CISC-365
2009
Lab # 4
Week 4

The Chip-R-Us Company produces computer chips. Due to shoddy algorithm design, many of the chips are faulty.

The chips have built-in diagnostic capability, but since it is obviously unwise to ask a potentially faulty chip to test itself, the chips are to be tested in pairs – each running its diagnostic on the other.

Good chips always correctly determine if their testing partner is good or faulty. A faulty chip may or may not give a correct answer when asked to diagnose another chip.

Thus if we are testing chip A against chip B, there are four possible outcomes and associated conclusions.

Chip A says ...	Chip B says ...	Conclusion
A says B is good	B says A is good	Both are good or Both are faulty
A says B is good	B says A is faulty	At least one of them is faulty.
A says B is faulty	B says A is good	At least one of them is faulty.
A says B is faulty	B says A is faulty	At least one of them is faulty.

A faulty chip will not change its answer if tested twice against the same chip.

The company wants to identify all the good chips (and sell the others to Dull Inc.) They have only one thing going for them – **they know that on any given day, *more than half* the chips produced are good.**

Create and implement a divide-and-conquer algorithm for solving this problem. Your algorithm should proceed in two stages:

1. Find one good chip
2. Use that chip to identify all the other good chips

First, we need to examine how the chips should be modeled. Each chip can be represented by

a single integer, with 1 representing “good” and 0 representing “faulty”. Of course we can't simply look at a chip's integer ... we have to ask another chip to look at it for us (as explained above). If the chip that is doing the checking is good (i.e. its integer is 1) then it will tell us the truth about the chip being checked. If the chip that is doing the checking is faulty (i.e. its integer is 0) then it may or may not tell us the truth about the chip being checked. As we will see later, it is never necessary to test the same pair more than once so there is no need for faulty chips to remember what they said about other chips.

Since Step 2. is trivial, let's think about Step 1. Our goal in a Divide and Conquer algorithm is to reduce the size of the problem by a factor of 2 or more after each iteration (or level of recursion). I strongly recommend that you take some time now to think about how we can use n/2 pairwise comparisons to reduce the size of the problem to at most half the original size. After you have thought about this, follow this link:
http://research.cs.queensu.ca/home/cisc365/2008F/Labs/Lab_4/Lab_4_Chips_Part2.pdf