UNIVERSIDADE DE COIMBRA
FACULDADE DE CIÊNCIAS E TECNOLOGIA

Departamento de Engenharia Informática

Projeto #4
Algoritmos e Estruturas de Dados

2020-2021 – 2º Semestre

Submissão relatório (InforEstudante) e Mooshak:
4.1 23 de abril 23:59 4.2 30 de abril 23:59
4.3 7 de maio 23:59 4.4 14 de maio 23:59

Anotações: *Em anexo ao enunciado do projeto são disponibilizados quatro templates do relatório para cada uma das semanas de duração do projeto.*

É incentivado que os alunos discutam ideias e questões relativas ao trabalho proposto, mas é entendido que quer a reflexão final sobre os resultados obtidos, quer o código desenvolvido, são da autoria de cada estudante. Procedimentos contrários ao que é dito acima, nomeadamente cópia de código desenvolvido por colegas ou obtido da net é entendido como fraude. Para além de a fraude denotar uma grave falta de ética e constituir um comportamento não admissível num estudante do ensino superior e futuro profissional licenciado, esta prejudica definitivamente o processo de aprendizagem do infrator.

Objetivos:

Com o desenvolvimento deste projeto pretende-se que o aluno consolide os conhecimentos sobre os algoritmos de ordenamento estudados em Algoritmos e Estruturas de Dados com foco na: (1) programação dos algoritmos; (2) vantagens e desvantagens de cada uma, nomeadamente no que diz respeito a complexidade temporal e espacial; (3) análise teórica e empírica da complexidade temporal.

Nota: os subprojetos serão desenvolvidos e entregues pela ordem da sua descrição: 4.1, 4.2, 4.3, e 4.4.

Base comum a todos os subprojetos

O ordenamento será feito sobre sequências de inteiros com valores entre 0 e 10,000,000. O resultado do ordenamento será a sequência de entrada ordenada de forma crescente.

Entrada:

A primeira linha indica o número de inteiros a ordenar (N). De seguida existem N linhas, com um inteiro por linha.

Saída:

O output terá N linhas, com um inteiro por linha.

Exemplo:

Entrada:

4
17
21
5
51

Saída:
5
17
21
51

Os diferentes tipos de sequências a testar serão os seguintes:

1) SEQ_ALEATORIA: sequência aleatória.

2) SEQ_ORDENADA_DECRESCENTE: sequência ordenada de forma decrescente.

3) SEQ_QUASE_ORDENADA_1% e SEQ_QUASE_ORDENADA_5%: sequência quase ordenada com x% de trocas a criarem desordenamento. Para criarem este tipo de sequência podem começar com uma sequência ordenada e depois sorteiam x% de trocas aleatórias.

Estas quatro sequências serão usadas em todos os subprojetos com os seus nomes respetivos: SEQ_ALEATORIA, SEQ_ORDENADA_DECRESCENTE, SEQ_QUASE_ORDENADA_1% e SEQ_QUASE_ORDENADA_5%.

Subprojeto 4.1: Shell Sort com exploração de diferentes sequências de incremento

Implementação e exploração de diferentes variantes do Shell Sort no que respeita às sequências de incremento que o algoritmo utiliza. A versão base do Shell Sort (referida como B-SS de Base Shell Sort) será definida como usando potências de 2 na sua sequência de incremento. Ou seja, o B-SS usará a seguinte sequência de incremento: 1, 2, 4, 8, 16, 32, etc. Têm liberdade de definir como é que esta sequência termina, se de forma fixa (por exemplo, até 128) ou se em função do tamanho da sequência (ou seja, definirem uma regra para a escolha do incremento máximo em função do valor de N).

Além do B-SS devem implementar e explorar diferentes sequências de incremento e analisar como é que estas diferentes sequências influenciam o desempenho do algoritmo, fazendo esta análise de

forma detalhada para todos os tipos de sequência descritos na secção inicial. Será valorizada a criatividade e raciocínio lógico na exploração das diferentes sequências de incremento.

No relatório vão apresentar as 4 melhores alternativas encontradas ao B-SS. Devem catalogar estas alternativas como I-SS-1, I-SS-2, I-SS-3 e I-SS-4 (I-SS de Improved Shell Sort), e indicar que sequência de incremento ou regra de incremento é que cada uma segue.

No Mooshak devem submeter o B-SS no problema A e o I-SS no problema B. O problema B será usado para verificarem as vossas implementações alternativas. Basta submeterem uma das variantes alternativas, mas podem submeter todas se quiserem validar cada uma.

Subprojeto 4.2: Quicksort vs. Quicksort + Insertion Sort

Implementação do Quicksort na sua versão completamente recursiva (referida como QS) e implementação do Quicksort com o uso do Insertion Sort para ordenar subsequências mais pequenas evitando assim chamadas recursivas adicionais (referida como QS+IS). Dentro da implementação do QS+IS devem explorar diferentes pontos de transição para o Insertion Sort e verificar como é que o desempenho do algoritmo é influenciado. Ou seja, devem verificar empiricamente a partir de que tamanhos de subsequência é que é útil passar a usar o Insert Sort em vez do Quicksort completamente recursivo. Devem fazer esta análise de forma detalhada para todos os tipos de sequência descritos na secção inicial.

No relatório, além do QS, vão apresentar os 4 melhores pontos de transição encontrados no QS+IS. Devem catalogar estas variantes como QS+IS <k1>, QS+IS <k2>, QS+IS <k3> e QS+IS <k4>, indicando o valor usado no ponto de transição (k1, k2, k3 e k4). Por exemplo, se um dos pontos de transição selecionados foi 10, o nome da variante será QS+IS 10.

No Mooshak devem submeter o QS no problema C e o QS+IS no problema D. No problema D basta submeterem o QS+IS com um qualquer ponto de transição, mas podem submeter com todos os valores testados se quiserem validar a implementação com diferentes valores de transição.

Subprojeto 4.3: Radix Sort

Implementação do Radix Sort (RS). Podem decidir entre uma implementação Most Significant Digit (MSD) ou Least Significant Digit (LSD). Como referido na secção inicial, os inteiros terão sempre valores entre 0 e 10,000,000. Podem ter em conta este conhecimento na abordagem a seguir (por exemplo, na escolha das estruturas de dados a usar). Devem analisar empiricamente o RS para todos os tipos de sequência descritos na secção inicial.

No relatório vão ter de discutir as diferenças de abordagem entre a implementação escolhida e a implementação alternativa. Isto é, se escolheram a abordagem MSD devem discutir a abordagem LSD, enquanto que se escolheram a abordagem LSD devem discutir a abordagem MSD.

No Mooshak devem submeter o RS no problema E.

Subprojeto 4.4: Análise comparativa dos algoritmos de ordenamento dos subprojetos anteriores

Análise comparativa das melhores variantes. Neste subprojeto devem ser comparadas as seguintes variantes:

- B-SS
- A melhor variante do I-SS
- QS
- A melhor variante do QS+IS
- RS

Devem analisar empiricamente estas variantes para os quatro tipos de sequência descritos na secção inicial. As melhores variantes do I-SS e do QS+IS devem ser escolhidas para cada tipo de sequência, em vez de forma global. Por exemplo, na sequência SEQ_ALEATORIA a melhor variante do QS+IS pode ser a QS+IS 10, no entanto na sequência SEQ_QUASE_ORDENADA_1% a melhor variante pode ser a QS+IS 100. Neste caso, para a comparação neste subprojeto deve ser usada a variante QS+IS 10 na sequência SEQ_ALEATORIA, e a variante QS+IS 100 na sequência SEQ_QUASE_ORDENADA_1%. O mesmo raciocínio aplica-se ao I-SS.

Relatório Projeto 4.1 AED 2020/2021

Nome: Nº Estudante:

TP (inscrição): *Login* no *Mooshak:*

Nº de horas de trabalho: *H* *Aulas Práticas de Laboratório:* *H* *Fora de Sala de Aula:* *H*

(A Preencher pelo Docente) CLASSIFICAÇÃO:

Comentários:

Registar os tempos computacionais do B-SS e das 4 variantes selecionadas do I-SS para os diferentes tipos de sequências. O tamanho das sequências (N) deve ser crescente e terminar em 10,000,000. Só deve ser contabilizado o tempo de ordenamento. Exclui-se o tempo de leitura do input e de impressão dos resultados. Devem apresentar e discutir as regressões para a melhor variante em cada tipo de sequência.

Gráfico para SEQ_ALEATORIA

Gráfico para SEQ_ORDENADA_DECRESCENTE

Gráfico para SEQ_QUASE_ORDENADA_1%

Gráfico para SEQ_QUASE_ORDENADA_5%

Sequência de incremento ou regra de incremento de cada variante (B-SS, I-SS-1, I-SS-2, I-SS-3, I-SS-4):

_____
_____
_____

Análise dos resultados:

_____
_____
_____
_____
_____
_____
_____
_____

Relatório Projeto 4.2 AED 2020/2021

Nome: Nº Estudante:

TP (inscrição): *Login* no *Mooshak:*

Nº de horas de trabalho: *H* *Aulas Práticas de Laboratório:* *H* *Fora de Sala de Aula:* *H*

(A Preencher pelo Docente) CLASSIFICAÇÃO:

Comentários:

Registar os tempos computacionais do QS e das 4 variantes selecionadas do QS+IS para os diferentes tipos de sequências. O tamanho das sequências (N) deve ser crescente e terminar em 10,000,000. Só deve ser contabilizado o tempo de ordenamento. Exclui-se o tempo de leitura do input e de impressão dos resultados. Devem apresentar e discutir as regressões para a melhor variante em cada tipo de sequência.

Gráfico para SEQ_ALEATORIA **Gráfico para SEQ_ORDENADA_DECRESCENTE**

Gráfico para SEQ_QUASE_ORDENADA_1% **Gráfico para SEQ_QUASE_ORDENADA_5%**

Análise dos resultados:

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

Relatório Projeto 4.3 AED 2020/2021

Nome: Nº Estudante:

TP (inscrição): *Login* no *Mooshak:*

Nº de horas de trabalho: *H* *Aulas Práticas de Laboratório:* *H* *Fora de Sala de Aula:* *H*

(A Preencher pelo Docente) CLASSIFICAÇÃO:

Comentários:

Registar os tempos computacionais do RS para os diferentes tipos de sequências. O tamanho das sequências (N) deve ser crescente e terminar em 10,000,000. Só deve ser contabilizado o tempo de ordenamento. Exclui-se o tempo de leitura do input e de impressão dos resultados. Devem apresentar e discutir as regressões para cada tipo de sequência.

Gráfico para SEQ_ALEATORIA **Gráfico para SEQ_ORDENADA_DECRESCENTE**

Gráfico para SEQ_QUASE_ORDENADA_1% **Gráfico para SEQ_QUASE_ORDENADA_5%**

Análise dos resultados, discutindo a implementação alternativa do RS (MSD ou LSD) e considerando também a complexidade espacial do algoritmo:

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

Relatório Projeto 4.4 AED 2020/2021

Nome: Nº Estudante:

TP (inscrição): *Login* no *Mooshak:*

Nº de horas de trabalho: *H Aulas Práticas de Laboratório:* *H Fora de Sala de Aula:* *H*

(A Preencher pelo Docente) CLASSIFICAÇÃO:

Comentários:

Registar os tempos computacionais das variantes em consideração para os diferentes tipos de sequências. O tamanho das sequências (N) deve ser crescente e terminar em 10,000,000. Só deve ser contabilizado o tempo de ordenamento. Exclui-se o tempo de leitura do input e de impressão dos resultados.

Gráfico para SEQ_ALEATORIA **Gráfico para SEQ_ORDENADA_DECRESCENTE**

Gráfico para SEQ_QUASE_ORDENADA_1% **Gráfico para SEQ_QUASE_ORDENADA_5%**

Sequência de incremento ou regra de incremento do I-SS para cada tipo de sequência:

_____
_____
_____

Análise dos resultados considerando também a complexidade espacial dos algoritmos:

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____