Relatório

1 - Descrição do problema:

Neste relatório, abordamos o **Problema da Cobertura de Vértices** um dos desafios clássicos na teoria dos grafos e na ciência da computação. O problema consiste em encontrar o **menor conjunto possível de vértices** em um grafo tal que **todas as arestas** do grafo sejam "cobertas". Uma aresta é considerada coberta se pelo menos um de seus vértices extremos pertence ao conjunto selecionado.

O Problema da Cobertura de Vértices é classificado como **NP-difícil**, o que significa que não se conhece um algoritmo eficiente (de tempo polinomial) para resolvê-lo em todos os casos. Por essa razão, a resolução exata do problema para grafos grandes ou densos torna-se inviável na prática, especialmente em aplicações que exigem respostas rápidas ou lidam com volumes massivos de dados.

Diante dessa complexidade, este relatório foca em abordagens heurísticas para encontrar soluções aproximadas. Essas heurísticas não garantem a solução ótima, mas são capazes de fornecer conjuntos de vértices que cobrem todas as arestas em um tempo computacional viável, mesmo para instâncias grandes do problema. As estratégias heurísticas exploradas incluem algoritmos gulosos, randomizados e reativos, que buscam equilibrar a qualidade da solução e o tempo de execução.

2 - Descrição das instâncias:

Foram buscadas na literatura instâncias de grafos com diferentes tamanhos e densidades. Utilizamos grafos de 5000 a 10000 Vértices, variando a quantidade de arestas para avaliar o impacto da densidade na eficiência dos algoritmos.

- **Instância 1:**
  - Referência: https://snap.stanford.edu/data/wiki-Vote.html
  - Tema: Redes Sociais
  - O Grafo representa a rede de votação nas eleições para administradores da Wikipédia até janeiro de 2008. Os vértices representam usuários da Wikipédia, e as arestas direcionadas representam votos dados em eleições para administrador.

  - Grau: 756
  - Ordem: 7.115
  - Grafo Direcionado: Sim
  - Número de Componentes Conexas: 5010
  - Vértices Ponderados: Não
  - Arestas Ponderadas: Não
  - Grafo Completo: Não
  - Quantidade de Arestas: 103689

- **Instância 2:**
  - Referência: https://networkrepository.com/ca-Erdos992.php
  - Tema: Redes de Colaboradores
  - O Grafo representa a rede de colaboração acadêmica de Paul Erdős até 1992. Os vértices representam pesquisadores que colaboraram em publicações científicas, e as arestas e representam coautorias entre pesquisadores.

  - Grau: 61
  - Ordem: 6100
  - Grafo Direcionado: Não
  - Número de Componentes Conexas: 1023
  - Vértices Ponderados: Não
  - Arestas Ponderadas: Não
  - Grafo Completo: Não
  - Quantidade de Arestas: 7515

- **Instância 3:**
  - Referência: https://snap.stanford.edu/data/p2p-Gnutella08.html
  - Tema: Redes ponto a ponto da Internet
  - Uma sequência de instantâneos da rede de compartilhamento de arquivos ponto a ponto Gnutella de agosto de 2002.
  - Grau: 48
  - Ordem: 6301
  - Grafo Direcionado: Sim
  - Número de Componentes Conexas: 81
  - Vértices Ponderados: Não
  - Arestas Ponderadas: Não
  - Grafo Completo: Não
  - Quantidade de Arestas: 39994

- **Instância 4:**
  - Referência: https://networkrepository.com/ia-reality.php
  - Tema: Redes de Interação
  - Dados de rede do experimento Reality Mining (2004) no MIT, com 100 telefones monitorando chamadas e correios de voz entre usuários ao longo de 9 meses. Nós representam pessoas e arestas, interações telefônicas.
  - Grau: 261
  - Ordem: 6100
  - Grafo Direcionado: Não
  - Número de Componentes Conexas: 2
  - Vértices Ponderados: Não
  - Arestas Ponderadas: Não
  - Grafo Completo: Não
  - Quantidade de Arestas: 7680

- **Instância 5:**

  - Referência: https://networkrepository.com/socfb-UChicago30.php
  - Tema: Analise de conexão de usuários em uma rede social universitária
  - Grau: 1621
  - Ordem: 6591
  - Grafo Direcionado: Não
  - Número de Componentes Conexas: 16
  - Vértices Ponderados: Nao
  - Arestas Ponderadas: Nao
  - Grafo Completo: Não
  - Quantidade de Arestas: 208.100

- **Instância 6:**
  - Referência: https://networkrepository.com/bio-dmela.php
  - Tema: Uma rede biológica onde os nós representam proteínas e as arestas representam interações entre proteínas.
  - Grau: 190
  - Ordem: 7393
  - Grafo Direcionado: Sim
  - Número de Componentes Conexas: 1
  - Vértices Ponderados: Não
  - Arestas Ponderadas: Não
  - Grafo Completo: Não
  - Quantidade de Arestas: 25.600

- **Instância 7:**
  - Referência: https://networkrepository.com/tech-WHOIS.php
  - Tema: Análise de Conectividade
  - O grafo tech-WHOIS é um grafo de rede de tecnologia que representa a conectividade entre diferentes entidades na base de dados WHOIS. A base de dados WHOIS é usada para armazenar informações sobre a propriedade e a administração de nomes de domínio na internet.
  - Grau: 1079
  - Ordem: 7476
  - Grafo Direcionado: Não
  - Número de Componentes Conexas: 1
  - Vértices Ponderados: Não
  - Arestas Ponderadas: Não
  - Grafo Completo: Não
  - Quantidade de Arestas: 56943

- **Instância 8:**
  - Referência: https://networkrepository.com/socfb-American75.php
  - Tema: Redes esparsas
  - Uma rede social de amizade extraída do Facebook, composta por pessoas (nós) com arestas representando laços de amizade.
  - Grau: 261
  - Ordem: 6809
  - Grafo Direcionado: não
  - Número de Componentes Conexas: 9
  - Vértices Ponderados: não
  - Arestas Ponderadas: não
  - Grafo Completo: não
  - Quantidade de Arestas: 217662

- **Instância 9:**
  - Referência: https://snap.stanford.edu/data/ca-GrQc.html
  - Tema: Redes de Coautoría Científica (Físicos no arXiv)
  - O Grafo representa as colaborações entre físicos que publicaram artigos no arXiv, com os vértices representando os autores e as arestas representando a coautoria entre os autores.
  - Grau: 17
  - Ordem: 5.242
  - Grafo Direcionado: Não
  - Número de Componentes Conexas: 4.808
  - Vértices Ponderados: Não
  - Arestas Ponderadas: Não
  - Grafo Completo: Não
  - Quantidade de Arestas:14.496

- **Instância 10:**
  - Referência: https://networkrepository.com/web-indochina-2004.php
  - Tema: Web e Conectividade
  - O Grafo representa a estrutura de links entre páginas web da região da Indochina, coletadas em 2004. Os vértices representam páginas da web, e as arestas representam links entre essas páginas.
  - Grau: 144
  - Ordem: 11.400
  - Grafo Direcionado: Sim
  - Número de Componentes Conexas:11.400
  - Vértices Ponderados: Não
  - Arestas Ponderadas: Não
  - Grafo Completo: Não
  - Quantidade de Arestas:47.600

3 - Descrição dos métodos implementados:

Implementamos três algoritmos gulosos para resolver o problema da Cobertura de Vértices com Matriz e com Lista. Os algoritmos seguem a estratégia de selecionar o melhor vértice considerando a soma dos graus de seus vizinhos, priorizando aqueles com maior influência na cobertura das arestas.

- **Algoritmo Guloso - Matriz:**
  Nesta implementação, o grafo é representado por uma matriz de adjacência. Inicialmente, são alocados vetores dinâmicos para armazenar:

  - O status dos vértices escolhidos;
  - Se a aresta X está ou não coberta
  - Os graus de cada vértice;

  E, para otimização, as listas de vizinhos de cada vértice (construídas a partir da matriz).
  A utilização das listas de vizinhos permite iterar somente sobre os vizinhos reais de um vértice, evitando varreduras desnecessárias na matriz completa. Em cada iteração, uma função lambda que calcula a soma dos graus dos vizinhos não cobertos para cada vértice. O algoritmo seleciona o vértice que maximiza essa soma, com critérios de desempate baseados também respectivamente no grau do próprio vértice e no seu índice. Após a escolha, as arestas incidentes ao vértice selecionado são marcadas como cobertas e o grau dos vizinhos é atualizado. O processo se repete até que todas as arestas estejam cobertas. Ao final, o algoritmo exibe o conjunto de vértices escolhidos, a quantidade de vértices na solução e o número total de arestas cobertas.

- **Algoritmo Guloso - Lista:**
  Nesta versão, o grafo é representado por meio de listas de adjacência, onde cada vértice armazena uma lista de suas arestas. O algoritmo segue uma abordagem gulosa para resolver o problema da Cobertura de Vértices, selecionando iterativamente o vértice que maximiza a soma dos graus dos vizinhos ainda não cobertos. Em caso de empate, o algoritmo prioriza o vértice com maior grau e, em seguida, o de menor índice.

  Estruturas utilizadas:
  1. Listas de Adjacência: Armazenam as conexões do grafo de forma compacta.
  2. Vetores auxiliares: verticeEscolhido, arestaCoberta, graus e somaVizinhos são usados para rastrear o estado do algoritmo.

Funcionamento do Algoritmo:

1. Inicialização: Calcula o grau de cada vértice e a soma dos graus dos vizinhos.
2. Seleção Gulosa: Escolhe o vértice que maximiza a soma dos graus dos vizinhos não cobertos.
3. Atualização: Marca o vértice como escolhido, cobre suas arestas e atualiza os graus dos vizinhos.
4. Condição de Parada: O algoritmo para quando todas as arestas são cobertas.
5. Resultados: Exibe a quantidade de vértices na solução, arestas cobertas e tempo de execução.

- **Algoritmo Guloso Randomizado - Matriz:**
  Nesta versão, o grafo é armazenado em uma matriz de adjacência, permitindo um acesso direto às conexões entre os vértices. O algoritmo implementa uma estratégia gulosa randomizada para encontrar uma cobertura de vértices eficiente.

  Para isso, são utilizados:

  - Uma matriz booleana para rastrear as arestas já cobertas.
  - Vetores booleanos para marcar os vértices escolhidos na solução.
  - Vetores inteiros para armazenar os graus dos vértices e a lista de vizinhos de cada um.

  A seleção do próximo vértice a ser adicionado à cobertura segue a seguinte estratégia:

  1. Cálculo da soma dos graus dos vizinhos não cobertos: Para cada vértice não escolhido, é calculada a soma dos graus dos seus vizinhos ainda não cobertos, o que reflete a "importância" de cada vértice com base nos seus vizinhos.
  2. Seleção baseada em α: A fração α, que varia entre 0.3 e 0.7, serve para selecionar quais vértices serão incluídos na lista restrita. Quando α está mais próximo de 0.3, mais vértices são considerados, promovendo maior diversidade e permitindo mais exploração. Quando α se aproxima de 0.7, o critério de seleção é mais restrito, favorecendo uma abordagem mais gulosa, com menos aleatoriedade.
  3. Seleção aleatória do vértice: Após filtrar os vértices com base no valor de α, o vértice final é escolhido aleatoriamente entre os candidatos restantes, o que adiciona um elemento de aleatoriedade à decisão.

Após a escolha, todas as arestas incidentes ao vértice selecionado são marcadas como cobertas, e os graus dos vértices vizinhos são atualizados. Esse processo se repete até que todas as arestas do grafo sejam cobertas.

Ao final, o algoritmo exibe a quantidade de vértices na solução e o tempo total de execução. O uso da abordagem gulosa randomizada equilibra eficiência e variabilidade, permitindo encontrar soluções próximas do ótimo em diferentes execuções.

- **Algoritmo Guloso Randomizado - Lista:**
  Nesta versão, o grafo é armazenado por meio de listas de adjacência, permitindo um acesso eficiente aos vizinhos de cada vértice. O algoritmo segue uma estratégia gulosa randomizada para encontrar uma solução aproximada para o problema da cobertura de vértices.

  Para isso, são utilizados:

  - Listas encadeadas para representar a conectividade do grafo de forma compacta.
  - Vetores booleanos para marcar os vértices escolhidos e as arestas cobertas.
  - Vetores inteiros para armazenar o grau de cada vértice e a soma dos graus de seus vizinhos.

  A seleção do próximo vértice a ser adicionado à cobertura segue a seguinte estratégia:

  1. Calcula-se a soma dos graus dos vizinhos para cada vértice ainda não escolhido.
  2. Os vértices candidatos são aqueles com a maior soma de graus dos vizinhos, sendo considerado também o grau do próprio vértice como critério de desempate.
  3. Em vez de escolher sempre o melhor candidato, o algoritmo adota uma abordagem randomizada, selecionando aleatoriamente dentro de uma "janela" contendo aproximadamente metade dos melhores candidatos.

  Após a escolha de um vértice, todas as arestas incidentes a ele são marcadas como cobertas, e os graus de seus vizinhos são atualizados. Esse processo se repete até que todas as arestas do grafo sejam cobertas.

  Ao final, o algoritmo exibe a quantidade de vértices na solução e o tempo total de execução. A introdução de aleatoriedade permite explorar diferentes soluções em execuções distintas, equilibrando desempenho e qualidade da solução encontrada.

- **Algoritmo Guloso Randomizado Reativo - Matriz:**
  Nesta implementação, o grafo é representado por uma matriz de adjacência. Inicialmente, são alocados vetores dinâmicos para armazenar o status dos vértices escolhidos (se foram ou não selecionados para a solução).
  Se a aresta X está ou não coberta, os graus de cada vértice (número de arestas incidentes).
  O algoritmo é executado em uma única iteração global para encontrar a melhor solução possível. Em cada iteração, são inicializadas as estruturas de dados para armazenar a solução atual.

  Funcionamento do Algoritmo
  - Inicialização:
    São alocados e inicializados vetores dinâmicos para:
    verticeEscolhido: Armazena o status de cada vértice (inicialmente, todos são marcados como não escolhidos).
    arestaCoberta: Matriz que indica se uma aresta entre dois vértices está coberta (inicialmente, todas as arestas são marcadas como não cobertas).
    graus: Armazena o grau de cada vértice, calculado com base na matriz de adjacência.
    vizinhos: Armazena a lista de vizinhos de cada vértice, preenchida com base na matriz de adjacência.
    O número total de arestas no grafo é obtido a partir de numArestasGrafo.

  - **Cálculo dos Graus e Lista de Vizinhos:**
    Para cada vértice, o algoritmo calcula o grau iterando sobre a matriz de adjacência.
    A lista de vizinhos de cada vértice é preenchida com base nas conexões presentes na matriz de adjacência.

  - **Loop Principal do Algoritmo:**
    O algoritmo executa um loop até que todas as arestas sejam cobertas.

  - **Em cada iteração:**
    Lista de Candidatos: Identifica todos os vértices não escolhidos e calcula a soma dos graus dos seus vizinhos não cobertos.
    Lista Restrita de Candidatos: Filtra os vértices candidatos com base no parâmetro alpha. Apenas os vértices com soma de graus maior ou igual a alpha * melhorSoma são considerados.
    Escolha Aleatória: Um vértice é escolhido aleatoriamente da lista restrita.
    Atualização da Solução: O vértice escolhido é marcado como parte da solução, e suas arestas incidentes são marcadas como cobertas.
    Atualização dos Graus: Os graus dos vizinhos são atualizados, e o contador de arestas cobertas é incrementado.

- **Finalização:**
  A memória alocada dinamicamente é liberada.
  O número de vértices na solução é retornado.

- **Algoritmo Guloso Randomizado Reativo - Lista:**
  Nesta implementação, o grafo é representado por uma lista de adjacência. Inicialmente, são alocados vetores dinâmicos para armazenar:
  - O status dos vértices escolhidos;
  - Se a aresta X está ou não coberta;
  - Os graus de cada vértice;
  - A soma dos graus dos vizinhos de cada vértice.

  O algoritmo é executado em uma única iteração global para encontrar a melhor solução possível. Em cada iteração, são inicializadas as estruturas de dados para armazenar a solução atual.

  Para cada vértice, o algoritmo calcula os graus dos vértices e a soma dos graus dos vizinhos. Isso é feito iterando sobre todas as arestas de cada vértice.
  O algoritmo executa um loop de iterações internas até que todas as arestas sejam cobertas. A cada 10 iterações, as probabilidades dos parâmetros alphas são ajustadas com base no desempenho observado.

  Um alpha é escolhido com base nas probabilidades ajustadas. Esse alpha é usado para selecionar vértices candidatos.

  Os vértices candidatos são selecionados com base no alpha escolhido.

  O algoritmo coleta os vértices que têm a maior soma dos graus dos vizinhos não cobertos.

  Entre os vértices candidatos, um vértice é escolhido aleatoriamente. Esse vértice é marcado como escolhido e as arestas incidentes a ele são marcadas como cobertas. Os graus dos vizinhos são atualizados.

  O desempenho do alpha escolhido é atualizado com base na quantidade de vértices escolhidos.
  Ao final da iteração, o algoritmo imprime a solução encontrada.

4 - Análise dos Algoritmos:

A análise de desempenho de algoritmos para o problema de cobertura de vértices em grafos foi realizada utilizando duas representações distintas: lista de adjacência e matriz de adjacência. O objetivo do estudo é comparar o tempo de execução de diferentes abordagens gulosa, randomizada e reativa em ambas as representações.

Para garantir uma avaliação estatisticamente significativa, cada uma das 10 instâncias descritas neste relatório foi executada 100 vezes. Dessa forma, o tempo médio de execução foi calculado para cada algoritmo em cada instância.

Os passos seguidos na coleta de dados foram:

1. **Execução dos algoritmos**: Foram considerados os três algoritmos (guloso, randomizado e reativo), os quais foram rodados tanto na representação por lista quanto por matriz de adjacência.
2. **Repetição das execuções**: Para cada instância, cada algoritmo foi executado 100 vezes. Para isso, implementamos uma função que automatiza esse processo, garantindo uniformidade na coleta dos dados.
3. **Medição do tempo**: O tempo de execução foi calculado utilizando a função *clock()* da biblioteca *<ctime>.* Esse tempo foi registrado para cada uma das 100 execuções e, ao final, calculamos a média para obter um valor representativo.
4. **Armazenamento e organização**: Os dados foram compilados na *tabela 1* abaixo, organizando os tempos médios de execução de cada algoritmo por instância e representação do grafo.
5. **Registro das soluções obtidas**: Além do tempo de execução, armazenamos a quantidade média de vértices na solução e a melhor solução obtida entre as 100 execuções. Esses valores foram organizados e apresentados nas *tabelas 2 e 3.*

Tabela 1 - Tempo médio (em segundos) da execução de cada algoritmo nas 10 instâncias

Instâncias	Matriz			Lista		
	Guloso	Randomizado	Reativo	Guloso	Randomizado	Reativo
1	0.74714	0.84692	8.15844	1.22448	1.24832	1.24675
2	0.16692	0.23706	2.18248	0.41335	0.429	0.43179
3	0.28678	0.39666	3.82867	0.37209	0.37651	0.37384
4	0.14828	0.22548	2.31424	0.53082	0.53425	0.53748
5	11,507	11,854	126,971	15.792	15.479	14.967
6	1,5003	1,6854	17,8664	2,6986	2,7019	2,6929
7	2.27225	2.50238	27.7516	5.74084	5.72293	5.73762
8	14.009	18.282	212.33	14.9083	14.8864	14.8566
9	0.187	0.165	1.685	0.111	0.097	0.1
10	2.937	3.237	49.402	1.832	1.935	1.883

Tabela 2 - Quantidade média de vértices na solução e a melhor solução de 100 execuções de cada algoritmo nas 10 instâncias no Grafo em Matriz

Instâncias	Matriz					
	Guloso		Randomizado		Reativo	
	Solução média	Melhor Solução	Solução média	Melhor Solução	Solução média	Melhor Solução
1	5352	5352	7114	7109	7112	7107
2	738	738	1650	1495	1582	1460
3	2465	2465	6294	6270	6280	6257
4	289	289	1883	1382	1607	1214
5	5002	5002	5226	5193	5197	6189

6	3018	3018	3512	3429	3484	3423
7	2489	2489	2970	2836	2890	2838
8	5044	5044	5224	5216	5212	5207
9	280	280	323	310	315	308
10	9826	9826	11399	11397	11397	11392

Tabela 3 - Quantidade média de vértices na solução e a melhor solução de 100 execuções de cada algoritmo nas 10 instâncias no Grafo em Lista

Instâncias	**Lista**					
	Guloso		**Randomizado**		**Reativo**	
	Solução média	**Melhor Solução**	**Solução média**	**Melhor Solução**	**Solução média**	**Melhor Solução**
1	5190	5190	5183	5178	5183	5178
2	4342	4342	4522	4393	4531	4410
3	2369	2369	2374	2364	2374	2364
4	4062	4062	4152	4107	4151	4106
5	6553	6553	6563	6557	6563	6559
6	6884	6884	7063	6987	7062	6959
7	7406	7406	7420	7409	7419	7410
8	6368	6368	6373	3669	6372	6368
9	335	335	391	383	390	382
10	5736	5736	5495	5479	5496	5478

4.1 - Análise do tempo de execução entre lista e matriz:

Em geral, a **Matriz de Adjacência** foi mais rápida na **maioria dos casos**, exceto para o algoritmo Reativo. Isso ocorre porque:

1. **Acesso Direto**: A Matriz permite acesso direto a qualquer aresta em O(1), o que é vantajoso em operações que requerem verificações frequentes de adjacência. Por exemplo, na Instância 1 (wiki-Vote), o algoritmo Guloso com Matriz teve um tempo de 0.74714s, enquanto com Lista foi 1.22448s.
2. **Custo de Iteração**: Em grafos densos, onde a maioria das entradas da matriz é preenchida, a Matriz é eficiente porque não há overhead de estruturas de ponteiros ou listas encadeadas, como ocorre na Lista de Adjacência. Isso é evidente na Instância 5 (socfb-UChicago30), onde o algoritmo Guloso com Matriz teve um tempo de 11.507s, enquanto com Lista foi 15.792s.
3. **Localidade de Memória**: A Matriz armazena os dados de forma contígua na memória, o que melhora o desempenho devido à melhor utilização da cache da CPU.

No entanto, a **Lista de Adjacência** foi mais eficiente apenas para o algoritmo **Reativo**. Isso ocorre porque:

1. **Custo de Atualização**: O algoritmo Reativo requer atualizações frequentes nos graus dos vértices e na cobertura das arestas. Na Matriz, essas atualizações envolvem percorrer linhas e colunas inteiras, o que é custoso em grafos grandes e densos. Por exemplo, na Instância 5, o tempo do algoritmo Reativo com Matriz foi 126.971s, enquanto com Lista foi 14.967s.
2. **Complexidade de Operações**: O algoritmo Reativo realiza operações complexas, como ajustes dinâmicos de parâmetros (α) e recálculos de probabilidades, que são mais eficientes na Lista devido à sua estrutura compacta e acesso direto aos vizinhos.

Desempenho dos Algoritmos:

- **Algoritmo Guloso:** Apresenta os menores tempos de execução em ambas as representações. Isso ocorre porque ele segue uma abordagem determinística, sem a necessidade de randomização ou ajustes dinâmicos de parâmetros. Por exemplo, na Instância 2 (ca-Erdos992), o tempo do algoritmo Guloso com Matriz foi 0.16692s, enquanto com Lista foi 0.41335s.
- **Algoritmo Randomizado:** Tem um tempo de execução ligeiramente maior que o Guloso, devido à introdução de aleatoriedade na seleção dos vértices. No entanto, essa abordagem pode explorar soluções mais diversificadas. Por exemplo, na Instância 3 (p2p-Gnutella08), o

tempo do algoritmo Randomizado com Matriz foi 0.39666s, enquanto com Lista foi 0.37651s.

- **Algoritmo Reativo:** Apresenta os maiores tempos de execução, especialmente na representação por Matriz. Isso se deve ao custo adicional de ajustar dinamicamente os parâmetros (α) e recalcular as probabilidades durante a execução. Por exemplo, na Instância 8 (socfb-American75), o tempo do algoritmo Reativo com Matriz foi 212.33s, enquanto com Lista foi 14.8566s.

5 - Análise de resultado com teste de hipótese entre os métodos:

Para verificar a diferença entre os métodos utilizados na resolução do Problema da Cobertura de Vértices, aplicamos um **Teste t de Student para amostras pareadas**. O objetivo foi determinar se há diferenças estatisticamente significativas entre os resultados obtidos para os algoritmos Guloso, Randomizado e Reativo em suas respectivas representações por Matriz e Lista de Adjacência.

Definição das Hipóteses

- **Hipótese nula (H_0)**: Não há diferença significativa entre os métodos.
- **Hipótese alternativa (H_1)**: Existe diferença significativa entre os métodos.

Nível de Significância

Adotamos um nível de significância de **α = 0,05 (5%)**, que é um valor padrão para testes estatísticos.

Resultados do Teste t

Os valores obtidos para cada algoritmo foram:

- **Algoritmo Guloso**: t = -3.41, p = 0.0112
- **Algoritmo Randomizado**: t = -1.22, p = 0.2615
- **Algoritmo Reativo**: t = -1.27, p = 0.2446

Análise dos Resultados

- Para o **algoritmo Guloso**, o valor de **p = 0.0112** é menor que o nível de significância (**α = 0,05**), indicando que há uma diferença estatisticamente significativa entre as soluções obtidas com Matriz e Lista. Isso sugere que a Lista de Adjacência tende a produzir soluções melhores.
- Para os **algoritmos Randomizado e Reativo**, os valores de **p = 0.2615 e p = 0.2446** são maiores que **α = 0,05**, indicando que não há evidências suficientes para afirmar que existe uma diferença significativa entre os resultados das representações.

6 - Conclusões:

Os experimentos realizados neste estudo demonstraram que a escolha da estrutura de dados influencia significativamente o desempenho dos algoritmos aplicados ao Problema da Cobertura de Vértices. O **teste de hipótese confirmou que a Lista de Adjacência proporciona soluções melhores para o algoritmo Guloso**, enquanto para os algoritmos Randomizado e Reativo, a diferença entre as representações não foi estatisticamente significativa.

Os resultados também mostram que a **Matriz de Adjacência tende a ser mais eficiente em termos de tempo de execução**, especialmente para os algoritmos Guloso e Randomizado, devido ao acesso direto às arestas. No entanto, a **Lista de Adjacência apresentou um melhor desempenho para o algoritmo Reativo**, pois permite atualizações mais ágeis nos graus dos vértices e na cobertura das arestas.

Além disso, a análise estatística revelou que a escolha da estrutura de dados deve levar em consideração as características do grafo. Em grafos densos, o impacto da estrutura escolhida é reduzido, enquanto em grafos esparsos a Lista de Adjacência demonstrou um desempenho superior na obtenção de soluções menores.

Portanto, a escolha da representação ideal dependerá do contexto da aplicação. Se a prioridade for a rapidez e a simplicidade, o **Guloso em Matriz** é uma opção eficiente. Entretanto, para obter soluções de melhor qualidade, a **Lista de Adjacência é mais adequada**, especialmente em abordagens Randomizadas e Reativas.