Projeto prático 1

SCC 0606 – Estrutura de Dados II
Prof. Maria Cristina Ferreira de Oliveira
13 de março de 2024

1. Descrição

O objetivo deste projeto, que pode ser feito em grupos de 2 ou 3 alunos, é contribuir para a melhor compreensão do conteúdo da disciplina. Sua entrega é fundamental para um bom desempenho nas avaliações e na disciplina como um todo. Neste projeto, pede-se:

(1) Implementar um algoritmo de ordenação, à escolha. Pode ser um algoritmo específico encontrado na literatura, ou um algoritmo pensado pelo grupo.
(2) Analisar as complexidades de tempo do algoritmo implementado, bem como dos algoritmos descritos na próxima seção. As complexidades devem ser analisadas em termos teóricos, usando análise assintótica, considerando os diversos casos apresentados (melhor caso, o caso médio e o pior caso), e também em termos empíricos, usando medições de tempo de execução.
    a. Espera-se ouvir as conclusões tiradas pelos alunos. Alguns exemplos de perguntas que poderão ser respondidas são: “Alguns dos algoritmos são assintoticamente mais eficientes, mas para pequenas entradas é mais demorado?”; “Os algoritmos são igualmente eficientes para todos os tipos de entrada de dados? (Ordenados, ordenados decrescentemente, aleatorizados, aleatorizados com repetição)”; “Existe algum caso de entrada no qual o tempo de execução dos algoritmos varia muito cada vez que este é executado? Porque?”; “Existe algum algoritmo que para algum caso de entrada destoa muito da curva assintótica esperada? Porque?”
(3) Entregar um relatório com os resultados e discussões.

Para a análise empírica serão disponibilizadas sequências de entrada de diferentes tamanhos (variando desde 100 a 10.000) no sistema RunCodes, Para cada sequência, serão fornecidas 3 variações: algumas entradas em que a sequência tem valores aleatórios; uma em que a sequência já está ordenada de maneira crescente e uma em que a sequência está ordenada de maneira decrescente. O tempo de execução de cada configuração (algoritmo vs sequência vs variação) deve ser medido pelo menos 10 vezes, para evitar variações arbitrárias nos tempos de execução, e portanto os tempos reportados para cada configuração devem ser a média. É recomendado seguir o padrão de desenvolvimento e análise de complexidade adotado nas aulas da disciplina.

2. Baseline de Algoritmos

Seguem abaixo os resultados de tempo dos algoritmos já desenvolvidos que serão utilizados como base de comparação com o algoritmo proposto pelo grupo, incluindo análise assintótica:

Dados de entrada	Ordenados					Ordenados decrescentemente				
Tamanho	100	500	1000	5000	10000	100	500	1000	5000	10000
Alg A	0.199	10.47	37.50	1177.1	4997.5	0.497	10.47	60.94	1379.2	5752.4
Alg B	0.199	2.294	3.787	15.36	36.48	0.194	1.599	3.586	22.54	32.71
Alg C	45.61	187.79	392.4	1999.0	3886.4	39.89	218.8	397.8	2032.3	4234.9
Alg D	0.0	1.563	3.124	17.277	40.779	0.0	1.577	3.13	14.105	41.129

Dados de entrada	Aleatorizados					Aleatorizados com repetição				
Tamanho	100	500	1000	5000	10000	100	500	1000	5000	10000
Alg A	0.0	9.375	57.25	1440.5	5479.5	0.595	13.76	47.07	1242.4	5286.8
Alg B	0.199	1.395	2.991	16.45	34.61	0.299	1.594	3.972	16.06	32.99
Alg C	40.17	194.2	404.0	2071.7	4153.1	43.18	200.5	375.5	2080.7	4132.3
Alg D	0.0	1.563	3.16	15.738	37.591	0.0	1.560	3.126	15.698	31.393

*Tempos das tabelas estão em milisegundos

3. Relatório de projeto

Deve ser redigido um relatório descrito do que foi desenvolvido, com no máximo 5 páginas, incluindo possíveis tabelas, gráficos e lista de referências. O relatório deve conter as seguintes informações:

- Uma breve introdução apresentando a organização do relatório;
- Integrantes do grupo, com nome e número USP;

- Possíveis descrições pertinentes à sua implementação do trabalho. Por exemplo, escolha do algoritmo utilizado, como foi feita a comparação das curvas assintóticas, etc;
- Tabelas apresentando as medições de tempo realizadas para cada algoritmo;
- Discussão dos resultados empíricos;
- Análise teórica (assintótica) de complexidade de tempo para cada algoritmo. A análise deve ser completa, portanto, é recomendado que analisem os *baselines* e descrevam as vantagens e desvantagens de cada algoritmo, bem como o comportamento assintótico em cada caso.
- Menção a eventuais dificuldades enfrentadas durante o trabalho;
- Uma seção de conclusão em que o grupo apresenta sua opinião sobre os algoritmos de ordenação, embasada nos resultados apresentados.

4. Conteúdo e data de entrega

O trabalho pode ser realizado em duplas ou trios, e a data máxima de entrega é 31/03/2023. Trabalhos com maior atraso não serão aceitos, recebendo nota zero. Quaisquer projetos similares terão nota zero independente de qual for o original e qual for a cópia. Será utilizada ferramenta automatizada para a detecção de plágio, com conferência manual de casos suspeitos. Os projetos devem ser entregues via RunCodes (https://runcodes.icmc.usp.br/). O formato da entrega deve ser um arquivo com o algoritmo de ordenação para os casos de entrada disponíveis e um *.ZIP contendo:

- O relatório descrevendo o que foi desenvolvido e as conclusões obtidas;
- Todos os códigos utilizados no desenvolvimento do trabalho;
- A pasta a ser zipada deve ter por nome só os números USP dos alunos, separados por traços. Trabalhos sem esse padrão de nome de arquivo não serão corrigidos e valerão zero;
- É recomendado incluir os nomes e números USP dos integrantes do grupo em todos os arquivos da entrega;
- O trabalho deve ser desenvolvido utilizando a linguagem C ou Python, em qualquer versão aceita pelo RunCodes. Não é permitido utilizar bibliotecas que já disponibilizam os algoritmos de forma parcial ou total;
- Todo o código do projeto deve ser desenvolvido pelos integrantes do grupo. A utilização de qualquer código criado por terceiros será tratada como plágio.

5. Critérios de avaliação

- 3,0 pontos pela implementação do algoritmo de forma correta (avaliado segundo resultado do RunCodes)
- 1,0 ponto por código legível e bem comentado. Incluindo tanto o código-fonte referente ao algoritmo, quanto o código referente às medições de tempo;
- 6,0 pontos pelo relatório, sendo:
    - 5,0 pontos pela análise assintótica de complexidade de tempo dos algoritmos;
    - 1,0 ponto pelo relatório, considerando estruturação, boa escrita e qualidade de apresentação de resultados.

6. Dicas RunCodes

- Os dados de entrada do trabalho no RunCodes são os mesmos utilizados para gerar os tempos da baseline de algoritmos descritos anteriormente.
- Os arquivos de entrada e saída são de tipos .in e .out, numerados de 1 a 20, da seguinte forma:
    - A cada 4 números, altera-se o tamanho da entrada:
        - 1.in a 4.in: 100 números para serem ordenados
        - 5.in a 8.in: 500 números para serem ordenados
        - 9.in a 12.in: 1000 números para serem ordenados

        - …
    - A diferença entre cada um dos 4 arquivos de cada grupo acima se define por como os números aparecem nos arquivos de entrada. O primeiro arquivo do grupo é completamente ordenado, o segundo está ordenado inversamente, o terceiro possui entrada aleatorizada, e o último possui entrada aleatória, mas com repetições de números.
- A entrada e saída dos dados pode ser tratada da maneira tradicional existente em cada linguagem. Por exemplo, usando a função input() e print() para Python e scanf() e printf() para C.
- O RunCodes não aceita todas as funções existentes para tratamento de dados e demais operações. Na dúvida, opte por funções e bibliotecas mais tradicionalmente utilizadas.