UNIVERSIDADE FEDERAL DO CEARÁ – UFC SOBRAL
TÉCNICAS DE PROGRAMAÇÃO – PROF. FISCHER

Trabalho Prático 1

Grupos
Time: Um coordenador (aluno da pós) e 5 alunos da graduação.
Data: 08/10/23
Entrega: apresentação e entrega de relatório.

Requisitos para apresentação do trabalho:

1) Apresentar as tabelas dos tempos de execução obtidos pelos algoritmos sobre as instâncias testadas, comparando sua evolução com a evolução dos tempos seguindo a complexidade teórica correspondente.

2) Quando o tempo de CPU for inferior a 5 segundos, faça uma repetição da execução tantas vezes quantas forem necessárias para que o tempo ultrapasse 5s (faça um while), conte quantas foram as execuções e reporte a média.

3) Computar a memória gasta para cada instâncias.

4) Fazer a documentação do código, com comentários, diagramas e artefatos que demonstrem a implementação realizada

5) Apresentar tabelas contendo seis colunas para cada algoritmo aplicado às instâncias. A primeira com o nome das instâncias. Uma coluna com o valor da complexidade teórica, uma com o tempo de CPU utilizado e uma com a razão destes dois valores. Quinta coluna com os valores de tempo mais altos encontrados e a sexta coluna com os valores mais baixos encontrados, considerando cada instância. Cada linha da tabela é associada a uma instância e contém a identificação da mesma. Nesta tabela coloque as instâncias em ordem crescente de tamanho.

1. **Problema de Programação Hiperbólica (PPH)**

PPH: Dado um conjunto de pares ordenados $\{(a_1, b_1), \ldots, (a_n, b_n)\}$ e um par obrigatório (a_0, b_0), onde $a_i, b_i \in Z^+, \quad \forall\, i = 0, 1, \ldots, n$, determinar $S \subseteq N$ onde $N = \{1, \ldots, n\}$ que maximiza:

$$R(S) = \frac{a_0 + \sum_{t \in S} a_t}{b_0 + \sum_{t \in S} b_t}$$

Lemma 1 *Seja R^* o valor da razão máxima obtida para o (PPH) e $S*$ um subconjunto de N tal que $R(S^*) = R^*$. Então, um par t pertence a qualquer S^* se e somente se $a_t/b_t > R^*$.*

Utilize este lema para projetar os algoritmos abaixo descrito para encontrar R∗ e S∗.

1) O primeiro algoritmo inicia com R = a0/b0 e testa repetidamente se existe algum par (ak, bk) que satisfaz as condições do lema. No caso afirmativo, inclui o par no conjunto S, atualiza o valor de R e repete o teste. Observe que se existir um elemento em S que não satisfaz às condições do lema, este elemento deve ser removido. Este primeiro algoritmo deve executar em O(n^2).

2) Que relação tem o PPH com o problema de ordenação?
   Utilize esta observação para projetar algoritmos com as complexidades:
   i) O(n^2) sendo eles:
      a) Bubble sort
      b) Insertion sort
      c) Selection sort
   ii) O(nlogn) sendo eles:
      a) Merge-sort
      b) Quick-sort
      c) Heap-sort

3) Observe novamente a relação do PPH com o problema de ordenação. O que caracteriza o conjunto S∗ ? Esta caracterização permite projetar um algoritmo de complexidade O(n)? Apresente este algoritmo caso seja possível.

4) Considere que o seu algoritmo do item (2) utiliza particionamentos em sequência com pivot calculado apropriadamente para garantir a complexidade O(n). Utilize agora como pivot o valor calculado pela expressão:

$$pivot = \frac{a_0 + \sum_{t \in K} a_t}{b_0 + \sum_{t \in K} b_t}$$

   onde K é o conjunto de todos os itens sendo considerados.

a) Estime sua complexidade sobre as instâncias testadas.
b) Apresente experiências computacionais comparativas.

2) **Problema da Mochila Fracionária (pode-se colocar parte de um objeto na mochila)**

[KP-frac] Dado um conjunto de n objetos divisíveis com pesos positivos w_j, $j = 1, \ldots, n$ e valores também positivos v_j, $j = 1, \ldots, n$. Sabendo que uma mochila tem a capacidade W, determinar os objetos que podem ser levados na mochila cuja soma dos valores é máxima.

1. Implementar algoritmos para o problema da mochila fracionária com as seguintes complexidades teóricas em função do número n de itens candidatos a serem colocados na mochila:
a) O(n log n)
b) O(n)
c) Considere que o seu algoritmo do item (b) utiliza particionamentos em sequência com pivot calculado apropriadamente para garantir a complexidade O(n). Utilize agora como pivot o valor calculado pela expressão:

$$pivot = \frac{1}{|K|} \sum_{j \in K} \frac{v_j}{w_j}$$

onde K é o conjunto de itens considerados.

a) Estime sua complexidade sobre as instâncias testadas.
b) Apresente experiências computacionais comparativas.

3. **Problema da Árvore Geradora Mínima**

1. Implementar o Algoritmo de Prim utilizando as estruturas de dados, listadas a seguir, para selecionar o vértice mais próximo da árvore corrente. Nestas estruturas, cada vértice tem como valor-chave o peso da menor aresta que o conecta à árvore corrente. Lista de estruturas de dados a utilizar:
a) Árvore Binária (sem cuidado do balanceamento)
b) AVL
c) Heap de Fibonacci

Organização dos grupos:

1) Formular os problemas
2) * Comunicação dos membros? WhatsApp? Discord? Email? Trello?
3) * Compartilhamento de arquivos: GitHub? Drive? Overleaf?
4) Organizar dentro do grupo as atividades:

a) * Escolher a linguagem: Java? Python? JavaScript?
b) Ler as instâncias:
c) * Como computar as métricas: tempo de CPU e memória.
d) * Metodologia para rodar as instâncias: Quais PCs vão rodar o trabalho (AWS, AZURE)? Quantidade de execuções?
e) Escolher a forma para criar gráficos: metodologia para criar os gráficos?
f) Criar as tabelas segundo o enunciado
g) * Escolher quem vai cuidar do relatório:Tópicos do relatório?
h) * Sobre a implementação do código-fonte, como será?
i) Criação da apresentação, slides

Bom trabalho para todos!