Uma imagem com captura de ecrã, símbolo, Tipo de letra, Gráficos

Descrição gerada automaticamente

**ANÁLISE DE DESEMPENHO DE ALGORITMOS DE ORDENAÇÃO**

**Projeto – Grupo I**

Kássia Guimarães – A105755

Sara Marques – A85571

Vanessa Gomes – A105871

**Maio, 2024**

**Índice**

[1. Introdução 1](#_Toc167998675)

[2. Elaboração de DataSets 2](#_Toc167998676)

[2.1 Elaboração de valores – distribuição aleatória 2](#_Toc167998677)

[2.2 Elaboração de valores – distribuição de *Poisson* 2](#_Toc167998678)

[2.3 Elaboração de valores – distribuição binomial 2](#_Toc167998679)

[2.4 Elaboração de valores – distribuição geométrica 2](#_Toc167998680)

[3. Frequência de duplicados 3](#_Toc167998681)

[4. Análise Gráfica 3](#_Toc167998682)

[4.1 Distribuição Aleatória 4](#_Toc167998683)

[4.2 Distribuição de Poisson 4](#_Toc167998684)

[4.3 Distribuição Binomial 5](#_Toc167998685)

[4.4 Distribuição Geométrica 6](#_Toc167998686)

[5. Conclusões e Limitações 7](#_Toc167998687)

# Introdução

Este projeto pretende realizar um estudo de desempenho e complexidade temporal dos seguintes algoritmos de ordenação, como demonstrado em sala de aula: *selectionSort, insertionSort, shellSort, quickSort* (várias versões) e *countingSort*.

Para tal, foi criado um conjunto de dados (*DataSet*) com base nos modelos de distribuição de *Poisson*, *binomial*, geométrica e aleatória. Estes modelos foram selecionados para avaliar como diferentes distribuições de dados afetam o desempenho dos algoritmos de ordenação. Para cada modelo de distribuição, foram geradas 20 listas de inteiros, cada uma com 750 valores.

O objetivo desta análise foi investigar o tempo de execução dos algoritmos de ordenação em diferentes cenários. Com vistas a alcançar este objetivo, o tempo de execução foi analisado considerando *arrays* com tamanhos variáveis, de 0 a 749, em incrementos de 50, considerando a média como medida de tendência central. Este método permitiu observar como os algoritmos se comportam em relação a diferentes tamanhos de conjuntos de dados e distribuições, proporcionando uma visão abrangente sobre a eficiência e a complexidade dos mesmos.

Os dados derivados das distribuições de *Poisson*, binomial, geométrica e aleatória fornecem um amplo espectro de cenários, desde distribuições com alta variabilidade até distribuições com valores mais concentrados. Esta variedade é crucial para compreender melhor as características de desempenho dos algoritmos de ordenação em situações práticas.

# Elaboração de DataSets

Neste capítulo serão abordados os códigos utilizados para a criação das listas referentes a cada tipo de distribuição discutidas no capítulo 1.

## Elaboração de valores – distribuição aleatória

Código utilizado para a criação das 10 (dez) listas com valores aleatórios.

for j in range(1, length + 1):

            V += [random.randrange(1, 101)]

        new\_row = pd.DataFrame({'Index': i, 'Distribution': 'Random', 'Sample': [V]})

        samples = pd.concat([samples, new\_row], ignore\_index=True)

## Elaboração de valores – distribuição de *Poisson*

Código utilizado para a criação do array com valores provenientes do modelo de *Poisson*.

for j in range(1, length + 1):

            V += [np.random.poisson(100)]

        new\_row = pd.DataFrame({'Index':i, 'Distribution':'Poisson', 'Sample': [V]})

        samples = pd.concat([samples, new\_row], ignore\_index=True)

## Elaboração de valores – distribuição binomial

Código para a elaboração da lista com inteiros provenientes do modelo binomial.

for j in range(1, length+1):

            V += [np.random.binomial(200, 0.5)]

        new\_row = pd.DataFrame({'Index':i, 'Distribution':'Binomial', 'Sample': [V]})

        samples = pd.concat([samples, new\_row], ignore\_index=True)

## Elaboração de valores – distribuição geométrica

Código para a elaboração do array com inteiros provenientes da distribuição geométrica.

for j in range(1, length+1):

            V += [np.random.geometric(0.01)]

        new\_row = pd.DataFrame({'Index':i, 'Distribution':'Geometric', 'Sample': [V]})

        samples = pd.concat([samples, new\_row], ignore\_index=True)

# Frequência de duplicados

A screen shot of a computer code

Description automatically generatedRelativamente a frequência de duplicados, criou-se uma função para verificar a quantidade de duplicados de cada distribuição.

Verificou-se que a distribuição binomial apresentava o maior número de duplicados (cerca de 709), enquanto a distribuição geométrica apresentou o menor número de duplicados (cerca de 493).

É percetível os resultados em termos de frequência, visto que o modelo binomial e o modelo de Poisson, quanto maior a amostra mais se aproxima de um modelo de distribuição normal, em que os valores apresentam pouca variância em relação à média.

Já na distribuição geométrica, são gerados valores que representam o número de tentativas até o primeiro sucesso – com uma probabilidade de sucesso muito pequena, há uma grande variedade de possíveis valores e à medida que se afasta, a probabilidade de gerar números maiores decresce em termos exponenciais.

# Análise Gráfica

Neste capítulo pretende-se apresentar um conjunto de gráficos que demonstram o desempenho de cada algoritmo ao longo de um conjunto de dados crescente em número de elementos, para cada um dos *DataSets* elaborados no capítulo 2.

Cada valor de desempenho temporal por conjunto de dados foi obtido através da média sobre uma série de 20 execuções.

Foi realizada a análise gráfica global dos algoritmos para cada tipo de distribuição.

## Distribuição Aleatória

Uma imagem com texto, captura de ecrã, Gráfico, diagrama

Descrição gerada automaticamente

A graph with colorful lines and dots

Description automatically generated A graph with blue and orange dots

Description automatically generated

## Distribuição de Poisson

Uma imagem com texto, captura de ecrã, Gráfico, file

Descrição gerada automaticamente

Uma imagem com texto, captura de ecrã, diagrama, file

Descrição gerada automaticamente Uma imagem com texto, captura de ecrã, Gráfico, file

Descrição gerada automaticamente

## Distribuição Binomial

Uma imagem com texto, captura de ecrã, diagrama, file

Descrição gerada automaticamente

Uma imagem com texto, captura de ecrã, diagrama, Gráfico

Descrição gerada automaticamente Uma imagem com texto, captura de ecrã, diagrama, file

Descrição gerada automaticamente

## Distribuição Geométrica

Uma imagem com texto, captura de ecrã, Gráfico, file

Descrição gerada automaticamente

Uma imagem com texto, captura de ecrã, diagrama, file

Descrição gerada automaticamente Uma imagem com texto, captura de ecrã, Gráfico, file

Descrição gerada automaticamente

# Conclusões e Limitações

Pode-se observar que para todas as distribuições estudadas os algoritmos *selectionSort* e *insertionSort* apresentam crescimento quadrático, enquanto o algoritmo *countingSort* apresenta um melhor comportamento, quase constante.

Para além disso, nota-se que nas distribuições com maior frequência de duplicados (*Poisson* e Binomial), o *quickSort* aparenta comportamento quase quadrático. Por esta razão é possível deduzir que este algoritmo é muito sensível aos duplicados.

Analisando o código do *quickSort3*, nota-se que utiliza a partição de três vias. Esta abordagem ajuda a lidar eficientemente com duplicados, garantindo que todos os elementos iguais ao pivô estejam agrupados, resultando numa melhor divisão e menor complexidade. Ao averiguar a diferença de tempo entre o *quickSort* e o *quickSort3*, observa-se que, na análise do tempo de execução com números inteiros provenientes da distribuição binomial – distribuição que possui um maior número de valores duplicados, o *quickSort3* apresenta uma melhor performance, quase linear.

Uma limitação do projeto foi criar *DataSets* de inteiros para os modelos: Gaussiano, Uniformes, Beta e Exponencial, por exemplo, visto que estes *arrays* apresentaram valores em *float number*, e arredondar seria enviesar a análise de dados das distribuições correspondentes.