MAC 338 - Análise de Algoritmos

Departamento de Ciência da Computação

Primeiro semestre de 2013

Lista 3

1. Considere o seguinte algoritmo que determina o segundo maior elemento de um vetor $v[1\,..\,n]$ com $n \geq 2$ números positivos distintos.

   Algoritmo Máximo (v, n)
   1. $maior \leftarrow 0$
   2. *segundo_maior* $\leftarrow 0$
   3. **para** $i \leftarrow 1$ **até** n **faça**
   4. **se** $v[i] > maior$
   5. **então** *segundo_maior* $\leftarrow maior$
   6. $maior \leftarrow v[i]$
   7. **senão se** $v[i] >$ *segundo_maior*
   8. **então** *segundo_maior* $\leftarrow v[i]$
   9. **devolva** *segundo_maior*

   Suponha que v é uma permutação de 1 a n escolhida ao acaso dentre todas as permutações de 1 a n, de acordo com a distribuição uniforme de probabilidade. Seja X o número de vezes que a variável *segundo_maior* é alterada (ou seja, o número de execuções das linhas 5 e 8 do algoritmo) numa chamada de Máximo(v, n). Note que X é uma variável aleatória. Calcule o valor esperado de X.

2. Considere o seguinte algoritmo que calcula o maior e o menor elemento de um vetor $v[1\,..\,n]$ com elementos distintos.

   Algoritmo MaiorMenor (v, n)
   1. *maior* $\leftarrow v[1]$
   2. *menor* $\leftarrow v[1]$
   3. **para** $i \leftarrow 2$ **até** n **faça**
   4. **se** $v[i] >$ *maior*
   5. **então** *maior* $\leftarrow v[i]$
   6. **senão se** $v[i] <$ *menor*
   7. **então** *menor* $\leftarrow v[i]$
   8. **devolva** *maior*, *menor*

   Suponha que a entrada do algoritmo é uma permutação de 1 a n escolhida uniformemente dentre todas as permutações de 1 a n.

   Qual é o número esperado de comparações executadas na linha 6 do algoritmo? Qual é o número esperado de atribuições efetuadas na linha 7 do algoritmo?

3. Desenhe a árvore de decisão para o SelectionSort aplicado a $A[1\,..\,3]$ com todos os elementos distintos.

4. **(CLRS 8.1-1)** Qual o menor profundidade (= menor nível) que uma folha pode ter em uma árvore de decisão que descreve um algoritmo de ordenação baseado em comparações?

5. Mostre que $\lg(n!) \geq (n/4) \lg n$ para $n \geq 4$ sem usar a fórmula de Stirling.

6. **(CLRS 8.1-3)** Mostre que não há algoritmo de ordenação baseado em comparações cujo consumo de tempo é linear para pelo menos metade das $n!$ permutações de 1 a n. O que acontece se trocarmos "metade" por uma fração de $1/n$? O que acontece se trocarmos "metade" por uma fração de $1/2^n$?

7. **(CLRS 8.2-1)** Simule a execução do COUNTINGSORT usando como entrada o vetor

$$A[1\,..\,11] = \langle 6, 0, 2, 0, 1, 3, 4, 6, 1, 3, 2 \rangle.$$

8. **(CLRS 8.2-2)** Mostre que o COUNTINGSORT é estável.

9. **(CLRS 8.2-3)** Suponha que o **para** da linha 7 do COUNTINGSORT é substituído por

para $j \leftarrow 1$ **até** n **faça**

Mostre que o COUNTINGSORT ainda funciona. O algoritmo resultante continua estável?

10. **(CLRS 8.2-4)** Descreva um algoritmo que, dados n inteiros no intervalo de 1 a k, preprocesse sua entrada e então responda em $\mathrm{O}(1)$ qualquer consulta sobre quantos dos n inteiros dados caem em um intervalo $[a\,..\,b]$. O preprocessamento efetuado pelo seu algoritmo deve consumir tempo $\mathrm{O}(n + k)$.

11. **(CLRS 8.3-2)** Quais dos seguintes algoritmos de ordenação são estáveis: insertionsort, mergesort, heapsort, e quicksort. Descreva uma maneira simples de deixar qualquer algoritmo de ordenação estável. Quanto tempo e/ou espaço adicional a sua estratégia usa?

12. **(CLRS 8.3-4)** Mostre como ordenar n inteiros no intervalo de 0 até n^2-1 em tempo $\mathrm{O}(n)$.

13. **(CLRS 8.4-1)** Simule a execução do BUCKETSORT com o vetor

$$A[1\,..\,10] = \langle 0.79, 0.13, 0.16, 0.64, 0.39, 0.20, 0.89, 0.53, 0.71, 0.42 \rangle.$$

14. **(CLRS 8.4-2)** Qual é o consumo de tempo de pior caso para o BUCKETSORT? Que simples ajuste do algoritmo melhora o seu pior caso para $\mathrm{O}(n \lg n)$ e mantem o seu consumo esperado de tempo linear.

15. **(CLRS 8.4-3)** Seja X uma variável aleatória que é igual ao número de caras em duas jogadas de uma moeda justa. Quanto vale $E[X^2]$? Quanto vale $E[X]^2$?

16. Escreva a versão melhorada do heapsort sugerida em aula.

17. Na aula, foram dadas sugestões para melhorar o desempenho prático do mergesort, do quicksort e do heapsort. Implemente cada uma das melhoras sugeridas e faça um estudo experimental de quanto diminui em média o tempo de execução de cada um destes algoritmos no seu computador, quando cada uma das melhoras é incorporada, e cada combinação possível delas é implementada. Faça um breve relatório contando as suas conclusões sobre o seu estudo experimental.