**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS**
Instituto de Ciências Exatas e Informática

| Disciplina | Curso | Turno |
|---|---|---|
| Algoritmos e Estruturas de Dados | Sistemas de Informação | Noite |

Professor
Kleber Jacques F. de Souza (klebersouza@pucminas.br)

# Lab 03 - Distância entre pontos

## Instruções

- O exercício prático deve ser entregue individualmente via SGA, na data e horário programado. Não serão aceitos trabalhos por e-mail e/ou fora do prazo.
- Deve ser entregue apenas os aquivos de código fonte (.cs), e arquivos de testes, se houver.
- Todo código deve ser comentado e identado.
- Plagio é crime! Trabalhos copiados serão anulados.

## Descrição

O objetivo desta atividade prática é exercitar os conceitos de Análise de Algoritmos, visto em sala de aula. Sua tarefa será criar e executar alguns algoritmos para avaliar a distância entre dois pontos e analisar a complexidade dos algoritmos implementados.

Imagine que você possua um vetor de N pontos e gostaria de extrair algumas informações desta lista de pontos. Cada ponto possui apenas duas informações: o valor de X e o valor de Y.

```
class Ponto {
  public int x { get; set; }
  public int y { get; set; }

  public Ponto(int _x, int _y){
    x = _x;
    y = _y;
  }
}
```

Dado uma vetor de pontos, você deve implementar duas funções que respondam as seguintes perguntas:

1. Dado um vetor de pontos, e um Ponto $P$ qualquer. Qual é o ponto do vetor que está mais próximo do Ponto $P$.
2. Dado um vetor de pontos, quais são os dois pontos mais próximos, ou seja, os dois pontos presentes no vetor que possuem a menor distância entre eles.

Para calcular a distância entre dois pontos, basta usar a Distância Euclidiana:

$$d = \sqrt{(P1_x - P2_x)^2 + (P1_y - P2_y)^2}$$

O vetor N de pontos deve ser preenchido de forma aleatória. Segue abaixo um exemplo de código:

```
// MÉTODO PARA GERAR UM VETOR DE PONTOS ALEATORIOS
public static Ponto[] getPontosAleatorios(int N) {
  // CRIA UM VETOR DE N PONTOS
  Ponto[] pontos = new Ponto[N];

  // PREENCHE O VETOR DE PONTOS DE FORMA ALEATÓRIA
  Random random = new Random();
  for (int i = 0; i < N; i++)
    pontos[i] = new Ponto(random.Next(0, N), random.Next(0, N));

  // RETORNA O VETOR DE PONTOS
  return pontos;
}
```

## Análise

Após implementar seus algoritmos você deve executar uma análise experimental e descobrir o custo computacional de cada um deles. Para cada algoritmo deverá ser analisado:

- **Tempo de processamento**: Você deverá calcular o tempo de processamento de cada algoritmo. Abaixo segue um exemplo de código de como calcular este valor:

```
var watch = System.Diagnostics.Stopwatch.StartNew();

// SEU CÓDIGO AQUI

watch.Stop();
var elapsedMs = watch.ElapsedMilliseconds / 1000.0;
Console.WriteLine("Tempo Gasto: " + elapsedMs + " segundos");
```

- **Espaço de processamento**: Você deverá calcular a quantidade de memória gasta por cada algoritmo. Abaixo segue um exemplo de código de como calcular este valor:

```
var ramUsage = System.Diagnostics.Process.GetCurrentProcess().
   PeakWorkingSet64;
var allocationInMB = ramUsage / (1024 * 1024);
Console.WriteLine("Memória utilizada: " + allocationInMB + "MB");
```

- **Função de Custo**: Deverá ser contabilizando o custo de execução das operações em relação ao valor N.

- **Complexidade usando a notação O**: A partir da função de custo determinar qual a classe de complexidade de cada algoritmo de acordo com a notação O.

A análise deverá ser executar para valores N de 1.000 até 100.000, variando 1.000 em 1.000. Deverão ser realizadas 5 medições e o tempo será dado pela média aritmética das 3 medições desconsiderando o maior e o menor valor de experimento.

Além do código deverá ser enviado a análise dos algoritmo contendo as informações levantadas e gráficos gerados.