UNIVERSIDADE DE SÃO PAULO – USP
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO
DEPARTAMENTO DE CIÊNCIAS DE COMPUTAÇÃO
CURSO DE ENGENHARIA DE COMPUTAÇÃO

ESTRUTURA DE DADOS II - SCC0606

PROJETO II

João Pedro Fidelis Belluzzo
Luis Fernando Costa de Oliveira

São Carlos
2019

Lista de ilustrações

Sumário

Lista de tabelas

1 Parte I - Análise de Algoritmos de Busca

1.1 Busca sequencial simples

Foi implementada uma função de busca sequencial simples. Neste caso, o algoritmo deve percorrer o vetor linearmente até encontrar o elemento desejado, logo, temos que a complexidade de pior caso será:

$$T(n) = \Theta(n)$$

Tendo em vista o algoritmo descrito acima, o mesmo foi executado dez vezes, sendo anotado o tempo de busca dos elementos para cada execução. Os resultados obtidos encontram-se descritos na seguinte tabela:

Execução	Tempo (s)
1	$3,45$
2	$3,45$
3	$3,45$
4	$3,44$
5	$3,54$
6	$3,46$
7	$3,47$
8	$3,42$
9	$3,50$
10	$3,44$

Tabela 1 – Tempos de execuções para a busca sequencial simples.

Tempo médio	Desvio padrão
$3,45$	$0,04$

Tabela 2 – Tempo médio e desvio padrão para a busca sequencial simples.

1.2 Busca sequencial com realocação: mover-para-frente

Neste caso, foi implementada uma função de busca sequencial com realocação mover-para-frente. Logo, quando um elemento for encontrado, ele deve ser movido para a primeira posição do vetor, realizando, assim, um *shift* para a direita de todos os demais elementos. Temos que a complexidade deste modelo será:

$$T(n) = \Theta(n)$$

Então, o programa foi executado dez vezes, sendo registrado o tempo para cada execução. Os dados obtidos encontram-se na tabela a seguir:

Execução	Tempo (s)
1	$6,11$
2	$6,12$
3	$5,99$
4	$6,00$
5	$6,00$
6	$6,09$
7	$6,03$
8	$6,05$
9	$6,09$
10	$6,25$

Tabela 3 – Tempos de execuções: busca sequencial com realocação (mover-para frente).

Tempo médio	Desvio padrão
$6,07$	$0,08$

Tabela 4 – Tempo médio e desvio padrão: busca sequencial com realocação (mover-para-frente).

1.3 Busca sequencial com realocação: transposição

Neste exercício, foi implementada uma busca sequencial com realocação por transposição. No modelo em questão, quando o elemento buscado for encontrado, ele deve ser trocado com o elemento da posição anterior. Caso o elemento buscado seja o primeiro da sequência, nada deve ser feito. A complexidade, neste caso, é:

$$T(n) = \Theta(n)$$

Então, o programa foi executado dez vezes, sendo registrado o tempo para cada execução. Os dados obtidos encontram-se na tabela a seguir:

Execução	Tempo (s)
1	$3,45$
2	$3,48$
3	$3,45$
4	$3,43$
5	$3,48$
6	$3,44$
7	$3,49$
8	$3,44$
9	$3,47$
10	$3,49$

Tabela 5 – Tempos de execuções: busca sequencial com realocação (transposição).

Tempo médio	Desvio padrão
$3,46$	$0,02$

Tabela 6 – Tempo médio e desvio: busca sequencial com realocação (transposição).

1.4 Busca sequencial: tabela de índices

Neste caso, foi implementada a busca sequencial através de uma tabela de índices. Neste modelo, inicialmente, o intervalo que contém o elemento deve ser buscado na tabela de índices. Então, a partir desse intervalo encontrado, é realizada um busca sequencial simples no vetor de elementos. A complexidade dessa busca é:

$$T(n) = \Theta(n)$$

Então, o programa foi executado dez vezes, sendo registrado o tempo para cada execução. Os dados obtidos encontram-se na tabela a seguir:

Execução	Tempo (s)
1	$0,75$
2	$0,75$
3	$0,77$
4	$0,77$
5	$0,76$
6	$0,77$
7	$0,76$
8	$0,76$
9	$0,79$
10	$0,76$

Tabela 7 – Tempos de execuções: busca sequencial com tabela de índices.

Tempo médio	Desvio padrão
$0,76$	$0,01$

Tabela 8 – Tempo médio e desvio: busca sequencial com tabela de índices.

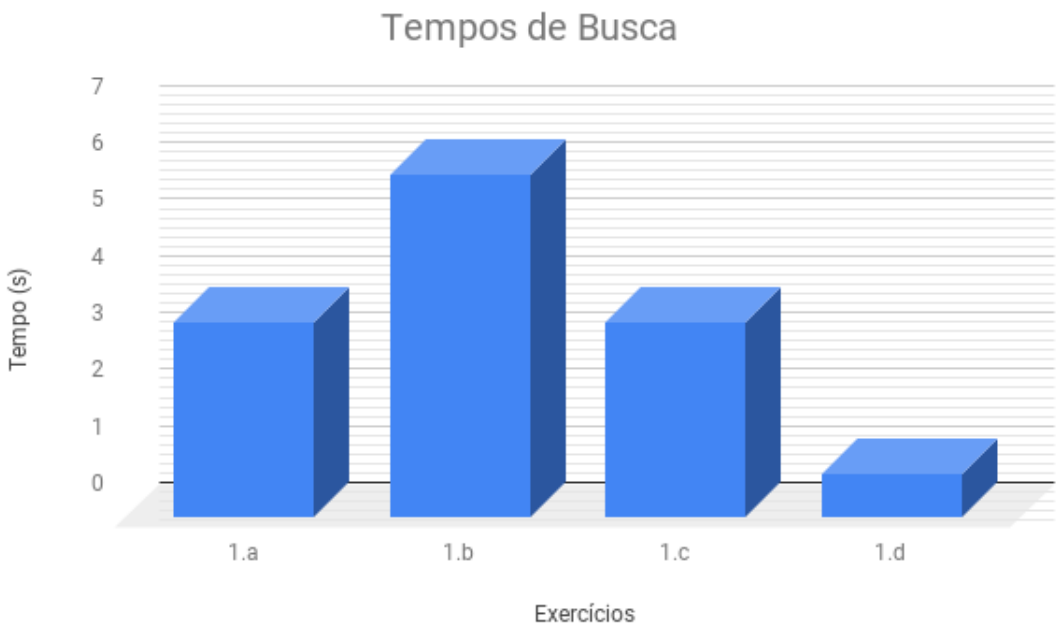
1.5 Análise dos resultados

Os resultados obtidos para cada variação da busca sequencial foram analisados e colocados na tabela a seguir:

Variação	Tempo médio (s)	Desvio padrão
Simples	$3,45$	$0,04$
Mover-para-frente	$6,07$	$0,08$
Transposição	$3,46$	$0,02$
Tabela de índices	$0,76$	$0,01$

Tabela 9 – Tempos médios e desvios padrão para as diferentes variações.

Em seguida, os valores foram postos no seguinte gráfico:



A partir do gráfico, nota-se que a busca através da tabela de índices apresentou um tempo médio muito melhor se comparada aos outros três métodos. Isso se deve à busca inicial na tabela de índices, que determina um intervalo menor de elementos para se realizar a busca sequencial simples, encontrando o elemento desejado mais rapidamente.

A busca com realocação através do método mover-para-frente, por sua vez, se mostrou o modelo mais lento. Nele, há a realização do *shift* de vários elementos quando o dado buscado é encontrado. Esse processo consome grande parte do tempo, já que são realizadas sucessivas trocas.

Já a busca sequencial simples e a que utilizava a técnica de transposição apresentaram tempos parecidos e intermediários entre o melhor e pior método. Isso acontece pois o método da transposição realiza apenas uma troca a cada elemento encontrado, tendo pouco impacto no tempo médio.

2 Parte II - Análise de Algoritmos de Espalhamento

2.1 Hash estático fechado com overflow progressivo

Foram implementadas funções de inserção e busca de elementos em um *Hash* estático fechado com *overflow* progressivo. Para tanto, foi tomado um vetor de tamanho máximo $B = 150001$ e duas funções *Hash* diferentes, de multiplicação e divisão.

2.1.1 Hash de Multiplicação

Neste caso, foi utilizada a função Hash de multiplicação. Então, a posição dos elementos a serem inseridos ou buscados no vetor era definida através da multiplicação inteira do valor pelo coeficiente $A = 0.6180$. O programa foi executado dez vezes, sendo os tempos de execução da inserção e busca de elementos anotados e dispostos na seguinte tabela:

Execução	Tempo de Inserção (s)	Tempo de Busca (s)
1	$0,401$	$0,859$
2	$0,401$	$0,843$
3	$0,396$	$0,846$
4	$0,395$	$0,851$
5	$0,392$	$0,840$
6	$0,395$	$0,836$
7	$0,389$	$0,864$
8	$0,409$	$0,840$
9	$0,390$	$0,841$
10	$0,398$	$0,843$

Tabela 10 – Tempos de execuções: *hash* de multiplicação (overflow progressivo).

	Tempo médio (s)	Desvio padrão
Inserção	$0,396$	$0,006$
Busca	$0,843$	$0,009$

Tabela 11 – Tempo médio e desvio: *hash* de multiplicação (overflow progressivo).

Para este modelo, foram reportadas 5058859 colisões.

2.1.2 Hash de Divisão

Neste caso, foi utilizada a função Hash de divisão. Então, a posição dos elementos a serem inseridos ou buscados no vetor era definida através do resto da divisão por $B = 150001$. O programa foi executado dez vezes, sendo os tempos de execução da inserção e busca de elementos anotados e dispostos na seguinte tabela:

Execução	Tempo de Inserção (s)	Tempo de Busca (s)
1	$0,076$	$0,177$
2	$0,073$	$0,172$
3	$0,072$	$0,169$
4	$0,072$	$0,160$
5	$0,072$	$0,160$
6	$0,072$	$0,163$
7	$0,073$	$0,165$
8	$0,072$	$0,163$
9	$0,072$	$0,163$
10	$0,072$	$0,170$

Tabela 12 – Tempos de execuções: *hash* de divisão (overflow progressivo).

	Tempo médio (s)	Desvio padrão
Inserção	$0,072$	$0,001$
Busca	$0,164$	$0,006$

Tabela 13 – Tempo médio e desvio: *hash* de divisão (overflow progressivo).

Para este modelo, foram reportadas 3172570 colisões.

2.2 Hash estático fechado com hash duplo

Foram implementadas funções de inserção e busca de elementos em um *Hash* estático fechado com *hash* duplo. Para tanto, foi tomado um vetor de tamanho máximo $B = 150001$ e uma função de *rehash* pelo algoritmo de *hash duplo*. O programa foi executado dez vezes, sendo os tempos de execução da inserção e busca de elementos anotados e dispostos na seguinte tabela:

Execução	Tempo de Inserção (s)	Tempo de Busca (s)
1	$0,113$	$0,213$
2	$0,106$	$0,226$
3	$0,108$	$0,213$
4	$0,115$	$0,234$
5	$0,108$	$0,222$
6	$0,126$	$0,231$
7	$0,109$	$0,236$
8	$0,109$	$0,228$
9	$0,113$	$0,227$
10	$0,110$	$0,228$

Tabela 14 – Tempos de execuções: *hash* duplo.

	Tempo médio (s)	Desvio padrão
Inserção	$0,109$	$0,006$
Busca	$0,228$	$0,008$

Tabela 15 – Tempo médio e desvio: *hash* duplo.

Neste caso, foram registradas 706898 colisões.

2.3 Hash estático aberto: encadeamento

Foram implementadas funções de inserção e busca de elementos em um *Hash* estático fechado com *hash* estático aberto. Para tanto, foi tomado um vetor de tamanho máximo $B = 150001$ e duas funções *Hash* diferentes, de multiplicação e divisão. As colisões foram tratadas seguindo a estratégia de encadeamento.

2.3.1 Hash de Multiplicação

Neste caso, foi utilizada a função Hash de multiplicação. Então, a posição dos elementos a serem inseridos ou buscados no vetor era definida através da multiplicação inteira do valor pelo coeficiente $A = 0.6180$. O programa foi executado dez vezes, sendo os tempos de execução da inserção e busca de elementos anotados e dispostos na seguinte tabela:

Execução	Tempo de Inserção (s)	Tempo de Busca (s)
1	$0,0555$	$0,1132$
2	$0,0552$	$0,1104$
3	$0,0541$	$0,1088$
4	$0,0545$	$0,1043$
5	$0,0548$	$0,1037$
6	$0,0238$	$0,1031$
7	$0,0530$	$0,1039$
8	$0,0533$	$0,1083$
9	$0,0537$	$0,1076$
10	$0,0553$	$0,1037$

Tabela 16 – Tempos de execuções: *hash* aberto de multiplicação (encadeamento).

	Tempo médio (s)	Desvio padrão
Inserção	$0,0543$	$0,0009$
Busca	$0,106$	$0,003$

Tabela 17 – Tempo médio e desvio: *hash* de multiplicação (encadeamento).

Para este modelo, foram reportadas 2551380 colisões.

2.3.2 Hash de Divisão

Neste caso, foi utilizada a função Hash de divisão. Então, a posição dos elementos a serem inseridos ou buscados no vetor era definida através do resto da divisão por $B = 150001$. O programa foi executado dez vezes, sendo os tempos de execução da inserção e busca de elementos anotados e dispostos na seguinte tabela:

Execução	Tempo de Inserção (s)	Tempo de Busca (s)
1	$0,020$	$0,042$
2	$0,017$	$0,037$
3	$0,017$	$0,035$
4	$0,019$	$0,039$
5	$0,016$	$0,034$
6	$0,016$	$0,034$
7	$0,017$	$0,035$
8	$0,019$	$0,037$
9	$0,017$	$0,036$
10	$0,017$	$0,034$

Tabela 18 – Tempos de execuções: *hash* aberto de divisão (encadeamento).

	Tempo médio (s)	Desvio padrão
Inserção	$0,017$	$0,001$
Busca	$0,035$	$0,003$

Tabela 19 – Tempo médio e desvio: *hash* aberto de divisão (encadeamento).

Para este modelo, foram reportadas 820909 colisões.

2.4 Análise dos resultados

No caso do algoritmo de espalhamento(Hash), pode-se dividir a análise em dois subtópicos, sendo um relacionado às diferentes funções de inserção e outro relacionado aos algoritmos de busca implementados.

2.4.1 Funções de Inserção

A partir dos resultados obtidos pelos diferentes métodos de espalhamento na função de inserção, foi criada a tabela a seguir:

Variação	Tempo médio (s)	Desvio padrão
Overflow Progessivo ($\times$)	$0,396$	$0,006$
Overflow Progessivo ($\div$)	$0,072$	$0,001$
Duplo	$0,109$	$0,006$
Encadeamento ($\times$)	$0,0543$	$0,0009$
Encadeamento ($\div$)	$0,017$	$0,001$

Tabela 20 – Inserção: tempos médios e desvios padrão para as diferentes variações.

Em seguida, os valores foram postos no seguinte gráfico:

A partir da análise do gráfico de tempos de inserção, podemos concluir que todos os algoritmos de espalhamento utilizados forneceram, em média, resultados extremamente satisfatórios. Vale destacar que todos os tempos medidos encontram-se abaixo dos 0.5 segundos, valor consideravelmente baixa tendo em vista o número de elementos trabalhados.

Também deve-se salientar a diferença de tempo observada entre o algoritmo 2.a com função *hash* de multiplicação e os demais programas. Neste caso específico, a média de tempo observada aproxima-se dos 0.4 segundos, enquanto nos demais a média encontra-se próxima a 0.1 segundo.

Pode-se enfatizar, também, a função de inserção implementada no algoritmo 2.c, através da função *hash* de divisão. Neste algoritmo, utilizando *hash* aberta e listas encadeadas, o tempo obtido foi o melhor dentre todos, com média inferior a 0.1 segundo. Tal fato demonstra a eficiência obtida através da utilização de listas encadeadas, que reduz as

colisões e facilita o tratamento das mesmas, evitando a necessidade de aplicação de funções de *rehash*

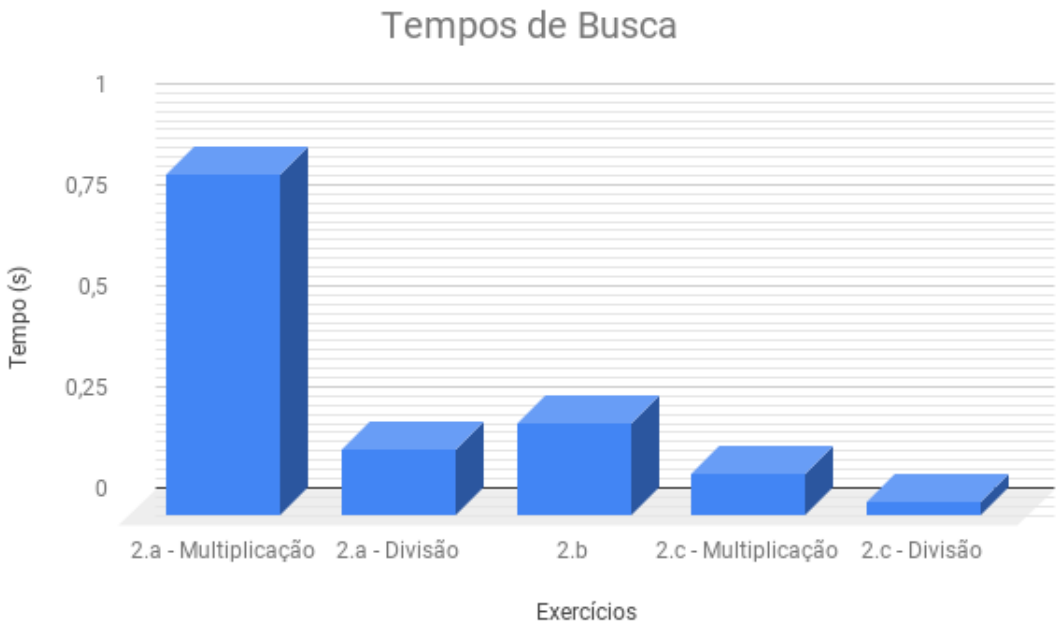
2.4.2 Funções de Busca

A partir dos resultados obtidos pelos diferentes métodos de espalhamento na função de busca, foi criada a tabela a seguir:

Variação	Tempo médio (s)	Desvio padrão
Overflow Progessivo ($\times$)	$0,843$	$0,009$
Overflow Progessivo ($\div$)	$0,164$	$0,006$
Duplo	$0,228$	$0,008$
Encadeamento ($\times$)	$0,106$	$0,003$
Encadeamento ($\div$)	$0,035$	$0,003$

Tabela 21 – Busca: tempos médios e desvios padrão para as diferentes variações.

Em seguida, os valores foram postos no seguinte gráfico:



A partir da análise dos gráficos, nota-se que os tempos de busca apresentam-se levemente superiores se comparados aos tempos de inserção. Entretanto, por mais que exista tal diferença, os tempos de busca ainda encontram-se consideravelmente baixos, estando sempre abaixo de 1 segundo.

Neste caso, assim como na inserção por método de espalhamento, a função de multiplicação do algoritmo 2.*a* foi a que apresentou maior disparidade nos resultados se comparada as demais, sendo também, a função com maior tempo necessário para realização da busca.

Também é importante citar a função de busca implementada no algoritmo 2.*c*, através da função *hash* de divisão. Neste algoritmo, assim como no caso da inserção, o tempo observado foi o menor dentre todos, com média inferior a 0.1 segundo. Logo, como no exemplo anterior, tal fato corrobora a eficiência fornecida devido a utilização de estruturas de dados mais sofisticadas.

Por fim, torna-se claro a superioridade do método de espalhamento em relação aos demais utilizados para busca. Já que, utilizando *hash*, os tempos obtidos foram consideravelmente menores do que o das demais implementações. Com isto, torna-se nítido a vantagem de se utilizar esse método de estrutura de dados, sempre que o mesmo encontra-se viável.