Lista de exercícios 1

Questão 1 (Notação assintótica)
Resolve os exercícios 2.5 (classificação de funções) e 2.6 (somas parciais) do livro de Kleinberg e Tardos.

Exercício 1 (KT 2.5)
Assume you have functions f and g such that $f(n)$ is $O(g(n))$. For each of the following statements, decide whether you think it is true of false and give a proof or counterexample.

a) $log_2 f(n)$ is $O(\log_2 g(n))$.

b) $2^{f(n)}$ is $O(2^{g(n)})$.

c) $f(n)^2$ is $O(g(n)^2)$.

Exercício 2 (KT 2.6)
Consider the following basic problem. You're given an array A consisting of n integers $A[1], A[2], \ldots, A[n]$. You'd like to output a two-dimensional n-by-n array b in which $B[i, j]$ (for $i < j$) contains the sum of array entries $A[i]$ through $A[j]$ – that is, the sum $A[i] + A[i + 1] + \cdots + A[j]$. (The value of array entry $B[i, j]$ is left unspecified whenever $i \geq j$, so it doesn't matter what is output for these values.)
Here's a simple algorithm to solve this problem.

for $i = 1, 2, \ldots, n$ **do**
  for $j = i + 1, i + 2, \ldots, n$ **do**
    Add up array entries $A[i]$ to $A[j]$
    Store the results in $B[i, j]$
  end for
end for

a) For some function f that you should choose, give a bound of the form $O(f(n))$ on the running time of this algorithm on an input of size n (i.e.,a bound on the number of operations performed by the algorithm).

b) For this same function f, show that the running time of the algorithm on an input of size n is also $\Omega(f(n))$. (This shows an asymptotically tight bound of $\Theta(f(n))$ on the running time.)

c) Although the algorithm you analyzed is parts (a) and (b) is the most natural way to solve the problem – after all, it just iterates through the relevant entries of the array B, filling in a value for each – it contains some highly unnecessary sources of inefficiency. Give a different algorithm to solve this problem, with an asymptotically better running time. In other words, you should design an algorithm with running time $O(g(n))$, where $\lim_{n\to\infty} g(n)/f(n) = 0$.

Questão 2 (Análise de algoritmos)
Resolve os exercícios 1.1, 1.2 (afirmações sobre emparelhamentos estáveis), e 1.3 (competição entre empresas) do livro de Kleinberg e Tardos.

Exercício 3 (KT 1.1)
Decide whether you think the following statement is true or false. If it is true, give a short explanation. If it is false, give a counterexample.

> True or false? In every instance of the Stable Marriage Problem, there is a stable matching containing a pair (m, w) such that m is ranked first on the preference list of w and w is ranked first on the preference list of m.

Exercício 4 (KT 1.2)
Decide whether you think the following statement is true or false. If it is true, give a short explanation. If it is false, give a counterexample.

> True or false? Consider an instance of the Stable Marriage Problem in which there exists a man m and a woman w such that m is ranked first on the preference list of w and w is ranked first on the preference list of m. Then in every stable matching S for this instance, the pair (m, w) belongs S.

Exercício 5 (KT 1.3)
There are many other settings in which we can ask questions related to some type of "stability" principle. Here's one, involving competition between two enterprises.
Suppose we have two television networks, whom we'll call A and B. There are n prime-time programming slots, and each network has n TV shows. Each network wants to devise a *schedule* – an assignment of each show to a distinct slow – so as to attract as much market share as possible.
Here is the way we determine how well two networks perform relative to each other, given their schedules. Each show has a fixed *rating*, which is based on the number of people who watched it last year; we'll assume that no two shows have exactly the same rating. A network *wins* a given time slot if the show that it schedules for the time slow has a larger rating than the show the other network schedules for that time slot. The goal of each network is to win as many time slots as possible.
Suppose in the opening week of the fall season, Network A reveals a schedule S and Network B reveals a schedule T. On the basis of this pair of schedules, each network wins certain time slots, according to the rule above. We'll say that the pair of schedules (S, T) is *stable* if neither network can unilaterally change its own schedule and win more time slots. That is, there is no schedule S' such that network A wins more slots with the pair (S', T) than it did with the pair (S, T); and symmetrically, there is no schedule T' such that Network B wins more slots with the pair (S, T') than it did with the pair (S, T).
The analogue of Gale and Shapley's question for this kind of stability is the following: For every set of TV shows and ratings, is there always a stable pair of schedules? Resolve this question by doing one of the following things:

a) give an algorithm that, for any set of TV shows and associated ratings, produces a stable pair of schedules; or

b) give an example of a set of TV shows and associated ratings for which there is no stable pair of schedules.

Questão 3 (Análise de algoritmos)
Resolve os exercícios 2.7 (folk songs) e 2.8 (stress test of jars) do livro de Kleinberg e Tardos.

Exercício 6 (KT 2.7)
There's a class of folk songs and holiday songs in which each verse consists of the previous verse, with one extra line added on. "The Twelve Days of Christmas" has this property; for example, when you get to the fifth verse, you sing about the five golden rings and then, reprising the lines from the fourth verse, also cover the four calling birds, the three French hens, the two turtle doves, and of course the partridge in the pear tree. The Aramaic song "Had gadya" from the Passover Haggadah works like this as well, as do many other songs.
These songs tend to last a long time, despite having relatively short scripts. In particular, you can convey the words plus instructions for on of these songs by specifying just the new line that is added in each verse, without having to write out all the previous lines each time. (So the phrase "five golden rings" only has to be written once, even though it will appear in verses five and onward.)
There's something asymptotic that can be analyzed here. Suppose, for concreteness, that each line has a length that is bounded by a constant c, and suppose that the song, when sung out load, runs for n words total. Show how to encode such a song using a script that has length $f(n)$, for a function $f(n)$ that grows as slowly as possible.

Exercício 7 (KT 2.8)
You're doing some stress-testing on various models of glass jars to determine the height from which they can be dropped and still not break. The setup for this experiment, on a particular type of jar, is as follows. You have a ladder with n rungs, and you want to find the highest rung from which you can drop a copy of the jar and not have it break. We call this the *highest safe rung*.

It might be natural to try binary search: drop a jar from the middle rung, see if it breaks, and the recursively try from rung $n/4$ or $3n/4$ depending on the outcome. But this has the drawback that you could break a lot of jars in finding the answer.
If your primary goal were to conserve jars, on the other hand, you could try the following strategy. Start by dropping a jar from the first rung, then the second rung, and so forth, climbing one higher each time until the jar breaks. In this way, you only need a single jar – at the moment it breaks, you have the correct answer – but you may have to drop it n times (rather than $\log n$ as in the binary search solution).
So here is the trade-off; it seems you can perform fewer drops if you're willing to break more jars. To understand better how this trade-off works at a quantitative level, let's consider how to run this experiment given a fixed "budget" of $k \geq 1$ jars. In other words, you have to determine the correct answer – the highest safe rung – and can use at most k jars in doing so.

a) Suppose you are given a budget of $k = 2$ jars. Describe a strategy for finding the highest safe rung that requires you to drop a jar at most $f(n)$ times, for some function $f(n)$ that grows slower than linearly. (In other words, it should be the case that $\lim_{n\to\infty} f(n)/n = 0$.)

b) Now suppose you have a budget of $k > 2$ jars, for some given k. Describe a strategy for finding the highest safe rung using at most k jars. If $f_k(n)$ denotes the number of times you need to drop a jar according to your strategy, then the functions $f_1, f_2, f_3, \ldots$ should have the property that each grows asymptotically slower than the previous one: $\lim_{n\to\infty} f_k(n)/f_{k-1}(n) = 0$ for each k.

Data de entrega: 12/03/2011.

Regras para listas de exercícios

1. Os exercícios podem ser resolvidos em colaboração com outros, mas a entrega é individual informando os eventuais colaboradores.
2. A entrega é eletrônica, não escrito a mão, em formato PDF.
3. Para receber pontos as respostas devem ser justificadas (i.e. provadas).
4. Somente entregam respostas que vocês sabem explicar pessoalmente.