

Relatório TP2 AED 2013/2014

Nome: _____ TP (inscrição): _____

Nº Estudante: _____ E-mail/login no *Mooshak*: _____

Nº de horas de trabalho (incluindo aulas TP e PL): _____

Meta A: _____ Meta B: _____ Meta C: _____ Meta D: _____ Meta E: _____ Relatório: _____

(A Preencher pelo Docente) Relatório:

(A Preencher pelo Docente) Código:

(A Preencher pelo Docente) Defesa Individual:

(A Preencher pelo Docente) CLASSIFICAÇÃO:

COMENTÁRIOS MAIS RELEVANTES

(opções específicas tomadas na implementação, eventuais problemas com *Mooshak*, etc.):

- primeiro foi usada a recursão na procura e inserção mas depois foi mudada.
- idem para a abordagem em termos de POO
- escolhi usar ponteiro para pai dentro da estrutura por ser muitas vezes necessário
- na eliminação das árvores binárias e AVL é sempre escolhido para substituição o nó mais à direita do ramo esquerdo da sub-árvore cuja raiz é o nó a eliminar. No caso da AVL é depois equilibrada a árvore. No caso das árvores vermelhas e pretas, dada a complexidade da operação da remoção, optou-se por pôr o contador do nó a 0 e ao fim de 300 eliminações e/ou no fim da construção da árvore todos os elementos não nulos são copiados para uma nova árvore. O mesmo processo é usado para as árvores aleatórias. No caso da lista de saltos são simplesmente encontrados todos os ponteiros para o elemento em questão e todos esses ponteiros são substituídos pelos ponteiros do elemento a remover.
- só depois de ter os ficheiros de teste para o relatório é que vi que as AVL estavam erradas na função equilibrar. Foi depois submetido o ficheiro corrigido.
- random n usei treap
- rb é bottom-up
- na lista de saltos é sempre calculado um array com todos os ponteiros que poderiam apontar para um novo elemento, sendo depois usados apenas os necessários para o nº de níveis do novo elemento
- os ficheiros usados para testes são sempre os últimos submetidos.

COMPLEXIDADE TEMPORAL – Medições Experimentais

Indicadores (*)	T1.txt	T1_D.txt	T1_DO.txt	T2.txt	T3.txt	T4.txt	T4_D.txt
Árvores Binárias de Pesquisa							
# nós atravessados	966449	1113282	627675942	1743615	3077846	6233116	7148687
# rotações							
Árvores AVL							
# nós atravessados	693311	856060	871553	1271831	2433867	4735374	5277309
# rotações	6866	12691	18337	6832	6984	7088	14025
Árvores Vermelhas e Pretas							
# nós atravessados	699183	865829	1550587	1278331	2443834	4811071	5305112
# rotações	5822	10677	18326	5782	5781	5839	11691
Listas de Saltos							
# nós atravessados	854380	1106253	794443	1723832	3168055	5184360	6554813
# rotações							
Árvores Aleatórias							
# nós atravessados	750574	806697	82556	1495739	3021984	5922753	6602756
# rotações							

(*) caso o indicador não se aplique ao caso específico em análise considere o valor “0” ou “-”.
considere que uma rotação dupla conta como duas rotações. No caso das medições de

REFLEXÃO SOBRE OS RESULTADOS MEDIDOS / ESPERADOS:

(pode continuar no verso se necessário):

- AVL demorou mais tempo e teve mais rotações que AVP, como esperado.
VOLTAR A FAZER AS MEDIÇÕES DAS AVL CONTANDO COM A CALCHEIGHT
- Binárias foram as com mais atravessamentos (embora as listas de saltos estivessem perto)
- Nas degeneradas todas têm melhor desempenho que as binárias
- árvores aleatórias próximas de avl e vp, como se esperaria, muitíssimo melhor no caso degenerado
- atravessamentos melhor nas avl que nas vp
- não considerei os removes
- nas árvores aleatórias não considerei os splits

COMPLEXIDADE ESPACIAL – observações mais importantes acerca do espaço ocupado em memória, quer ao nível da estrutura de dados quer ao nível da execução das operações de inserção e pesquisa (memória ocupada por estruturas de dados auxiliares, memória implicitamente ocupada por métodos recursivos, etc.). *(pode continuar no verso se necessário)*

As estruturas dos nós nas diferentes árvores são semelhantes, tendo sempre a palavra e o contador, ponteiros para esquerda, direita e pai. Nalguns casos temos mais 1 ou 2 booleanos, para indicar se o nó é o filho esquerdo do seu pai (para evitar um elevado número de comparações, dado que ocupa muito pouco espaço), e se é vermelho no caso das VP. Nas árvores AVL haverá também um inteiro extra correspondente ao factor de equilíbrio, facto que também acontece para as árvores aleatórias, como o nº de nós da sub-árvore. A árvore em si terá n nós, sendo portanto a complexidade espacial da estrutura da árvore de $O(n)$. Em termos de variáveis auxiliares teremos:

- Árvore Binária: Procura, Inserção : 1 variável auxiliar; Apagamento: 2 variáveis auxiliares, mais, com uma recursão, 2 variáveis adicionais

- AVL: cálculo da altura: em cada chamada recursiva são criadas 3 novas variáveis, pelo que para o cálculo da altura teremos $3n$ variáveis. Para equilibrar, função também recursiva, temos 2 chamadas a si própria, mais uma chamada à função de cálculo da altura. Assim sendo, tendo n chamadas à função (uma por cada nó), chamamos n vezes a função de cálculo de altura, usando $3*n*n$ variáveis.

- VP:

- Aleatórias:

No caso da lista de saltos...

PARAMETRIZAÇÃO DAS LISTAS DE SALTOS: parâmetros usados e justificação

foi escolhido como nível máximo o valor 16, pois $2^{16} = 65536$, valor maior que o número máximo de entradas diferentes dos ficheiros de input fornecidos.

O número de níveis de cada nó foi calculado pela fórmula $\log_2(\text{aleat})$, em que aleat é um número aleatório entre 0 e 1. Deste modo garante-se que a probabilidade de o número de níveis ser 1 é de $1/2$, a probabilidade de ser 2 é de $1/4$, ou seja, a probabilidade de ser n é de $1/2^n$.

CONCLUSÕES, ASPECTOS A SALIENTAR EM CADA ABORDAGEM

Árvores Binárias de Pesquisa

Como esperado são as que têm mais atravessamentos, sendo óbvio no caso da árvore degenerada que é um modo de pesquisa nada eficiente, já que a árvore binária se torna uma lista duplamente ligada de apenas um nível.

Árvores AVL

As árvores AVL são, em termos de atravessamento, melhores que as alternativas. Poderão, no entanto, demorar mais tempo dadas as muitas rotações que executam, pois a execução de uma rotação é uma tarefa pesada computacionalmente.

Árvores Vermelhas e Pretas

Nas árvores Vermelhas e Pretas encontram-se também menos atravessamentos que na maioria. O número de rotações é menor que o das AVL, apesar de, no caso T1_DO (degenerado) tem pouco menos rotações e muito mais atravessamentos. Nesta árvore a eliminação é de tal modo complexa que não tem utilidade implementá-la (a complexidade de operações faria com que se perdesse a eficiência da árvore). Assim, uma implementação que ponha apenas o contador a 0 e faça, periodicamente, a cópia da árvore para uma nova ignorando os contadores nulos, é mais eficiente.

Listas de Saltos

O caso em que se vê melhor a vantagem da lista de saltos é, sem sombra de dúvida, o caso T1_DO, já que a entrada ordenada de inputs é até mais vantajosa para este tipo de estrutura. Nos restantes casos tem performances semelhantes às das árvores binárias.

Árvores Aleatórias de Pesquisa

Nestas estruturas pode-se observar uma performance semelhante às AVL em termos de atravessamento, o que é o pretendido em termos teóricos, já que a probabilidade de um nó ficar como raiz de uma sub-árvore vai aumentando à medida que se desce na árvore, tornando-a bastante equilibrada para muitos elementos de input.