Homework 2
Due Oct 30, 2003

Amit Chakrabarti
Department of Computer Science
Dartmouth College

As usual, please think carefully about how you are going to organise your proofs *before* you begin writing. You shouldn't need more than a page for each solution, but we shall relax things a bit for problems 1 and 6.

Warning: This homework is fairly challenging. Start early!

1. Treating an n-bit vector $\vec{x}$ as a string (so that the ordering of the variables is important) we can define a Boolean function $A_n^s : \{0,1\}^n \to \{0,1\}$ as follows: $A_n^s(\vec{x}) = 1$ iff $\vec{x}$ contains s as a substring. Here s is a fixed string.

   1.1. Show that A_5^{111} and A_6^{111} are not evasive, i.e. it is possible to compute these functions without ever having to look at every input bit.

   1.2. Show that A_3^{111} and A_4^{111} are evasive.

   1.3. Prove that A_n^{111} is evasive iff $n \equiv 0$ or $3 \pmod 4$. Hint: For the "if" direction, use a recurrence for the number of strings of length n that satisfy A_n^{111}, and prove that sometimes just looking at this number tells you that the function is evasive. For the "only if" direction, use the ideas from your solution to 1.1, plus induction.

   1.4. Find all integers n for which A_n^{100} is evasive. Hint: Consider n mod 3.

2. Let f be a bipartite graph property, i.e., a Boolean function on mn Boolean variables $\{x_{ij} : 1 \le i \le m, 1 \le j \le n\}$ that is invariant under permutations of the variables that preserve the bipartition. This problem walks you through Yao's proof that, if f is nontrivial and monotone, then f is evasive.

   Call i the left index and j the right index of variable x_{ij}. Let σ be a permutation that keeps the left index fixed and adds $1 \pmod n$ to the right index. Let Γ be the permutation group generated by σ.

   2.1. As discussed in class, the action of Γ partitions the mn variables into orbits. How many orbits? What does each orbit look like?

   2.2. Recall that a face of the fixed-point complex $\Delta_{f,\Gamma}$ is a set of orbits such that setting all variables inside those orbits to 1 and setting variables outside to 0 gives an assignment $\vec{x}$ such that $f(\vec{x}) = 0$. Based on what you showed above, what do the faces of $\Delta_{f,\Gamma}$ look like?

   2.3. Recall that the Euler characteristic $\chi(\Delta)$ of an abstract complex Δ is defined by

   $$\chi(\Delta) = \sum_{\emptyset \ne F \in \Delta} (-1)^{|F|-1} = \sum_{i=1}^{\infty} (-1)^{i-1} \cdot \#\{\text{faces of size } i\}\,.$$

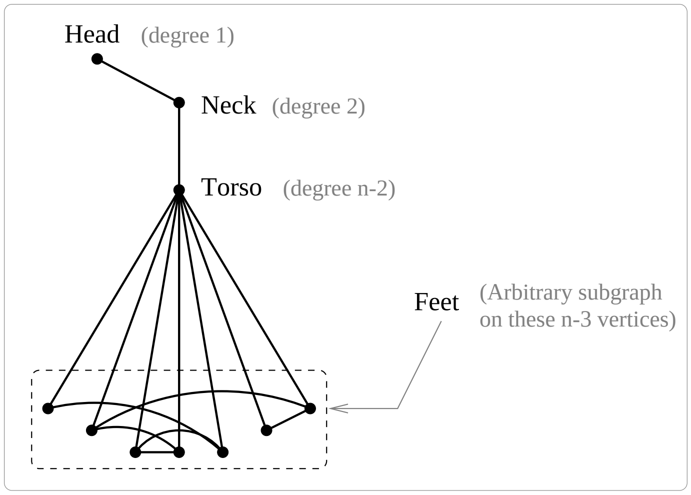
   Based on your work above, write out an expression for $\chi(\Delta_{f,\Gamma})$.

   2.4. Simplify the above expression and figure out when it can equal 1. Using the topological method, conclude that f is evasive.

3. Even while doing research on lower bounds one often has to prove upper bounds, if only to provide counterexamples for plausible but false lower bound conjectures. In the early 1970's it was conjectured that any nontrivial graph property f_n on n-vertex graphs has $\mathrm{D}(f_n) = \Omega(n^2)$. The Rivest-Vuillemin theorem proves this for monotone f_n, but what about non-monotone properties?

   Call an n-vertex graph a *scorpion*[*] if it has the structure shown in the following figure.

[*] This is the historical name for such graphs. I couldn't think of a better name, so I went ahead and called it a scorpion.



Let f_n be the property of being a scorpion.

3.1. Argue that f_n is not monotone.

3.2. Design an algorithm that computes f_n while querying at most $6n$ of the $\binom{n}{2}$ Boolean variables representing the possible edges of the n-vertex graph. This shows that far from having an $\Omega(n^2)$ lower bound, we have an upper bound: $\mathrm{D}(f_n) \leq 6n = O(n)$.

Hint: Once you find out which vertex is the torso, it is easy to check if the graph is a scorpion.

4. The *extreme points problem* asks whether the convex hull of n given points in the plane has n vertices; note that as this is an easier problem than computing the convex hull, the convex hull lower bound does not apply to it directly.

   Model this problem as a set recognition problem for an appropriate set $W \subseteq \mathbb{R}^{2n}$. Prove that $\#W \geq (n-1)!$ and conclude that the algebraic computation tree complexity of the problem is $\Omega(n \log n)$.

5. Let $\mathbf{a_1}, \ldots, \mathbf{a_k}$ and $\mathbf{b}$ be fixed nonzero vectors in $\mathbb{R}^n$ such that the system of inequalities $\{\mathbf{a_i} \cdot \mathbf{x} \geq 0,\ i = 1, \ldots, k\}$ is feasible and implies the inequality $\mathbf{b} \cdot \mathbf{x} \geq 0$. Then, it can be shown that $\mathbf{b}$ is a non-negative linear combination of the $\mathbf{a_i}$'s, i.e., $\mathbf{b} = \sum_{i=1}^{k} \lambda_i \mathbf{a_i}$ for some non-negative reals λ_i. This fact is sometimes known as Farkas's Lemma.

   Using Farkas's Lemma, prove the following two lower bounds in the *linear* decision tree model.

   5.1. The complexity of finding the largest of n given reals is $n-1$.

   5.2. The complexity of finding the second largest is at least $n - 2 + \log n$.

   Hint: Once you have solved the first subproblem, use what you learnt along with a leaf counting argument to solve the second.

6. Give a clear proof of Theorem 9 in Ben-Or's paper (Handout 4). This is a relatively easy exercise since you don't have to invent any new techniques, but it is important to get a feel for formalizing a model of computation and writing up a clear proof for a theorem that is "obvious" to you.