PROJETO FINAL

ATENÇÃO: Descrever as soluções com o máximo de detalhes possível, no caso de programas, inclusive a forma como os testes foram feitos. Todos os artefatos (relatório, código fonte de programas, e outros) gerados para este trabalho devem ser adicionados em um repositório no site `github.com`, com o seguinte formato:

`Aluno1Aluno2_FinalProject_AA_RR_2024.`

- ✔ Somente é necessário a criação de um repositório por equipe.
- ✔ Para todas as DESCRIÇÕES (projetos) devem ser apresentados:
  - Uma documentação no formato de um relatório (no mínimo 4 páginas), seguindo o padrão de artigo da IEEE, disponivel em:
    `https://journals.ieeeauthorcenter.ieee.org/create-your-ieee-journal-article/authoring-tools-and-templates/tools-for-ieee-authors/ieee-article-templates/`
  - No relatório deve conter o a URL no repositório no site `github.com`
  - O relatório deve ser também enviando pelo SIGAA, no Tópico de Aula **Apresentação do Projeto Final – Parte 1 - Aula Extra [Reposição (devido à greve)] (23/09/2024 - 23/09/2024)** .
  - Apresente e descreva uma aplicação prática para o problema abordado.
  - Apresente uma análise de complexidade para cada algoritmo abordado no projeto.
  - Caso o problema abordo seja um NP-Completo, descreva e apresente o algoritmo exato.
- ✔ Na data da entrega, será realizado um seminário para a apresentação do projeto final. Vale ressaltar também que os programas devem ser codificados na linguagem C ou C++.

[DESCRIÇÃO - 1] BST and Heap : Huffman coding and decoding

A codificação Huffman é um dos algoritmos mais simples para compactar dados. Embora, seja muito antigo e simples, ainda é amplamente utilizado (por exemplo: em alguns estágios de JPEG, MPEG, etc.). Neste projeto, você implementará a codificação e decodificação huffman.

- Seu sistema deve receber como entrada um arquivo contendo uma página html (exemplo, um artigo da Wikipedia) e você precisa formar uma árvore binária (huffman) para o mesmo;
- Durante a construção da huffman tree, use a fila de prioridade para selecionar nós com menores frequências;
- Depois de construir a árvore, percorra a árvore e crie um dicionário de palavras de código (letra para código);
- Dadas quaisquer novas frases, seu sistema deve mostrar como a sentença é convertida em código de huffman e depois decodificada de volta para a sentença original;
- Observe que você deve implementar o BST e o Heap por conta própria e não deve depender de nenhuma biblioteca da liguagem de programação;
- Você pode usar bibliotecas externas como GraphViz para exibir sua árvore huffman.

[DESCRIÇÃO - 2] Network Flow: Task allocation using Bipartite Graph

- Neste projeto, você recebe dois conjuntos: conjunto de funcionários e tarefas. Para cada tarefa, você também recebe a lista de funcionários que podem concluir a tarefa;
- Modele esse cenário como um gráfico bipartido e aloque o trabalho de forma que o trabalho seja concluído o mais rápido possível;
- Resolva esse problema usando o fluxo de rede. Implemente os algoritmos Ford-Fulkerson e Edmond-Karp;
- Sua interface do usuário deve permitir a visualização o caminho aumentado (atualização do fluxo) durante cada iteração; e
- Apresente um avaliação experimental com diversos testes de casos.

[DESCRIÇÃO - 3] Approximate Graph Coloring

- Faça um análise e descrição do seguinte artigo - Karger, Motwani, Sudan, 1998, Approximate Graph Coloring by Semidefinite Pro-gramming;
- Pesquise e implemente duas possíveis soluções aproximadas para resolver a coloração de grafos com resultado aproximados;
- Sua interface do usuário deve permitir a visualização do resultado final do grafo colorido;
- Pesquise e adote um benchmark de grafos para uma avaliação experimental usando coloração de grafos; e
- Apresente uma análise sobre os resultados encontrados com os algoritmos propostos.

[DESCRIÇÃO - 4] Genetic Algorithms

- Faça um análise e descrição do seguinte artigo - Simulating nature's methods of evolving the best design solution. Cezary Janikow e Daniel Clair. IEEE. 1995;
- Faça um descrição sobre o projeto Dinossauro da Google apresentado em https://www.youtube.com/watch?v=P7XHzqZjXQs
- Apresente os conceitos sobre algoritmos genéticos;
- Aqueles que gastam parte de seu tempo jogando jogos de computador como o Sims (criando suas próprias civilizações e evoluindo-os) freqüentemente se encontram jogando contra sofisticados GAs (Genetic Algorithms) de inteligência artificial em vez de contra outros jogadores humanos online. Neste sentido, implemente um algoritmo genético para um jogo de sua escolha; e
- Apresente um avaliação experimental para o algoritmo genético implementado.

[DESCRIÇÃO - 5] Problema do Clique

- Faça um análise e descrição do seguinte artigo - Finding All Maximal Connected s-Cliques in Social Networks: https://openproceedings.org/2018/conf/edbt/paper-28.pdf
- Implemente um algoritmo guloso para o problema do Clique;
- Usando o dataset disponível em https://www.kaggle.com/datasets/thoughtvector/customer-support-on-twitter . Em seguida, defina conjuntos que deverão ser análisado como cliques, para identificar a relação/conexão entre os tweets;
- Pesquise e adote um benchmark de grafos para uma avaliação experimental; e
- Apresente uma análise sobre os resultados encontrados com os algoritmos.

[DESCRIÇÃO - 6] Satisfabilidade

- Faça um análise e descrição do seguinte artigo - Z3: An Efficient SMT Solver: **https://dl.acm.org/citation.cfm?id=1792766**
- O projeto consistirá em uma série de programas simples que aceitam amostras de wffs (*well formed formulas*) representadas como arquivos no formato DIMACS CNF (apresentado abaixo), e determinar se a formula é satisfatória ou não. Além dessa determinação, você também medirá seu tempo de execução para cada solução e representará esse tempo como uma função do número de variáveis no wff. Em particular, o projeto inclui o desenvolvimento dos seguintes programas:
  - Um solucionador que determina a satisfatibilidade simplesmente gerando todas as atribuições possíveis (2^V deles em que V é o número de variáveis), testando cada atribuição em relação ao wff e parando quando uma solução é encontrada ou quando todas as atribuições possíveis foram tentadas;
  - Um solucionador que usa backtracking simples e que tem desempenho médio muito melhor que o anterior;
  - Um solucionador para apenas 2SAT que faz melhor que $O(2^V)$.
- **Testes com WFFs**.
  - Em https://www3.nd.edu/~kogge/courses/cse30151-fa17/Public/Projects/Project1/TestFiles/ existem vários arquivos de teste, cada um contendo vários wffs;
  - O site http://www.satcompetition.org tem uma rica descrição de uma série de problemas e desafios no formato de entrada para o projeto;
  - Um exemplo do formato CNF para o problema mencionado é:

```
c 17 3 S
p cnf 3 4
1 -2 0
2 3 0
-1 -3 0
-1 -2 3 0
```

[DESCRIÇÃO - 7] Caixeiro Viajante

- Faça um análise e descrição do seguinte artigo - Practical Approach for Solving School Bus Problems:
  http://onlinepubs.trb.org/Onlinepubs/trr/1988/1202/1202-007.pdf
- Implemente um algoritmo aproximado para o caixeiro viajante multi-objetivo;
- Modele e apresente uma routa para os ônibus de Boa Vista-RR, considerando dois objetivos: maximização do número de passageiros e minimização do tempo da rota.
- Pesquise e adote um benchmark de grafos para uma avaliação experimental; e
- Apresente uma análise sobre os resultados encontrados com os algoritmos.

[DESCRIÇÃO - 8] Problema da mochila

O Problema da Mochila 0/1, que consiste em escolher n ítens, tais que o somatório das utilidades é maximizado sem que o somatório dos pesos extrapolem a capacidade da Mochila.

- Faça um análise e descrição do seguinte artigo - Ricardo Fukasawa, Joe Naoum-Sawaya e Daniel Oliveira. The price-elastic knapsack problem Author links open overlay panel. Omega, Volume 124, April 2024. https://doi.org/10.1016/j.omega.2023.103003
- Implemente uma solução usando Backtracking
- Implemente uma solução usando Programação Dinâmica
- Apresente um solução para o contexto de um Planejamento de Missão Espacial, conforme abaixo:

Situação: Imagine que uma equipe de cientistas está planejando uma missão espacial para explorar um novo planeta. A missão envolve enviar um rover para o planeta para coletar amostras e realizar experimentos. O rover tem uma capacidade limitada de carga (peso), e há uma lista de equipamentos e suprimentos disponíveis para serem levados. Cada equipamento ou suprimento tem um peso específico e contribui de maneira diferente para o sucesso da missão (valor).

Problema: A equipe precisa decidir quais itens devem ser carregados no rover para maximizar o valor científico da missão, sem exceder a capacidade de carga máxima.

Dados:

- Capacidade máxima de carga do rover: 100 kg.
- Itens disponíveis:
  - **Câmera de alta resolução**: Peso: 20 kg, Valor: 40 pontos.
  - **Braço robótico**: Peso: 50 kg, Valor: 100 pontos.
  - **Analisador de solo**: Peso: 30 kg, Valor: 60 pontos.
  - **Detector de radiação**: Peso: 10 kg, Valor: 30 pontos.
  - **Fonte de energia extra**: Peso: 40 kg, Valor: 70 pontos.

Objetivo: Escolher os itens que devem ser levados para maximizar o valor total (o sucesso da missão) sem exceder a capacidade máxima de 100 kg do rover.

Solução: A solução ótima será a combinação de itens que maximiza o valor total da missão sem exceder o limite de peso de 100 kg.

[DESCRIÇÃO - 9] DFS paralelo

- Faça um análise e descrição do seguinte artigo: A. Aggarwal, R. J. Anderson, and M.-Y. Kao. 1989. Parallel depth-first search in general directed graphs. In Proceedings of the twenty-first annual ACM symposium on Theory of computing (STOC '89). Association for Computing Machinery, New York, NY, USA, 297–308. https://doi.org/10.1145/73007.73035
- Material de consulta https://www.lrde.epita.fr/~bleton/doc/parallel-depth-first-search.pdf
- Implemente uma solução para Navegação em Rede de Transportes Urbanos, conforme descrição abaixo:

Contexto: Imagine que você está desenvolvendo um sistema de navegação para uma grande rede de transportes urbanos, que inclui ônibus, trens e metrôs. A cidade é dividida em várias regiões, cada uma conectada por várias rotas de transporte. O objetivo do sistema é encontrar rotas alternativas entre diferentes pontos de interesse na cidade, considerando possíveis congestionamentos ou interrupções no serviço.

Problema: A rede de transportes urbanos pode ser modelada como um grafo onde os vértices representam estações ou pontos de interesse e as arestas representam as rotas de transporte. Cada aresta tem um custo associado (por exemplo, tempo de viagem, distância ou custo monetário). Como a rede é grande e pode haver várias rotas alternativas, um algoritmo eficiente é necessário para explorar múltiplos caminhos possíveis de forma paralela.

Atividade: A tarefa é desenvolver um algoritmo que use a busca em profundidade paralela (Parallel Depth-First Search - DFS) para percorrer o grafo de fluxo e encontrar todas as rotas alternativas entre dois pontos de interesse específicos na cidade.

Descrição da Atividade:

1. **Modelagem do Grafo:**
   - Cada ponto de interesse na cidade é representado por um vértice no grafo.
   - Cada rota de transporte entre os pontos de interesse é representada por uma aresta no grafo.
   - As arestas podem ter pesos que representam o custo de percorrer essa rota (tempo, distância, etc.).
2. **Paralelização da Busca:**
   - O algoritmo de busca em profundidade será modificado para ser executado em paralelo.
   - Cada thread ou processo do sistema computacional iniciará a busca a partir de um vértice de partida, explorando diferentes caminhos simultaneamente.
   - Quando uma thread encontra um caminho que leva ao destino desejado, ela registra o caminho encontrado.
3. **Coordenação e Sincronização:**
   - Como várias threads estarão percorrendo o grafo simultaneamente, será necessário coordenar o acesso às estruturas de dados compartilhadas (como a lista de caminhos encontrados) para evitar condições de corrida.
   - A sincronização pode ser realizada usando locks, semáforos ou outras técnicas de

controle de concorrência.

4. **Agregação dos Resultados:**
    - Após todas as threads concluírem a busca, os caminhos encontrados são agregados.
    - O sistema pode então selecionar a melhor rota ou apresentar várias alternativas ao usuário.

Resultado Esperado:

- O algoritmo deve retornar todas as rotas alternativas entre a Estação Central e o Aeroporto.
- Com base nas preferências do usuário (tempo de viagem mais curto, menor número de transferências, etc.), a melhor rota pode ser escolhida.

[DESCRIÇÃO - 10] Executor simbolico

- Faça um análise e descrição do seguinte artigo: James C. King. 1976. Symbolic execution and program testing. Commun. ACM 19, 7 (July 1976), 385–394. https://doi.org/10.1145/360248.360252
- Apresente e descreve a complexidade dos principais algoritmos utilizados pela ferramenta Z3 - https://github.com/Z3Prover/z3
- Implemente uma solução para o desenvolvimento de um Executor Simbólico para o Problema do Caixeiro Viajante com Python e Z3

Objetivo:

Criar um executor simbólico que usa o solver Z3 para encontrar uma solução para o problema do Caixeiro Viajante, minimizando a distância total percorrida.

Ferramentas Necessárias:

- Python 3
- Biblioteca Z3 (solver SMT)

Passos para Implementação:

- **Instalação do Z3:** Instale a biblioteca Z3 para Python se ainda não a tiver instalada.
- **Definição do Problema:** O problema do Caixeiro Viajante (TSP) pode ser formulado como um problema de otimização onde você precisa encontrar o caminho mais curto que visita um conjunto de cidades exatamente uma vez e retorna à cidade inicial.
- **Criação do Modelo no Z3:** Defina variáveis que representam as distâncias entre as cidades e variáveis de decisão para o caminho do caixeiro viajante.
- **Adição das Restrições do Problema:** Adicione restrições que garantem que o caminho seja válido e que o circuito seja fechado (retorna à cidade inicial).
- **Definição da Função Objetivo:** Defina a função objetivo para minimizar a distância total percorrida.
- **Resolução do Problema:** Resolva o problema usando o solver Z3 e extraia a solução.
- **Teste e Validação:** Teste o seu executor simbólico com diferentes matrizes de distância e valide a solução encontrada.
- **Documentação e Resultados:** Documente o código e explique as decisões tomadas na modelagem e resolução do problema. Inclua exemplos de entradas e saídas para ilustrar o

funcionamento do seu executor simbólico.

- **Exemplo de funcionamento:**

```
distance_matrix = [
    [0, 10, 15, 20],
    [10, 0, 35, 25],
    [15, 35, 0, 30],
    [20, 25, 30, 0]
]

tsp_solver(distance_matrix)
```

[DESCRIÇÃO - 11] Algoritmo de Parentização

- Faça um análise e descrição do seguinte artigo: Haruno, M., Shirai, S. & Ooyama, Y. Using Decision Trees to Construct a Practical Parser. *Machine Learning* **34**, 131–149 (1999). https://doi.org/10.1023/A:1007597902467
- Implemente uma solução para Decisão de Investimento em Projetos

Objetivo:

A empresa precisa tomar uma decisão sobre qual projeto investir utilizando uma árvore de decisão. O algoritmo de parentização será utilizado para estruturar a árvore de decisão com base em critérios de priorização e lógica de negócios.

Passos para a Atividade:

1. **Identificação dos Critérios:**
    - Os participantes devem listar todos os critérios relevantes para a decisão, como:
        - Orçamento necessário
        - Potencial de retorno sobre o investimento
        - Riscos associados
        - Alinhamento com a estratégia de longo prazo
2. **Estruturação da Árvore de Decisão:**
    - Utilizando papel ou um software de diagramas, os participantes devem desenhar a árvore de decisão.
    - No nó raiz, insira a decisão principal: “Qual projeto devemos escolher?”
    - A partir do nó raiz, crie nós filhos baseados nos critérios identificados. Por exemplo, um nó pode perguntar: “O orçamento está dentro dos limites estabelecidos?” com ramos que levam a diferentes nós de decisão baseados na resposta (sim ou não).
3. **Aplicação do Algoritmo de Parentização:**
    - Os participantes devem aplicar o algoritmo de parentização para garantir que a árvore esteja organizada de maneira lógica, sem nós ambíguos ou redundantes.
    - Isso envolve revisar as ramificações da árvore e reorganizar os nós, se necessário, para assegurar que cada decisão leve a uma conclusão clara e bem estruturada.
4. **Tomada de Decisão:**
    - Com a árvore de decisão completamente parentizada, os participantes devem utilizar

a árvore para simular a tomada de decisão.
- Cada critério será avaliado passo a passo, e a árvore guiará os participantes até a decisão final sobre qual projeto escolher.

5. **Discussão e Análise:**
   - Após completar a árvore de decisão e tomar a decisão final, os participantes devem discutir o processo e analisar se a estrutura da árvore facilitou uma tomada de decisão clara e eficiente.
   - Eles devem identificar possíveis melhorias na árvore, como critérios adicionais ou diferentes ordens de priorização.

[DESCRIÇÃO - 12] Clusterização

- Faça um análise e descrição do seguinte artigo: T. N. Pappas and N. S. Jayant, "An adaptive clustering algorithm for image segmentation," *International Conference on Acoustics, Speech, and Signal Processing,*, Glasgow, UK, 1989, pp. 1667-1670 vol.3, doi: 10.1109/ICASSP.1989.266767.
- Implemente uma solução para Organização de Fotografias de Eventos

Contexto: Você possui uma grande coleção de fotos tiradas em diferentes eventos ao longo de um ano. Quer organizar essas fotos em categorias, como "Aniversário", "Férias", "Casamento", etc., usando clustering para agrupar imagens similares sem a necessidade de treinar um modelo de aprendizado de máquina.

Objetivo: Agrupar fotos em diferentes categorias com base em características visuais semelhantes, usando um algoritmo de clustering, como K-Means ou DBSCAN, para facilitar a organização e a busca das imagens.

Passos para Implementação:

Preparação dos Dados:

- **Coleta de Imagens:** Reúna todas as fotos em uma pasta específica.
- **Pré-processamento:** Reduza o tamanho das imagens, se necessário, para otimizar o processamento.

Extração de Recursos:

- Utilize bibliotecas como `OpenCV` ou `PIL` para carregar as imagens e converter para um formato que pode ser utilizado pelo algoritmo de clustering.
- Extraia características relevantes das imagens, como histogramas de cores, descritores de bordas, ou outros vetores de características. Pode-se usar métodos como o Histograma de Cores ou o Histogram of Oriented Gradients (HOG) para isso.

Aplicação do Algoritmo de Clustering:

- **Escolha do Algoritmo:** Utilize o K-Means ou DBSCAN do `scikit-learn`. O K-Means pode ser uma escolha simples para começar, onde você define o número de clusters com base em uma estimativa inicial.

Análise e Organização:

- **Classificação:** Use os rótulos gerados pelo clustering para organizar as imagens em pastas ou categorias. As imagens que compartilham o mesmo rótulo serão agrupadas na mesma categoria.

Verificação e Ajustes:

- Revise os agrupamentos e ajuste o número de clusters ou parâmetros do algoritmo, se necessário, para melhorar a precisão dos agrupamentos.