

# Probability and Computing: Projects

**Due:** exact date will be decided later (sometime during last two weeks of the semester)

**Project 1** The first project is the exploratory assignment in MU Section 5.8.

**Project 2** Let  $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$  be a black-box function with the following promise

$$f(x) = f(x') \text{ if and only if } x = x' \oplus s$$

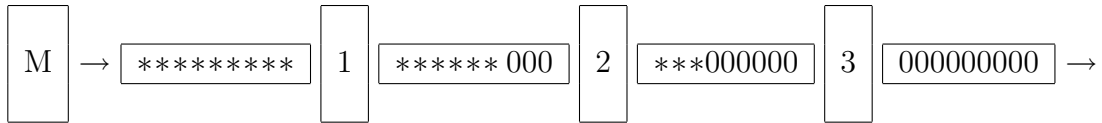
where  $00 \dots 0 \neq s \in \{0, 1\}^n$  is the unknown string to be determined ( $\oplus$  denotes the bitwise XOR). Derive a lower bound on the number of queries that any algorithm has to make to the black-box in order to correctly determine the hidden string  $s$  with probability  $c > 0$ . Use game-theoretic techniques, which enable us to think of a randomized algorithm as a probability distribution on deterministic algorithms, and Yao's Minmax Principle (these topics will be presented in class; see Chapter 2 "Game-theoretic techniques" in the book *Randomized Algorithms* by R. Motwani and P. Raghavan, Cambridge University Press).

**Project 3** (Bonus) Consider the following scenario. We are dealing with objects that are labelled by binary strings of length  $n$ . Our goal is to obtain an object whose label is  $00 \dots 0$ . These objects are created by a machine that we have at our disposal. This machine outputs both an object and its label. We are guaranteed that the machine outputs objects whose label is chosen uniformly at random from  $\{0, 1\}^n$ . Each time we ask the machine for a new object we pay one time unit. So here is a first algorithm: call the machine repeatedly until it happens to output an object whose label is  $00 \dots 0$ . Clearly, this algorithm has expected running time  $O(2^n)$ .

Assume that these objects have the following nice property: given two objects, labelled by  $x, y \in \{0, 1\}^n$ , we can *combine* them and obtain a new object whose label is  $x \oplus y$  (where  $\oplus$  denotes the bitwise XOR) and the two original objects are gone. This combination operation succeeds with probability 50%; with probability 50%, the operation fails and then both original objects are gone.

This property makes it possible to design a better algorithm. Try to find an algorithm whose expected running time is  $2^{O(n^\alpha)}$  for  $\alpha < 1$ .

**Hints:** Assume that  $n = k^2$ . The overall structure of the algorithm is that of a pipeline consisting of  $k$  stages. This means that the input to the  $(i + 1)$ th stage is the output of the  $i$ th stage. The inputs to stage 1 are objects directly created by the machine, that is, objects that are chosen uniformly at random. For  $i = 1, \dots, k$  the outputs of stage  $i$  (and the inputs to stage  $(i + 1)$ ) are objects whose labels have the following distribution: the  $ik$  least significant bits are all equal to 0 and the remaining  $n - ik$  bits are chosen uniformly at random. This structure is depicted below for  $n = 9$ :



Show how to implement this pipeline structure using the ‘combine’ operation and determine the expected number of objects that have to be created.