Universidade Federal de Minas Gerais
Departamento de Ciência da Computação
DCC004 - Algoritmos e Estruturas de Dados II
Profs. Cristiano Arbex Valle e Gisele L. Pappa

Trabalho Prático 2
Entrega: 02/07/2018
Valor: 25 pontos

Preparação para a Copa do Mundo

Breno Claudio de Sena Pimenta
Graduação em Engenharia Elétrica
Matrícula: 2017074424

1 Implementação

O programa cria uma lista de ranking de Copas do Mundo e de confrontos com três complexidades distintas:

- Método 1: Inserção em O(1) e pesquisa em O(n) utilizando lista encadeada.
- Método 2: Inserção em O(log n) e pesquisa em O(log n) utilizando árvore binária.
- Método 3: Inserção em O(1) e pesquisa em O(1) utilizando tabela hash de endereçamento aberto.

Cada um dos métodos entrega para o programa principal um vetor desordenado que contém as informações de ranking e confrontos para ordenação. A ordenação é feita utilizando o MergeSort, portanto em O(n log n).

Além dos argumentos obrigatórios (results.csv e o número do método desejado), há a possibilidade de informar o ano inicial ou o ano inicial e final, ou simplesmente o arquivo results.csv inteiro. O cálculo de linhas a serem armazenadas ocorre antes da cópia dos dados de resultados, portanto., somente é armazenado pelo programa os dados que serão usados.

2.1 Estrutura de Dados

Para o desenvolvimento do programa, foram implementados alguns tipos abstratos de dados. Esses tipos são listados a seguir. O método 3 não utiliza nenhum estrutura própria.

2.1.1 Estruturas Gerais

typeResult: estrutura para armazenar os dados do arquivo results.csv
typeTeam: estrutura para armazenar dados de uma seleção de futebol para fazer o ranking.
typeMatch: estrutura para armazenar dados de um confronto entre duas seleções para listar os confrontos.

2.1.1 Estruturas para o Método 1

typeTeamsList: ponteiro para uma lista de seleções.
typeTeamsNode: célula de uma lista de seleções.

typeMatchesList: ponteiro para uma lista de confrontos.
typeMatchesNode: célula de uma lista de confrontos.

2.1.1 Estruturas para o Método 2

typeTeamsTree: ponteiro para uma árvore de seleções.
typeTeamsLeaf: folha de uma árvore de seleções.

typeMatchesTree: ponteiro para uma árvore de confrontos.
typeMatchesLeaf: folha de uma árvore de confrontos.

2.3 Programa Principal

Inicialmente o programa testa se a quantidade de argumentos passado está correta, caso esteja, é chamada a função openAndCopyResults para importar os dados do arquivo results.csv passado por argumento. Em seguida, é testado o valor do terceiro argumento para identificar qual dos três métodos será utilizado. De forma geral, pode-se resumir que cada método criar um vetor do tipo typeTeams e typeMathes para que seja ordenado. Assim, a função do MergeSort é chamada para realizar a ordenação em seguida é criado o arquivo saidas.txt, que contém o resultado do ranking e a lista de confrontos. Por fim, os dados alocados dinamicamente são desalocados e o programa finaliza com sucesso.

2.4 Organização do Código, Decisões de Implementação e Detalhes Técnicos

O código foi dividido em três arquivos, sendo eles:

- *main.c*: que contém as diretrizes principais do programa.
- *world-cup.c*: que contém a definição das funções iterativas e recursivas.
- *world-cup.h*: que contém os cabeçalhos das funções iterativas e recursivas.

Foi definida a constante MAX_NAME 50 para limitar o número de caracteres de um nome. Foi definida a constante EMPTY "!@#$" para identificar se uma posição da tabela hash está vazia. Foi definida a constante MAX_HASH 40000 para ser o tamanho do vetor da tabela hash de ranking e confrontos.

O compilador utilizado foi o ambiente Debian (versão 9.3) instalado no Windows 10 Pro 64 bits. Para executá-lo, basta digitar ./tp1, seguido do nome do arquivo que contém os resultados, seguido pelo método a ser utilizado e opcionalmente ano de início ou ano de início e ano de fim da busca. O ano de início deve ser menor que o ano final.

3 Análise de Complexidade

A análise de complexidade de tempo será feita em função da variável n, i ou j, sua representação será especificada de acordo com a função analisada.

3.1 Funções Gerais

getTime(): O(1), independe do tamanho da entrada.
testAllocation: O(1), independe do tamanho da entrada.
createSaidasFile: O(i +j), sendo i o número de times e j o número de confrontos.
computeScores: O(1), independe do tamanho da entrada.

3.2 Funções de Leitura de Arquivo de Resultados

numberOfLines: O(n), sendo n o número de linhas de um arquivo.
skipLines: O(n), sendo n o número de linhas passadas para serem puladas.

defineCopyRange: O(n), sendo n o número de linhas de um arquivo.
openAndCopyResults: O(n), sendo n o número de linhas de um arquivo.

3.3 Funções do MergeSort para Times

compareTeams, compareMatches: O(1), independe do tamanho da entrada.
mergeTeams, mergeMatches: O(n), sendo n o número de subconjuntos divididos, ou seja, será de acordo com o número de países ou confrontos.
sortTeams, sortMatches: O(n/2), sendo n o tamanho do vetor de entrada.
mergesortTeams, mergesortMatches: O(n log n), sendo n o número de países ou confrontos.

3.4 Funções do Método 1

insertionTeamsList, insertionMatchesList: O(1), independe do tamanho da entrada.
searchTeamsList, searchMatchesList: O(n), sendo n o número de itens na lista.
createTeamsList, createMatchesList: O(n), sendo n o número de linhas de um arquivo.
freeAndConvertTeamsListToArray, freeAndConvertMatchesListToArray: O(n), sendo n o número de itens na lista.

3.5 Funções do Método 2

insertionTeamsTree, insertionMatchesTree: O(log n), sendo n o número de folhas da árvore.
searchTeamsTree, searchMatchesTree: O(log n), sendo n o número de folhas da árvore.
createTeamsTree, createMatchesTree: O(n), sendo n o número de linhas de um arquivo.
freeAndConvertTeamsTreeToArray, freeAndConvertMatchesTreeToArray: O(n), sendo n o número de folhas da árvore.

3.6 Funções do Método 3

i**nsertionTeamsHashTable, insertionMatchesHashTable:** O(1), independe do tamanho da entrada.
searchTeamsHashTable, searchMatchesHashTable: O(1) no caso médio, independe do tamanho da entrada. Ou O(n) no pior caso visto que há a possibilidade de haver colisão e haver de procurar outra posição para determinado nome.
createTeamsHashTable, createMatchesHashTable: O(n), sendo n o número de linhas de um arquivo.

3.7 Programa Principal

main: O(xn), a função que com a maior complexidade é o MergeSort, porém, o MergeSort atua com valores muito pequenos em relação ao arquivo passado (cerca de 39000 linhas). Portanto, no pior caso a complexidade do programa principal é O(xn) sendo que x é o número de linhas do arquivo passado e o número de buscas que ocorrem e n é o número de países. Assim sendo caso seja escolhido o método 1 será O(xn), para o método 2 será O(x log n) e método 3 será O(x).

4 Testes

Foram executados alguns testes para analisar a complexidade de tempo das funções criada. O principal objetivo foi identificar qual era o método mais rápido conforme a variação do tamanho da entrada. Cada ano (1872, 1910, 1940, 1980 e 2012) representa o ano inicial passado por argumento, então, quanto maior o ano,

menor o arquivo a ser processado. Além disso, para cada método foi realizado dez medições de tempo e foi realizado a média entre eles. Assim, foi possível elaborar a seguinte tabela:

TABELA 1: *Tempo de execução em segundos dos métodos de pesquisa e inserção para diferentes anos*

Métodos	**Complexidade de Caso Médio**	**1872**	**1910**	**1940**	**1980**	**2012**
Método 1	Pesquisa O(n) Inserção O(1)	0,4220	0,4270	0,4140	0,3770	0,1780
Método 2	Pesquisa O(log n) Inserção O(log n)	0,0340	0,0340	0,0330	0,0250	0,0130
Método 3	Pesquisa O(n) Inserção O(1)	0,0220	0,0210	0,0180	0,0140	0,0080

É possível concluir que o melhor método, em vista de tempo de execução é o método 3 (Tabela Hash), seguido pelo método 2 (Árvore Binária de Busca) e pelo método 1 (Pesquisa Sequencial). Além disso, apesar do método 3 ser O(1), a forma que a tabela é preenchida irá impactar no tempo do programa, devido às colisões e busca por posições vazias nesse caso.

5 Referências

[1] ZIVIANI, Nivio. Projeto de Algoritmos com Implementações em Pascal e C. 3. ed. [S.l.]: Cengage Learning, 2010. 660 p.

6 Anexos

Lista dos arquivos:

1. *main.c*
2. *world-cup.c*
3. *world-cup.h*