Universidade Federal de Roraima
Departamento de Ciência da Computação
Análise de Algoritmos – DCC606

PROJETO FINAL - Prazo de Entrega: 14/06/2023

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_2023.`

- ✔ 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 (26/06/2023 - 26/06/2023)**.
  - 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;
- Colete tweets do Twitter utilizando como chave de coleta os trending topics. 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.