Diskrete Mathematik II WS 2012 / 2013
Tibor Szabó

Exercise Sheet 1

Due date: Oct 23rd, 12:30 PM, beginning of exercises
NO LATE SUBMISSIONS!

You should try to solve and write up all the exercises. You are welcome to submit **at most** two neatly written exercises each week. You are encouraged to submit in pairs, but don't forget to mark the name of the scriber.

Exercise 1.

(a) For each pair of expressions (A, B) below, determine whether A is O, o, Ω, ω or Θ of B (recall that A $= \omega(B)$ iff $B = o(A)$).

A	B
$\log(n!)$	$\lceil \log n \rceil !$
$n!$	n^n
$\log(n!)$	$\log(n^n)$
$(\log_3(\log_2 n))^{\ln n}$	n^e

(b) Argue formally from the definition that the $O(\cdot)$ is a transitive relation; that is, show that if $f(n) = O(g(n))$ and $g(n) = O(h(n))$, then $f(n) = O(h(n))$.

Exercise 2 (Binary search).

For this exercise we assume that we are given a vector A of n integer numbers, which is stored in a continous sequence of memory locations in a computer. We assume that accessing the i-th element $A[i]$ of the vector takes one unit of time, for any $1 \leq i \leq n$, and any comparison or algebraic operation between two numbers (such as addition, multiplication, division, etc.) also takes one unit of time.

Most importantly, we assume that the elements of the vectors are *increasing* integers, that is, $A[i] \leq A[i+1], 1 \leq i < n$.

Now we are given an integer x, and we would like to find out if there exists some $1 \leq i \leq n$ such that $A[i] = x$.

Consider the following recursive procedure for solving this problem.

At every step, we consider given as parameters x and the vector A, and we perform the following.

If A is empty, we announce that the number x does not belong to A.

Otherwise, we let $m := \lfloor (n+1)/2 \rfloor$.

If $x = A[m]$, we stop and announce that x belongs to A.

If $x > A[m]$, we recursively apply the same procedure to x and $A[m+1 \dots n]$. Here $A[m+1 \dots n]$ is the vector consisting of the last $n-m$ numbers from A. Note that this vector has also as elements increasing integers, it is stored in the memory as a continous sequence, and accessing any element of it takes constant time.

If $x < A[m]$, we recursively apply the same procedure to x and $A[1 \dots m-1]$. Here $A[1 \dots m-1]$ is the vector consisting of the first $m-1$ numbers from A.

Argue that the procedure is correct and always finishes. Define a function $f(n)$ (where n is the size of the vector) which represents the running time of the procedure in the worst case on a vector of size n and some x satisfying the assumptions, and determine $f(n)$ asymptotically.

Exercise 3.

Show by example that if we don't assume the triangle inequality for the weight function, then the tour found by the Tree Shortcut Algorithm can be longer than 1000 times the optimum tour.

Exercise 4.

The following algorithm, called `Cut`(G), creates a partition of the vertex set of the input graph G.

The algorithm starts by setting $A = V(G) = \{v_1, \dots, v_n\}$ and $B = \emptyset$. Then it iterates the following. One by one for each vertex it checks whether its degree into its own part is greater than to the other part. That is, if $v_i \in A$ for example, then the algorithm checks whether $d_A(v_i) > d_B(v_i)$.[1] If it is so, then `Cut`(G) moves v_i over from part A to part B. Otherwise it does nothing (that is, it goes on to check the next vertex). When the degree of each vertex to its own part is not larger than its degree to the other part, then `Cut`(G) outputs the sets A and B.

(a) Prove that `Cut`(G) terminates in finite time, and at the end the two sets A and B are disjoint. Argue that the running time of the algorithm is $O(mn)$, where m is the number of edges of G.

(b) The **Max-Cut problem** is the following: Given a graph G, find a partition of $V(G)$ into A and B such that $e(A, B)$ (the number of edges with one endpoint in A and the other in B) is as large as possible. Explain why the above algorithm is a $1/2$-approximation algorithm for the **Max-Cut problem**. (That is, the output of the algorithm is always at least $1/2$ of the optimal solution.)

Exercise 5.

Let G be a graph. Recall that $\alpha(G)$ denotes the size of largest independent set in G. A set $L \subseteq E(G)$ of edges is called an *edge cover* if every vertex of G is an endpoint of some member of L. Let $\beta'(G)$ denote the minimum size of an edge cover of G. Prove that $\beta'(G) \geq \alpha(G)$.

[1]If G is any graph, $S \subseteq V(G)$ and $x \in V(G)$, we shall denote by $d_S(x)$ the number of vertices in S adjacent to x in G.