MAC 338 - Análise de Algoritmos

Departamento de Ciência da Computação
Primeiro semestre de 2011

Lista 7

1. Considere um conjunto de livros numerados de 1 a n. Suponha que o livro i tem peso p_i e que $0 < p_i < 1$ para cada i.

   Problema: Dado n e os números $p_1, \ldots, p_n$, acondicionar os livros no menor número possível de envelopes de modo que cada envelope tenha no máximo 2 livros e o peso do conteúdo de cada envelope seja no máximo 1.

   Escreva um algoritmo guloso que resolva o problema em tempo $O(n \log n)$. (Sugestão: comece por escrever um algoritmo recursivo que apenas calcula o número mínimo de envelopes.) Aplique seu algoritmo a um exemplo interessante. Mostre que seu algoritmo está correto.

2. **(CLRS 16-4)** Seja $1, \ldots, n$ um conjunto de *tarefas*. Cada tarefa consome um dia de trabalho; durante um dia de trabalho somente uma das tarefas pode ser executada. Os dias de trabalho são numerados de 1 a n. A cada tarefa t está associado um *prazo* p_t: a tarefa deveria ser executada em algum dia do intervalo $1\,..\,p_t$. A cada tarefa t está associada uma *multa* não-negativa m_t. Se uma dada tarefa t é executada depois do prazo p_t, sou obrigado a pagar a multa m_t (mas a multa não depende do número de dias de atraso).

   Problema: Programar as tarefas (ou seja, estabelecer uma bijeção entre as tarefas e os dias de trabalho) de modo a minimizar a multa total.

   Escreva um algoritmo guloso para resolver o problema. Prove que seu algoritmo está correto. Analise o consumo de tempo.

3. A entrada é uma seqüência de números $x_1, x_2, \ldots, x_n$ onde n é par. Projete que particione a entrada em $n/2$ pares da seguinte maneira. Para cada par, computamos a soma de seus números. Denote por $s_1, s_2, \ldots, s_{n/2}$ as $n/2$ somas. O algoritmo deve encontrar uma partição que minimize a máximo das somas e deve ser tão eficiente quanto possível. Explique porque ele funciona e determine a sua complexidade.

4. Um *código* (binário) para um alfabeto é uma função que associa a cada letra do alfabeto uma cadeia binária única — o código da letra. Um código é *de prefixo* se nenhuma letra é associada a um prefixo do código de uma outra letra do alfabeto. Note que é fácil decodificar um código de prefixos. (É mesmo? Pense um pouco e explique porquê.)

   Problema: Dado um alfabeto A e uma freqüência estimada $f(a)$ para cada letra a em A, determinar um código que minimize

   $$\sum_{a \in A} f(a) l(a),$$

   onde $l(a)$ é o número de bits da cadeia binária associada a letra a.

   Esse problema é elegantemente resolvido pelos chamados *Códigos de Huffman*, que, por sinal, é um código de prefixo. Leia a seção 16.3 do CLRS, sobre *Huffman codes*, entenda o algoritmo de geração dos códigos de Huffman, sua análise de correção e o correspondente algoritmo de decodifição. Se for curioso e implementeiro, implemente esses dois algoritmos e veja qual é a taxa de compressão que ele atinge para comprimir os arquivos textos que você tem na sua área (programas, textos usuais, etc).

5. Use códigos de Huffman para o conjunto de caracteres do enunciado deste exercício. Inclua todos os caracteres. Quantos bits foram economizados no armazenamento do enunciado desse exercício usando códigos de Huffman versus uma codificação onde todos os caracteres são codificados por cadeias de bits do mesmo comprimento?