



Universidade Federal de Uberlândia
Curso de Graduação em Gestão da Informação
5º Trabalho de Estruturas de Dados I – Prof. Daniel A. Furtado
Trabalho Individual – Comparando $O(n)$ com $O(\log n)$

INTRUÇÕES GERAIS – LER ATENTAMENTE

- Esta atividade deve ser realizada individualmente;
- Este trabalho deve ser feito utilizando a linguagem C# ou Python. Trabalhos implementados utilizando outras linguagens serão anulados;
- **Todo o código deve ser colocado em um único arquivo**, pois apenas um arquivo (.cs ou .py) deve ser entregue. Não crie arquivos adicionais para as classes (o único arquivo a ser entregue deve conter todas as definições de classes, assim como o programa principal).
- **Não compactar** os arquivos no momento da entrega;
- Os recursos adequados das linguagens devem ser utilizados;
- Esteja atento às **observações sobre plágio** apresentadas no final deste documento;
- Trabalhos com implementações utilizando trechos de códigos retirados de sites da Internet, de trabalhos de colegas de classe ou de semestres anteriores serão anulados;
- As implementações não devem conter qualquer conteúdo de caráter imoral, desrespeitoso, pornográfico, discurso de ódio, desacato, etc.;
- O trabalho deve ser entregue até a data definida pelo professor em aula síncrona pelo Sistema de Aplicação de Testes (SAAT);
- Trabalhos enviados por e-mail ou por outros meios digitais (como o MS Teams) **não serão considerados** (veja instruções no final);

1. Este trabalho utiliza como base a implementação de lista ligada em array solicitada anteriormente no Trabalho 3 (parte 2). Crie um novo projeto em sua IDE de desenvolvimento aproveitando o código do referido trabalho.
2. Acrescente um método na classe para verificar, sequencialmente, se a lista contém ou não um dado elemento. Utilize o código a seguir:

C#:

```
public bool Contains_LinearSearch_Sorted(int key)
{
    int k = 0;
    while (k < this.count && this.items[k] < key)
        k++;

    if (k == count)
        return false; // não encontrado
```

```

        return this.items[k] == key;
    }

```

Python:

```

def Contains_LinearSearch_Sorted(self, key):
    k = 0
    while (k < self.count && self.items[k] < key):
        k += 1

    if (k == self.count):
        return false

    return self.items[k] == key

```

3. Acrescente um método na classe para verificar, utilizando busca binária, se a lista contém ou não um dado elemento. Utilize o código a seguir:

C#:

```

public bool Contains_BinarySearch(int key)
{
    int min = 0;
    int max = this.count - 1;

    while (min <= max)
    {
        int mid = (min + max) / 2;
        if (key == this.items[mid])
            return true;
        else if (key < this.items[mid])
            max = mid - 1;
        else
            min = mid + 1;
    }

    return false;
}

```

Python:

```

def Contains_BinarySearch(self, key):

    min = 0
    max = self.count - 1

    while (min <= max):
        mid = (min + max) / 2
        if (key == this.items[mid]):
            return true
        else if (key < this.items[mid]):
            max = mid - 1
        else:
            min = mid + 1

    return false

```

4. No programa principal, vamos criar um código para gerar um *array* de inteiros contendo N elementos aleatórios no intervalo de 0 a MAX. Portanto, defina as constantes N e MAX com valores iniciais de 1.000 e 100.000, respectivamente.

5. No programa principal, insira o código para criar uma nova “ArrayList” utilizando o valor de **N** como sendo o tamanho inicial do *array*. Em seguida, insira o código para gerar **N** números aleatórios e inserir no final do *array* utilizando o método **AddLast**. Veja a seguir um exemplo de uso das classes das linguagens para gerar números aleatórios:

C#

```
Random random = new Random(); // criação de objeto para gerar nros aleatórios
int x = random.Next(0, MAX);
```

Python

```
import random

x = random.randint(0,MAX)
```

6. Ordenação do *array*. Os métodos de busca utilizados neste trabalho consideram que o *array* esteja ordenado. Neste trabalho vamos ordená-lo rapidamente utilizando algoritmos da própria linguagem. Para isso, crie um método de nome SortList na classe da lista que faça a chamada do respectivo método de ordenação da linguagem:

C#

```
public void SortList()
{
    Array.Sort(this.items, 0, this.count);
}
```

Python

```
def SortList(self):
    self.items.sort()
```

7. No programa principal, depois de inserir os números aleatórios na lista, acrescente o código para chamar o método de ordenação da lista.
8. No programa principal, acrescente o código para efetuar 200 buscas de números aleatórios no *array* utilizando o método de **busca binária**. O valor a ser procurado no *array* deve ser gerado aleatoriamente no intervalo de 0 a 2*M. Não é necessário mostrar nenhuma mensagem informando se o elemento foi encontrado ou não no *array*.
9. Acrescente um código para contabilizar o tempo necessário para efetuar as 200 buscas no *array*. Veja os exemplos a seguir:

C#

```
var watch1 = new System.Diagnostics.Stopwatch();
Console.WriteLine("Efetuando 200 buscas...");
watch1.Start();

// coloque aqui o código cujo tempo de execução será verificado

watch1.Stop();
Console.WriteLine("Tempo gasto nas buscas: " + watch1.Elapsed);
```

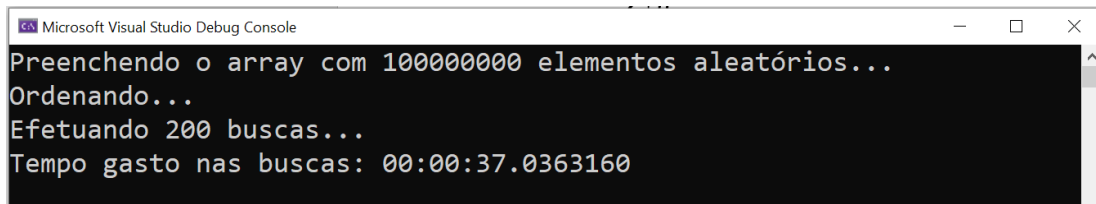
Python

```
print("Efetuando 200 buscas...")
start = datetime.datetime.now()

# coloque aqui o código cujo tempo de execução será verificado
```

```
end = datetime.datetime.now()
delta = end - start
print('Tempo gasto nas buscas: ', delta)
```

10. Mensagens informativa. Acrescente mensagens informativas no programa principal para que o usuário possa acompanhar as operações. Veja o exemplo a seguir:



```
Microsoft Visual Studio Debug Console
Preenchendo o array com 100000000 elementos aleatórios...
Ordenando...
Efetuando 200 buscas...
Tempo gasto nas buscas: 00:00:37.0363160
```

11. Variando os valores de N. Execute o programa principal alterando o valor de N para cada um dos valores a seguir:
- 1.000
 - 50.000
 - 100.000
 - 500.000
 - 1.000.000
 - 100.000.000
 - 500.000.000
 - 1.000.000.000

Para cada um dos valores, anote o tempo gasto e insira em uma planilha utilizando um software de sua disponibilidade.

12. Altere o método de busca utilizado no programa para o método de **busca linear** e repita a execução para os valores de N mencionados acima, copiando os tempos para a planilha.
13. Crie um gráfico comparativo na planilha utilizando os tempos de execução dos dois métodos de busca.
14. Estudar o código fornecido nos métodos de busca.

Entrega

Um único arquivo (.cs ou .py) contendo todas as definições de classes e também o programa principal deve ser enviado pelo Sistema de Aplicação de Testes (SAAT) até a data limite indicada pelo professor. Não envie arquivos compactados ou outros arquivos do Visual Studio que não sejam o arquivo de código principal.

Sobre Eventuais Plágios

Este é um trabalho individual. Os alunos envolvidos em qualquer tipo de plágio, total ou parcial, seja entre equipes ou de trabalhos de semestres anteriores ou de materiais disponíveis na Internet (exceto os materiais de aula disponibilizados pelo professor), serão

duramente penalizados (art. 196 do Regimento Geral da UFU). Todos os alunos envolvidos terão seus **trabalhos anulados** e receberão **nota zero**.