21.08.2024

Tarefa 1 - Busca em vetor ordenado

Júlia Tadeu - 2312392

Theo Canuto - 2311293

Professor Luiz Fernando Seibel

INF1010 - 3WA

2024.2

1. **Objetivo**

O objetivo deste trabalho é gerar um vetor com 10.000 números inteiros aleatórios no intervalo de [0, 10.000], ordenar o vetor gerado, e então gerar um novo conjunto de 10.000 números aleatórios. Em seguida, buscar esses números no vetor previamente ordenado, listar aqueles que não foram encontrados, e medir o tempo de execução do processo de busca.

2. **Estrutura do programa**

O programa é composto por três funções principais, além da função main. Cada uma delas desempenha um papel específico na execução das tarefas propostas no trabalho. A seguir, são descritas essas funções:

*Função genNumbers(int *vNums):*

- Descrição: Esta função é responsável por gerar números inteiros aleatórios e preencher o vetor fornecido como parâmetro. Ela utiliza a função rand() para gerar números dentro do intervalo [0, 10.000].
- Parâmetros: int *vNums: Ponteiro para o vetor de inteiros que será preenchido com números aleatórios.
- Retorno: Não possui retorno (função void).

*Função sortArray(int *vNums):*

- Descrição: Ordena o vetor passado como parâmetro em ordem crescente. A função implementa um algoritmo simples de ordenação baseado em comparação e troca de elementos.
- Parâmetros: int *vNums: Ponteiro para o vetor de inteiros que será ordenado.
- Retorno: Não possui retorno (função void).

Função int findDifferences(int *v1, int *v2, int *numDifferences):*

- Descrição: Compara dois vetores e identifica os números que estão presentes no segundo vetor (v2) mas não no primeiro (v1). A função utiliza a busca binária para verificar a existência dos elementos de v2 em v1. Se um número de v2 não for encontrado em v1, ele é adicionado a um vetor de diferenças.
- Parâmetros: int *v1: Ponteiro para o primeiro vetor (ordenado), int *v2: Ponteiro para o segundo vetor (não ordenado), int *numDifferences: Ponteiro para uma variável que armazenará o número total de diferenças encontradas.
- Retorno: Retorna um ponteiro para o vetor que contém os números não encontrados em v1. Se ocorrer um erro de alocação de memória, a função retorna NULL.

Função main(void):

- Descrição: É o ponto de entrada do programa.
- Esta função:

    - Inicializa o gerador de números aleatórios com srand(time(NULL)). Gera e ordena o primeiro vetor (vNums).
    - Gera o segundo vetor (vNums2).
    - Busca as diferenças entre os dois vetores utilizando a função findDifferences.
    - Exibe os vetores gerados e os números que não foram encontrados em vNums.
    - Calcula e exibe o tempo de execução total do programa.
- Parâmetros: Não possui parâmetros.
- Retorno: Retorna um inteiro (int) indicando o status de execução do programa, onde 0 geralmente indica que o programa foi executado com sucesso.

3. **Solução**

A solução deste trabalho foi desenvolvida seguindo uma abordagem estruturada e dividida em etapas claras para atingir o objetivo proposto. Abaixo está a descrição passo a passo da solução implementada:

a. Geração dos Vetores: inicialmente, dois vetores de inteiros (vNums e vNums2) foram gerados, ambos contendo 10.000 números aleatórios no intervalo de [0, 10.000]. A função genNumbers foi responsável por preencher esses vetores.

b. Ordenação do Primeiro Vetor: o vetor vNums foi ordenado em ordem crescente utilizando a função sortArray, que implementa um algoritmo de ordenação simples baseado na comparação e troca de elementos.

c. Busca de Diferenças entre os Vetores: após a geração e ordenação, o segundo vetor (vNums2) foi comparado com o primeiro vetor ordenado. Para cada elemento de vNums2, foi realizada uma busca binária no vetor vNums utilizando a função findDifferences. Essa função verifica se cada número de vNums2 está presente em vNums. Se não estiver, o número é adicionado a uma lista de diferenças.

d. Listagem dos Números Não Encontrados: os números que não foram encontrados em vNums foram listados na saída do programa, permitindo uma visualização clara dos elementos que não estavam presentes no primeiro vetor ordenado.

e. Medição do Tempo de Execução: por fim, foi medido o tempo total de execução do programa, desde a geração dos vetores até a listagem dos números não encontrados. Essa medida foi feita utilizando a função clock() da biblioteca <time.h>, e o tempo foi exibido em segundos.

A saída do programa inclui os dois vetores gerados, os números que não foram encontrados em vNums e o tempo de execução do programa.

4. **Observações e conclusões**

Nesse programa, foi necessário pesquisar uma biblioteca de geração de números aleatórios, relembrar o conceito de busca binária, construir um algoritmo que ordenasse o vetor e, também, pesquisar uma biblioteca para calcular o tempo que o programa levou para rodar.

As bibliotecas novas (rand() e clock()) foram de fácil implementação através de guias/exemplos em sites conhecidos, como StackOverflow. No entanto, os algoritmos de busca binária e de ordenação do vetor se mostraram como o verdadeiro desafio do laboratório.

O processo levou em torno de 1.32 segundos, o que nos surpreendeu. Esperávamos que o computador fosse mais rápido em relação a isso, considerando ser um programa razoavelmente simples no qual foram implementados algoritmos eficientes.