Pontifícia Universidade Católica de Minas Gerais
Instituto de Ciências Exatas e Informática
Curso: Sistemas de Informação
Disciplina: Laboratório de Algoritmos e Estruturas de Dados
Professora: Joyce Christina de Paiva Carvalho

Aula 02 – Revisão ATP – Parte 02

Instruções:

- O exercício deve ser entregue individualmente via SGA.
- Devem ser entregues SOMENTE os arquivos de código fonte (.cs).
- Comentários e identação serão considerados na avaliação.
- Trabalhos copiados serão anulados.

Descrição

O objetivo deste laboratório 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 um 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, quis 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 memória:** 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 contabilizado o custo de execução das operações em relação ao valor de N.

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

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

Além do código, deverá ser enviado a análise dos algoritmos contendo as informações dos gráficos gerados.