grave falta de ética e constituir um comportamento não admissível num estudante do ensino superior e futuro profissional licenciado, esta prejudica definitivamente o processo de aprendizagem do infrator.</p> | |

Objetivos:

Com o desenvolvimento deste projeto pretende-se que o aluno consolide os conhecimentos sobre a estruturas de dados estudadas em Algoritmos e Estruturas de Dados com foco na (1) programação das estruturas (2) vantagens e desvantagens de cada uma, nomeadamente no que diz respeito a complexidade temporal e espacial (3) análise teórica e empírica da complexidade temporal.

NOTA: com exceção do subprojecto 3.4 os restantes subprojectos podem ser desenvolvidos pela ordem que o aluno achar mais adequada em função das estruturas já estudadas nas sessões teóricas da disciplina.

Sub-projeto 3.1: Merkle Tree para Integração numa Nova Criptomoeda

Uma *Merkle Tree* ou *Hash Tree* é uma estrutura muito usada em criptografia em que cada folha da árvore é anotada com o *hash code* referente a um bloco de dados. Os nós internos são anotados com o *hash code* dos seus filhos (ver figura).

Estas árvores servem para fazer uma verificação segura do conteúdo de extensas estruturas de dados. Vamos ainda assumir que a *Hash Tree* é formada a partir de uma árvore binária completa.

A figura abaixo apresenta uma *Merkle Tree*.

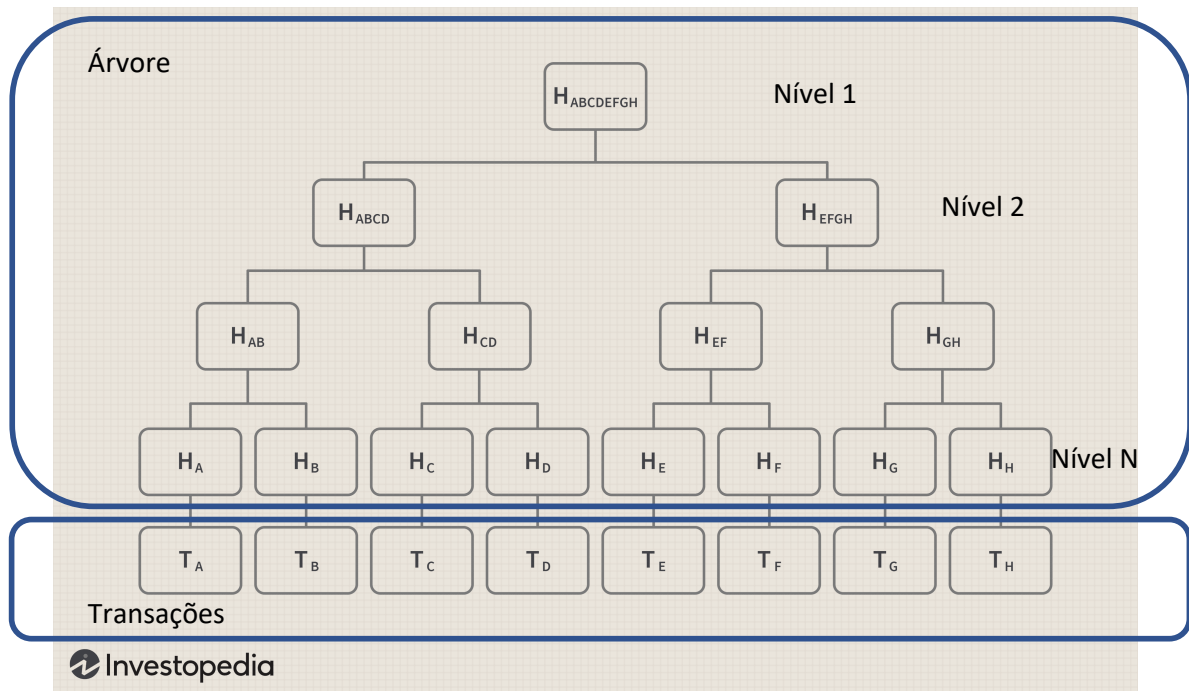


Image by Julie Bang © Investopedia 2020

Para desenvolver este subprojecto:

- Escolher a estrutura de dados mais adequada
- Implementar a estrutura de dados e desenvolver o programa de acordo com a entrada/saída pretendida
- Analisar a complexidade temporal

Entrada:

Como entrada o programa recebe uma linha com o número de transações a codificar, seguida de uma linha com as transações na forma BBBBddmmaaaa, separadas por espaço em branco, em que BBBB é um valor 0..9999, dd o dia 01..31, mm o mês 0..12, aa o ano 2007..2100. Sempre que os valores da dimensão tempo forem menores que 10 devem ser precedidos de 0. A transação é guardada e tratada na forma de um long integer.

O número de transações a codificar é uma potência de 2!

Saída:

Os valores de hash inscritos na Merkle Tree varrendo a árvore da raiz para as folhas e da esquerda para a direita. Um valor por linha, terminando também a última linha por “\n”. Nos exemplos abaixo foi usado o método `public int hashCode() ...` mas para que possamos ter uma função de hash igual para qualquer linguagem, devem usar a seguinte para um elemento

```
long hashcode(long x) {
    return x % 1000000007;
}
```

e esta para dois elementos

```

long hashCode(long x, long y) {
    int mod = 1000000007;
    return ((x % mod) + (y % mod)) % mod;
}

```

Exemplos:

Entrada

1
77706112012

Saída

706111473 // comentário: nível 1

Entrada

2
99920032021 77706112012

Saída

626142794 // comentário: nível 1
920031328 // comentário: nível 2
706111473 // comentário: nível 2

Entrada

4
1001092010 255501092010 99920032021 77706112012

Saída

128325015 // comentário: nível 1
502182228 // comentário: nível 2
626142794 // comentário: nível 2
1092003 // comentário: nível 3
501090225 // comentário: nível 3
920031328 // comentário: nível 3
706111473 // comentário: nível 3

Entrada

8
1001092010 255501092010 99920032021 77706112012 231012021 44403122020 6601122015 333303032019

Saída

666610402 // comentário: nível 1
128325015 // comentário: nível 2
538285387 // comentário: nível 2
502182228 // comentário: nível 3

| | |
|-----------|------------------------|
| 626142794 | // comentário: nível 3 |
| 634133733 | // comentário: nível 3 |
| 904151661 | // comentário: nível 3 |
| 1092003 | // comentário: nível 4 |
| 501090225 | // comentário: nível 4 |
| 920031328 | // comentário: nível 4 |
| 706111473 | // comentário: nível 4 |
| 231012021 | // comentário: nível 4 |
| 403121712 | // comentário: nível 4 |
| 601121973 | // comentário: nível 4 |
| 303029688 | // comentário: nível 4 |

Compreende a elaboração do Relatório 3.1 (formulário fornecido) e vai incidir sobre a análise de complexidade temporal do programa desenvolvido.

O relatório a realizar com base no formulário disponibilizado deve ter em conta:

- Tabela concisa com as medições efetuadas;
- Gráficos com as medições e resultado da regressão para cada solução desenvolvida;
- Reflexão crítica sobre o resultado da regressão e possíveis *outliers*.
- Análise de complexidade com base nos resultados empíricos obtidos.

Relatório (a submeter no inforestudante)

Sub-projeto 3.2: Sistema de Gestão de Vacinação

Pretende-se desenvolver um programa que guarda registos dos utentes da uma rede nacional de vacinação. Cada registo compreende o **número de utente** seguido um conjunto de dados associados de vacinação. Para cada utente é inicialmente criado um registo que depois vai ser acedido com bastante frequência (muito mais consultas ao sistema do que inserções, seja para incluir novas vacinações seja para verificar se o utente tem uma determinada vacina em dia).

As operações (comandos) possíveis sobre o Sistema de Vacinação são:

1. ACRESCENTA <numUtente> <vacina> <data limite>

Se ainda não existe um registo com < numUtente > cria esse registo. Se já existe verifica se a <vacina> já existe e nesse caso atualiza <data limite>, se vacina ainda não existe acrescenta <vacina> e <data limite>. A data deve ser inserida no formato ddmmyyyy.

Terminada a inserção deve imprimir uma linha com “NOVO UTENTE CRIADO” se ainda não existir o < numUtente > no sistema; “NOVA VACINA INSERIDA” se já existe *numUtente* mas ainda não tem informação sobre essa vacina; “VACINA ATUALIZADA” se vacina já existe, caso em que só atualiza data.

2. CONSULTA < numUtente >

Mostra todas as vacinas e datas limite associadas a < numUtente >. Vacinas apresentadas por ordem alfabética. Termina com uma linha com a palavra “FIM”.

Se < numUtente > não está presente no sistema devolve uma linha com “NAO ENCONTRADO”.

3. LISTAGEM

Faz a listagem de todos os < numUtente > guardados e respectivas <vacinas> e <data limite> seguida de linha com a palavra “FIM”. < numUtente > e <vacinas> por ordem crescente.

4. APAGA

Elimina todos os registos no sistema, de seguida imprime a linha com “LISTAGEM DE NOMES APAGADA”

5. FIM

Termina a sequência de comandos com uma linha com a palavra “FIM”:

Para desenvolver este subprojecto:

- A. Escolher a estrutura de dados principal mais adequada.
- B. Implementar a estrutura de dados e desenvolver o programa de acordo com a entrada/saída pretendida.
- C. Analisar a complexidade temporal.

Entrada:

Como entrada o programa recebe todos os comandos a realizar, um por linha.

Saída:

Os resultados para todos os comandos realizados, tal como especificado acima.

Exemplos:

Entrada

```
APAGA
ACRESCENTA 12340 polio 20022025
ACRESCENTA 56700 covid 10082023
ACRESCENTA 88100 tuberculose 15122030
CONSULTA 56700
ACRESCENTA 56700 covid 10102024
ACRESCENTA 56700 papeira 12112026
LISTAGEM
APAGA
FIM
```

Saída

LISTAGEM DE NOMES APAGADA

NOVO UTENTE CRIADO

NOVO UTENTE CRIADO

NOVO UTENTE CRIADO

56700 covid 10082023

FIM

VACINA ATUALIZADA

NOVA VACINA INSERIDA

12340 polio 20022025

56700 covid 10102024 papeira 12112026

88100 tuberculose 15122030

FIM

LISTAGEM DE NOMES APAGADA

Entrada

ACRESCENTA 23456 polio 20032023

CONSULTA 23456

APAGA

CONSULTA 23456

FIM

Saída

NOVO UTENTE CRIADO

23456 polio 20032023

FIM

LISTAGEM DE NOMES APAGADA

NAO ENCONTRADO

Entrada

ACRESCENTA 234 polio 20122025

ACRESCENTA 123 zeneca 12122030

ACRESCENTA 234 abacus 12032026

ACRESCENTA 123 covid 2122026

ACRESCENTA 123 abacus 2122032

CONSULTA 123

LISTAGEM

APAGA

CONSULTA 123

FIM

Saída

NOVO UTENTE CRIADO

NOVO UTENTE CRIADO

NOVA VACINA INSERIDA

NOVA VACINA INSERIDA

NOVA VACINA INSERIDA

123 abacus 2122032 covid 2122026 zeneca 12122030

FIM

123 abacus 2122032 covid 2122026 zeneca 12122030

234 abacus 12032026 polio 20122025

FIM

LISTAGEM DE NOMES APAGADA

NAO ENCONTRADO

Compreende a elaboração do Relatório 3.2 (formulário fornecido) e vai incidir sobre a análise de complexidade temporal do programa desenvolvido.

O relatório a realizar com base no formulário disponibilizado deve ter em conta:

- Tabela concisa com as medições efetuadas.
- Gráficos com as medições e resultado da regressão para cada solução desenvolvida.
- Reflexão crítica sobre o resultado da regressão e possíveis *outliers*.
- Análise de complexidade com base nos resultados empíricos obtidos.

Relatório (a submeter no inforestudante)

Sub-projeto 3.3: Sistema de Gestão de Clientes Preferenciais

Pretende-se desenvolver um programa que guarda o registo dos clientes de empresa, com a particularidade de o acesso a estes dados ser bastante assimétrica – 90% dos acessos são feitos a 5% dos clientes.

As operações (comandos) possíveis sobre o Sistema de Gestão de Clientes são:

1. CLIENTE <nome> <morada> <volume de compras em euros>

Se ainda não existe um registo com <nome> cria esse registo. Se já existe devolve a mensagem “CLIENTE JA EXISTENTE” e não faz nada no registo de clientes. Se insere um novo cliente termina com uma linha com a frase “NOVO CLIENTE INSERIDO”

O <nome> é composto por uma única palavra e a <morada> pela palavra *Rua* ou *Avenida* seguida de uma única palavra. O <volume de compras em euros> é do tipo inteiro.

2. AQUISICAO <nome> <volume de compras em euros>

Adiciona <volume de compras em euros> ao valor anteriormente guardado. Termina com uma linha com a frase “AQUISICAO INSERIDA”. SE <nome> não existe no sistema termina com a frase “CLIENTE NAO REGISTADO”.

2. CONSULTA <nome>

Mostra os dados do cliente, a saber <nome> <morada> <volume de compras em euros> a que se segue a linha “FIM”.

Se <nome> não existe no sistema termina com uma linha com a frase “CLIENTE NÃO REGISTADO”.

3. LISTAGEM

Faz a listagem de todos os <nomes> guardados e respectivas <morada> <volume de compras em euros>, um por linha. A listagem deve apresentar os clientes por ordem alfabética. Termina com uma linha com a palavra “FIM”.

<nomes> apresentados por ordem alfabética.

4. APAGA

Elimina todos os registos no sistema. Imprime uma linha com “LISTAGEM DE CLIENTES APAGADA”.

5. FIM

Termina a sequência de comandos.

Para desenvolver este subprojecto:

- A. Escolher a estrutura de dados principal mais adequada.
- B. Implementar a estrutura de dados e desenvolver o programa de acordo com a entrada/saída pretendida.
- C. Analisar a complexidade temporal.

Entrada:

Como entrada o programa recebe de uma só vez todos os comandos a realizar, um por linha.

Saída:

Os resultados para todos os comandos ativados, tal como especificado acima.

Exemplos:

Entrada

APAGA
CLIENTE Joao Rua Dourada 500
CLIENTE Rui Rua Verde 2000
CLIENTE Manuel Rua Cheia 30000
CONSULTA Rui
AQUISICAO Rui 100000
LISTAGEM
APAGA
FIM

Saída

LISTAGEM DE CLIENTES APAGADA

NOVO CLIENTE INSERIDO

NOVO CLIENTE INSERIDO

NOVO CLIENTE INSERIDO

Rui Rua Verde 2000

FIM

AQUISICAO INSERIDA

Joao Rua Dourada 500

Manuel Rua Cheia 30000

Rui Rua Verde 102000

FIM

LISTAGEM DE CLIENTES APAGADA

Compreende a elaboração do Relatório 3.3 (formulário fornecido) e vai incidir sobre a análise de complexidade temporal do programa desenvolvido.

O relatório a realizar com base no formulário disponibilizado deve ter em conta:

- Tabela concisa com as medições efetuadas;
- Gráficos com as medições e resultado da regressão para cada solução desenvolvida;
- Reflexão crítica sobre o resultado da regressão e possíveis *outliers*.
- Análise de complexidade com base nos resultados empíricos obtidos;

Relatório (a submeter no inforestudante)

Sub-projeto 3.4: Análise Comparativa das Estruturas de Dados usadas nos subprojetos

Compreende a elaboração do Relatório 3.4 (template fornecido) e vai incidir sobre os seguintes pontos:

- 1) Análise geral comparativa das várias estruturas de dados usadas (PROS/CONS).
- 2) Para cada um dos subprojectos descrição da tomada de decisão sobre estruturas de dados para cada um dos subprojetos

Relatório (a submeter no inforestudante)

Relatório Projeto 3.1 AED 2020/2021 Versão 1.0

Nome: _____ N° Estudante: _____

TP (inscrição): _____ Login no Mooshak: _____

N° de horas de trabalho: _____ H Aulas Práticas de Laboratório: _____ H Fora de Sala de Aula: _____ H

(A Preencher pelo Docente) CLASSIFICAÇÃO:

Comentários:

1. Análise Empírica de Complexidade

Correr a implementação do projeto 3.1 para um número crescente de transações e obter os tempos de execução (excluindo tempo de leitura e impressão de resultados). Produzir tabela, gráfico e regressão relevantes.

| | |
|--|--|
| | |
|--|--|

A expressão $f(N)$ está de acordo com o esperado? Justifique.

O projeto 3.1 pode ser implementado seguindo uma abordagem iterativa e uma recursiva.

Explique sucintamente o essencial das duas implementações em termos de estruturas de dados utilizadas e da propagação dos *hashcodes* na árvore

Relatório Projeto 3.2 AED 2020/2021

Nome: _____ N° Estudante: _____

TP (inscrição): _____ Login no Mooshak: _____

N° de horas de trabalho: _____ H Aulas Práticas de Laboratório: _____ H Fora de Sala de Aula: _____ H

(A Preencher pelo Docente) CLASSIFICAÇÃO:

Comentários:

1. Análise Empírica de Complexidade

Correr a implementação do projeto 3.2 para um número crescente de registos/acessos com dois cenários: (1) 10% de inserções (2) 90% de inserções. Obter os tempos de execução (excluindo tempo de leitura e impressão de resultados). Produzir respetivas tabelas, gráficos e regressões relevantes.

Cenário 1

Cenário 2

Os tempos de execução estão de acordo com o esperado? Justifique.

Relatório Projeto 3.3 AED 2020/2021

Nome: _____ Nº Estudante: _____

TP (inscrição): _____ Login no Mooshak: _____

Nº de horas de trabalho: _____ H Aulas Práticas de Laboratório: _____ H Fora de Sala de Aula: _____ H

(A Preencher pelo Docente) CLASSIFICAÇÃO:

Comentários:

1. Análise Empírica de Complexidade

Correr a implementação do projeto 3.3 para um número crescente de acessos com dois cenários: (1) 90% dos acessos são feitos a 5% dos clientes (2) todos os clientes têm sensivelmente o mesmo número de acessos. Obter os tempos de execução (excluindo tempo de leitura e impressão de resultados). Produzir respetivas tabelas, gráficos e regressões relevantes.

| | |
|--|--|
| | |
| | |

A evolução dos tempos de execução está de acordo com o esperado? Justifique.

| |
|--|
| |
| |
| |
| |
| |
| |

Relatório Projeto 3.4 AED 2020/2021 Versão 1.0

Nome: _____ Nº Estudante: _____

TP (inscrição): _____ Login no Mooshak: _____

Nº de horas de trabalho: _____ H Aulas Práticas de Laboratório: _____ H Fora de Sala de Aula: _____ H

(A Preencher pelo Docente) CLASSIFICAÇÃO:

Comentários:

Estrutura de Dados Principal usada em cada sub-projeto:

PROJ 3.1 _____

PROJ 3.2 _____

PROJ 3.3 _____

| Estruturas de Dados usadas | | | |
|---|---|---|---|
| VANTAGENS GERAIS (max 3) | <ul style="list-style-type: none">••• | <ul style="list-style-type: none">••• | <ul style="list-style-type: none">••• |
| DESVANTAGENS GERAIS (max 3) | <ul style="list-style-type: none">••• | <ul style="list-style-type: none">••• | <ul style="list-style-type: none">••• |
| Justificação para a escolha no PROJ 3.1 | <hr/> <hr/> <hr/> | | |
| Justificação para a escolha no PROJ 3.2 | <hr/> <hr/> <hr/> | | |
| Justificação para a escolha no PROJ 3.3 | <hr/> <hr/> <hr/> | | |