Projeto e Análise de Algoritmos (2022-1)

Segunda Avaliação

Prof. Flávio L. C. de Moura

20 de julho de 2023

- Por favor, coloque **nome** e **matrícula** em todas as folhas
- A resolução pode ser feita à lápis ou caneta, mas seja organizado.
- Esta avaliação é **individual** e **sem consulta**.
- Início: 19:00
- Término: 20:40
- Segue o enunciado do Teorema Mestre:

  Teorema 1. *Sejam $a \geq 1$ e $b > 1$ constantes, $f(n)$ uma função assintoticamente positiva, e $T(n)$ definida nos inteiros não-negativos pela recorrência:*

  $$T(n) = a.T(n/b) + f(n)$$

  onde n/b deve ser interpretado como $\lfloor n/b \rfloor$ ou $\lceil n/b \rceil$. Então $T(n)$ tem as seguintes cotas assintóticas:

  1. *Se $f(n) = O(n^{\log_b a - \epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) = \Theta(n^{\log_b a})$.*
  2. *Se $f(n) = \Theta(n^{\log_b a})$, então $T(n) = \Theta(n^{\log_b a}.\lg n)$.*
  3. *Se $f(n) = \Omega(n^{\log_b a + \epsilon})$ para alguma constante $\epsilon > 0$, e se $a.f(n/b) \leq c.f(n)$ para alguma constante $c < 1$, então para todo n suficientemente grande, temos que $T(n) = \Theta(f(n))$.*

1. (2.5 pontos) O pseudocódigo para o algoritmo recursivo de busca binária em um vetor ordenado de inteiros com n elementos é dado a seguir:

```
1  if high < low then
2  |  return -1;
3  end
4  mid = ⌊(high + low)/2⌋;
5  if key > A[mid] then
6  |  return BinarySearch(A, mid + 1, high, key);
7  end
8  else
9  |  if key < A[mid] then
10 |  |  return BinarySearch(A, low, mid − 1, key);
11 |  end
12 |  else
13 |  |  return mid;
14 |  end
15 end
```

Algoritmo 1: BinarySearch($A[1..n], low, high, key$)

   (a) (1.0 ponto) Escreva a equação de recorrência que representa o custo $T(n)$, no pior caso, para encontrar um elemento em um vetor ordenado contendo n elementos.

(b) (1.5 pontos) Resolva a recorrência do item anterior e faça a análise assintótica do algoritmo BinarySearch no pior caso.

Solução:

(a) $T(n) = T(n/2) + \Theta(1)$

(b) Como $n^{\log_2 1} = 1$, temos que o caso 2 do Teorema Mestre pode ser aplicado. Logo $T(n) = \Theta(\lg n)$.

2. (2.0 pontos) Considere o pseudocódigo a seguir, e faça a análise assintótica da complexidade deste algoritmo.

```
1  let C[0..h − l] be a new array;
2  for i = 0 to h − l do
3  |  C[i] ← 0;
4  end
5  for i = 0 to n − 1 do
6  |  C[A[i] − l] ← C[A[i] − l] + 1;
7  end
8  for j = 1 to h − l do
9  |  C[j] ← C[j] + C[j − 1];
10 end
11 for i = n − 1 downto 0 do
12 |  j ← A[i] − l;
13 |  B[C[j] − 1] ← A[i];
14 |  C[j] ← C[j] − 1;
15 end
16 return B;
```

Algoritmo 2: counting-sort$(A[0..n-1], l, h)$

Solução:

Denote $k = h - l$, ou seja, k denota o tamanho do intervalo que contém os elementos a serem ordenado. Assim, o laço **for** das linhas 2-4 é executado em tempo $\Theta(k)$, o laço das linhas 5-7 em tempo $\Theta(n)$, o laço das linhas 8-10 em tempo $\Theta(k)$, e por fim o laço das linhas 11-14 em tempo $\Theta(n)$, o que perfaz um total de $\Theta(n+k)$. Assumindo que $k = O(n)$, temos que o tempo de execução de counting-sort é $\Theta(n)$.

3. (2.5 pontos) Considere uma enumeração qualquer $1, 2, \ldots, |V|$ dos vértices de G. A matriz de adjacências $G.A$ de dimensão $|V| \times |V|$ é dada por:

$$G.A[i][j] = \begin{cases} 1, & \text{se } (i,j) \in G.E \\ 0, & \text{caso contrário.} \end{cases} \quad (1)$$

O pseudocódigo a seguir apresenta o algoritmo BFS onde o grafo G é representado por sua matriz de adjacências:

```
1  for i = 1 to |V| do
2  |  i.color ← WHITE;
3  end
4  s.color ← GRAY;
5  Q ← ∅;
6  enqueue(Q, s);
7  while Q ≠ ∅ do
8  |  u ← dequeue(Q);
9  |  for i = 1 to |V| do
10 |  |  if G.A[u][i] = 1 and i.color = WHITE then
11 |  |  |  i.color ← GRAY;
12 |  |  |  enqueue(Q, i);
13 |  |  end
14 |  end
15 end
```

Algoritmo 3: BFS(G, s)

Qual é a complexidade de tempo de BFS neste caso?

Solução:

O laço das linhas 1-3 é executado $|V|$ vezes, enquanto que as operações das linhas 4, 5 e 6 são executadas em tempo constante. O laço das linhas 7-15 é executado enquanto a fila Q possuir algum elemento. A fila é iniciada com o vértice s que é removido da fila na linha 8, quando o laço FOR (linhas 9-14) percorre toda a linha da matriz de adjacências $G.A$ correspondente ao vértice s, ou seja, é executado $|V|$ vezes. Cada vértice desta linha, ou seja, cada vértice adjacente a s é marcado como visitado (linha 11) e inserido na fila Q (linha 12). Em seguida o laço FOR é executado novamente para o vértice que foi removido da fila. Note que cada vértice é inserido na fila uma única vez depois que ele é marcado como visitado (GRAY), e portanto o laço FOR é executado uma vez para cada um dos $|V|$ vértices visitados. Como cada execução percorre toda a linha da matriz, o custo total do laço WHILE é quadrático. Assim, podemos concluir que a complexidade de BFS neste caso é $O(V^2)$.

4. Mostre que 3-SAT $\leq_p$ CLIQUE, isto é, mostre que 3-SAT é redutível polinomialmente a CLIQUE considerando a função que transforma instâncias de 3-SAT em instâncias de CLIQUE da seguinte forma:

   Seja φ uma fórmula contendo k cláusulas da seguinte forma:

   $$\varphi = (a_1 \vee b_1 \vee c_1) \wedge (a_2 \vee b_2 \vee c_2) \wedge \ldots \wedge (a_k \vee b_k \vee c_k)$$

   A função de redução recebe φ como argumento e constrói o par $\langle G, k \rangle$, onde G é um grafo com a seguinte estrutura: os vértice de G são organizados em k grupos de três vértices cada, digamos $t_1, t_2, \ldots, t_k$. Cada tripla t_i $(1 \leq i \leq k)$ corresponde a uma cláusula de φ, e cada vértice da tripla corresponde a um literal da cláusula associada. Marcamos cada vértice com o nome do literal correspondente. Cada aresta de G conecta todos os vértices de G, excetuando-se os seguintes casos:

   - vértices que pertencem à mesma tripla;
   - vértices cujos nomes são contraditórios, como x e $\overline{x}$

   A prova de que a função acima é, de fato, uma redução polinomial é dividida em duas partes:

   (a) (1.5 pontos) Mostre que se φ é satisfatível então o grafo G possui um k-clique.
   Solução: Suponha que φ seja satisfatível, isto é, existe um designação de variáveis que faz com que todas as k cláusulas sejam verdadeiras. Em cada tripla de G, selecione o vértice correspondente ao literal verdadeiro da cláusula. Se mais de um literal for verdadeiro, então pode-se escolher qualquer um deles. Os vértices de G assim selecionados formam um k-clique. De fato, o número de vértices selecionados é igual a k (um em cada tripla), e cada par de

vértices selecionados está ligado por uma aresta já que eles não pertencem à mesma tripla e não possuem nomes contraditórios já que ambos são verdadeiros na designação dada. Logo G possui um k-clique.

(b) (1.5 pontos) Mostre que se G possui um k-clique então φ é satisfatível.

Solução: Suponha que G possui um k-clique. Como cada vértice do clique pertence a uma tripla diferente, construiremos uma designação de variáveis fazendo com que o literal correspondente a cada vértice do clique seja verdadeiro. Observe que é possível construir tal designação porque dois literais contraditórios não correspondem a vértices distintos do clique. Por fim, esta designação satisfaz φ porque cada tripla contém um vértice do clique, e portanto cada cláusula de φ contém um literal verdadeiro.