Relatório Projeto 2 V1.0 AED 2024/2025

Nome: Vasco Guilherme da Silva Jacinto Gaspar Alves Nº Estudante: 2022228207
PL (inscrição): PL-6 Email: vasco.guilherme.alves@gmail.com

IMPORTANTE:

- **Os textos das conclusões devem ser manuscritos...** o texto deve obedecer a este requisito para não ser penalizado.
- Texto para além das linhas reservadas, ou que não seja legível para um leitor comum, não será tido em conta.
- **O relatório deve ser submetido num único PDF** que deve incluir os anexos. A não observância deste formato é penalizada.

1. Planeamento

	Semana 1	Semana 2	Semana 3	Semana 4	Semana 5
Árvore Binária					
Árvore AVL					
Árvore VP					
Árvore TREAP					
Finalização Relatório					

4. Conclusões *(as linhas no template representam a extensão máxima de texto manuscrito)*

4.1 Tarefa 1

A árvore binária Tem a maior complexidade para as operações de pesquisa e inserção. No pior caso, inserir requer pesquisar todos os elementos. Sendo assim, inserir N chaves tem uma complexidade de $O(n \cdot (n+1)/2)$ (Soma de Gauss), ou aproximadamente $O(n^2)$. A regressão feita no gráfico correspondente suporta esta conclusão.

Por outro lado, a natureza implícita desta estrutura e a falta de ordenação permite remover elementos em $O(1)$ e N elementos em $O(N)$.

4.2 Tarefa 2

A árvore AVL Tem o melhor balanceamento por natureza, e logo deveria realizar ~~rotações~~ um maior número de rotações que os restantes. Para conjuntos pequenos de keys esta overhead pode não valer a pena, mas para conjuntos maiores o balanceamento mais estrito resulta em operações de pesquisa mais eficientes.

A AVL mantém uma complexidade de tempo de $O(\log_2 N)$ consistente para as suas pesquisas.

4.3 Tarefa 3

A árvore Red-Black é menos estritamente balanceada que a AVL, o que é benéfico para conjuntos com tamanhos mais pequenos e tamanhos médios. O menor número de rotações significa que é mais rápido inserir em média, mas à medida que o balanceamento vai diminuindo, esta vantagem pode desaparecer, o que aparenta ser o caso nas ~~outras~~ medidas tomadas.

A operação de pesquisa ~~tipicamente~~ pode perder eficiência quando o balanço diminuir.

4.4 Tarefa 4

A Treap teve o melhor desempenho de todas as árvores. A sua natureza aleatória permite inserir elementos com um número mínimo de rotações. O balanceamento é o menos restrito de todas as árvores logo as pesquisas vão ser mais demoradas em média do que nas árvores AVL e Red Black.

Nota também que para o conjuntos A e B, os quais são ordenados, o desempenho é significativamente menor que para os conjuntos aleatórios C e D.

Anexo B - Código de Autor

```
/* Código feito para C99, compilado com GCC 14.2.1 com flags --std=c99 -O2 e --fast-math
 *
 * Hardware Original: TOSHIBA SATELLITE_C50-A PSCG6P-01YAR1,
 * CPU: Intel i5-3320M (4) @ 3.300GHz,
 * GPU: Intel 3rd Gen Core processor Graphics Controller
 * RAM: 7821MiB, SSD SATA3 1TB
 *
 * Aluno: Vasco Alves, 2022228207
*/

#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <assert.h>
#include <time.h>

#define RESIZE_FACTOR 1.61803

#define IDX_INVALID 4294967295

#define SEED 95911405

#define BLACK 0
#define RED 1

typedef uint32_t idx_t;
typedef int32_t key_t;

static int32_t g_treesize;
static int32_t g_average;
static uint32_t g_rotation_count = 0;

typedef struct BinTreeNode {
```