**3-SUM in quadratic time.** Design an algorithm for the 3-SUM problem that takes time proportional to *n²* in the worst case. You may assume that you can sort the *n* integers in time proportional to *n²* or better.

**Search in a bitonic array.** An array is *bitonic* if it is comprised of an increasing sequence of integers followed immediately by a decreasing sequence of integers. Write a program that, given a bitonic array of *n* distinct integer values, determines whether a given integer is in the array.

* Standard version: Use ∼3lg*n* compares in the worst case.
* Signing bonus: Use ∼2lg*n* compares in the worst case (and prove that no algorithm can guarantee to perform fewer than ∼2lg*n* compares in the worst case).

**Egg drop.** Suppose that you have an *n*-story building (with floors 1 through *n*) and plenty of eggs. An egg breaks if it is dropped from floor *T* or higher and does not break otherwise. Your goal is to devise a strategy to determine the value of *T* given the following limitations on the number of eggs and tosses:

* Version 0: 1 egg, ≤*T* tosses.
* Version 1: ∼1lg*n* eggs and ∼1lg*n* tosses.
* Version 2: ∼lg*T* eggs and ∼2lg*T* tosses.
* Version 3: 2 eggs and ∼2*√n* tosses.
* Version 4: 2 eggs and ≤ *c√T*​ tosses for some fixed constant *c*.