Construção e Análise de Algoritmos

Segunda avaliação remota

Parte síncrona (entrega até as 18h de hoje, dia 31/08)

1. **Intercalação in-place**

   Na 1a avaliação remota (parte síncrona), eu acabei esquecendo de colocar o seguinte ítem:

   > *f) Estime a complexidade do algoritmo Mergesort quando ele utiliza o procedimento descrito acima com a sua rotina de intercalação.*

   Agora, você tem a oportunidade de resolver essa questão com o conhecimento adicional que você ganhou nas últimas aulas.

   Caso você deseje resolver a questão utilizando o método das equações de recorrência, você não deve utilizar o teorema mestre, e deve mostrar todas as contas que levam você até a resposta.

2. **Seleção múltipla**

   Na versão original do problema da seleção, nós temos um vetor desordenado `V[1..n]`, e alguém nos pede para localizar o k-ésimo menor elemento desse vetor.

   Uma solução simples para esse problema consiste em ordenar o vetor e retornar o elemento da posição k, mas isso leva tempo $O(n \log n)$.

   Outra solução consiste em realizar particionamentos sucessivos até encontrar o k-ésimo menor elemento. Essa solução resolve o problema em tempo médio $O(n)$, mas no pior caso ela executa em tempo $O(n^2)$.

   Na aula 11, no entanto, nós vimos uma solução um pouquinho mais complicada, que encontra o k-ésimo menor elemento do vetor em tempo $O(n)$ no pior caso.

   Agora, suponha que ao invés de encontrar apenas um elemento, nós precisamos encontrar vários.

   Por exemplo, imagine que você quer encontrar o 3^o menor elemento do vetor, e o 8^o menor elemento, e o 25^o menor elemento.

   Em geral, o problema consiste em encontrar toda uma coleção de elementos especificada pelos números

   $$k_1,\ k_2,\ k_3,\ \ldots,\ k_m$$

   (i.e., o k_1-ésimo menor elemento, o k_2-ésimo menor elemento, e assim por diante.)

   A solução ingênua para o problema consiste em repetir m vezes a execução do seu melhor algoritmo de seleção, o que leva tempo $O(m \cdot n)$.

   Mas, você consegue fazer melhor do que isso?

a) Apresente um algoritmo de divisão e conquista para esse problema, descrevendo em alto nível e de maneira clara

- a estratégia de quebra do problema
- a estratégia de combinação de soluções
- a solução de versões pequenas do problema

Apresente também o pseudo-código do seu algoritmo.

Dica: Você pode utilizar o algoritmo de seleção de um único elemento como subrotina do seu algoritmo.

b) Estime a complexidade do algoritmo que você apresentou no ítem (a).

(Note que a sua resposta vai ser uma função de m e n, como no caso da solução ingênua.)

Dica: Se você tiver dificuldade para trabalhar com a equação de recorrência, examine a árvore de recursão do algoritmo.

Mais especificamente,

- examine a parte do vetor em que cada chamada recursiva está trabalhando
- e examine o que está acontecendo em cada nível da árvore

c) Faz sentido utilizar o seu algoritmo para valores grandes de m?

Por exemplo, $m = 100$, $m = 1000$, $m = \log n$, $m = \sqrt{n}$?

Em todos os casos, você deve assumir que n é muito, muito maior do que m.

3. **Corretude da ordenação por inserção**

Abaixo nós temos a implementação recursiva do algoritmo de ordenação por inserção

```
Procedimento Inserção-Rec ( V[k..n] )    |  Procedimento  enc-Menor ( V[k..n] )
 {                                       |   {
    Se ( k = n )  Retorna                |      menor  <--  k
                                         |      Para  j <-- k+1  Até  n
    aux  <--  enc-Menor ( V[k..n] )      |          Se ( V[j] < V[menor] )
    Troca(aux,k)                         |             menor <-- j
                                         |
    Inserção-Rec ( V[k+1..n] )           |      Retorna (menor)
 }                                       |   }
```

Argumente que essa implementação funciona corretamente, utilizando as ideias da aula 14.