

UNIVERSIDADE FEDERAL DE ITAJUBÁ
Campus Itabira

Engenharia da Computação
ECO027 – Projeto e Análise de Algoritmos

Projeto #1 R2D2

25037 – André Viana Sena de Souza

Prof. M.e Walter Aoiama Nagai
walternagai@unifei.edu.br

Monitor: Emmanuell Miqueleti
emmanuelnogmiq@hotmail.com

Itabira – MG

29 de agosto de 2014

Sumário

1	Introdução	1
2	Algoritmos e estruturas de dados	2
2.1	Algoritmos	2
3	Apresentação e discussão dos resultados obtidos pelos experimentos	4
4	Bibliografia	5

1 Introdução

O projeto consiste em executar uma tarefa simples e de maneira otimizada. Para tal, a linguagem de programação escolhida foi C++, se utilizando da IDE CodeBlocks para armazenamento do arquivo de projeto. Os cabeçalhos do mesmo foram organizados da seguinte maneira:

- algoritmo.h - Dispõe todos métodos necessários para execução plena do algoritmo principal.
- arquivo.h - Contém uma enumeração dos arquivos a serem lidos e a função LEIA.
- cronometro.h - Define a estrutura utilizada para contagem de tempo das rotinas da aplicação.
- lista.h - Template de lista encadeada simples, a estrutura de dados base em todo o programa.

No módulo principal estão contidas a implementação da interface do usuário, e a lista contendo os resultados. Durante o desenvolvimento deste programa foi utilizado o compilador g++ 4.8 no sistema operacional Ubuntu 12.04.

2 Algoritmos e estruturas de dados

Diversos algoritmos foram utilizados para a implementação do projeto, por outro lado somente uma única estrutura foi criada, o template de lista encadeada simples. Essa lista conta com um contador de elementos, métodos de iteração por elementos, e também uma função que retorna um vetor com os dados inseridos. O processo de inserção utiliza um ponteiro extra, apontando para o último elemento da lista, acelerando o processo como um todo. Os métodos iteradores foram implementados com esse uso em mente:

```
for(lista->It_Inicio(); lista->Iterador()!=NULL; lista->It_Prox())  
    foo(*lista->Iterador());
```

Lista->Iterador() sempre aponta para um dos membros da lista, porém quando a lista chega ao fim, retorna um ponteiro nulo.

Não foi necessário implementar métodos de remoção, pois a função Descarregar() exclui todos os membros da lista sempre quando possível.

2.1 Algoritmos

- AlgoritmoR2D2 - Lê os arquivos a partir da função LEIA e a enumeração ArquivosComputadorCentral. LEIA retorna uma lista com todos os números contidos no arquivo e a partir deles é efetuada a contagem de pares de números positivos e seus opostos negativos. A função CONTE retorna a QUANTIDADE de pares, esse valor é passado como parâmetro para a função MAIORNUMEROPRIMO como o limite superior para o cálculo do algoritmo. Logo após, o Algoritmo de Invasão do R2D2 insere esse número primo em uma lista e procede com a leitura do próximo arquivo. Quando todos os arquivos terminam de ser lidos, o algoritmo retorna a lista de números primos.
- CONTE - Seguindo o princípio de que somente $x + (-x) = 0$, o algoritmo conta quantos pares resultam em zero. O primeiro passo é guardar os números em um vetor, ordenar os dados utilizando o algoritmo QuickSort, e iterar o vetor até que o primeiro número positivo apareça. A cada número negativo iterado, é efetuada uma pesquisa binária pelo seu oposto positivo no vetor. Se o par for completado, o contador é incrementado. Quando o primeiro número positivo é iterado, o algoritmo assume que todos os pares foram reconhecidos ($\text{par}(a,b) == \text{par}(b,a)$) e retorna o contador.
- BuscaBinária - Necessário para a função CONTE. Busca recursivamente pelo elemento elemento no vetor vetor. Retorna o index do elemento ou -1 caso ele não exista.

- QuickSort - Necessário para a função CONTE. Ordena recursivamente utilizando-se da técnica de divisão para conquista.
- MAIORNUMEROPRIMO - Em um for de limite superior a zero, busca por números primos, retornando o primeiro resultado encontrado.
- VerificaPrimo - Necessário em MAIORNUMEROPRIMO. Checa se o número é divisível por todos os números anteriores a ele. Esse processo pode ser otimizado utilizando-se de teoremas matemáticos para verificar se um número é primo ou não.

3 Apresentação e discussão dos resultados obtidos pelos experimentos

Para cada arquivo lido, a função CONTE apresentou uma duração diferente em seu tempo de execução:

Itens	Tempo de execução
1000	0 ms
2000	0 ms
4000	0 ms
8000	0 ms
16000	10 ms
32000	10 ms
64000	20 ms
128000	50 ms
256000	90 ms

O processo de otimização torna o programa muito mais rápido e eficiente, pois devido ao uso da busca binária e do quicksort para ordenação, o algoritmo mantém uma complexidade de $n \cdot \log_2 n$. Esses resultados ficam ainda mais evidentes quando em comparação aos testes feitos com uma busca linear em um container não ordenado, que levou aproximadamente 10000 vezes mais tempo (18,5 minutos).

```
1-Executar algoritmo.
2-Sair.
1
Efetuando leitura do arquivo 1000int.txt.
Leitura encerrada com sucesso.
Efetuando leitura do arquivo 2000int.txt.
Leitura encerrada com sucesso.
Efetuando leitura do arquivo 4000int.txt.
Leitura encerrada com sucesso.
Efetuando leitura do arquivo 8000int.txt.
Leitura encerrada com sucesso.
Efetuando leitura do arquivo 16000int.txt.
Leitura encerrada com sucesso.
Efetuando leitura do arquivo 32000int.txt.
Leitura encerrada com sucesso.
Efetuando leitura do arquivo 64000int.txt.
Leitura encerrada com sucesso.
Efetuando leitura do arquivo 128000int.txt.
Leitura encerrada com sucesso.
Efetuando leitura do arquivo 256000int.txt.
Leitura encerrada com sucesso.
47 229 829 3541 14087 56489 228311 911269 3636547

Process returned 0 (0x0)   execution time : 1114,864 s
Press ENTER to continue.
█
```

Figura 1: Resultado do programa com busca linear.

4 Bibliografia

- <http://stackoverflow.com>
- <http://rosettacode.org>