**Interview Questions: Analysis of Algorithms (ungraded)**

**总分：{TOTALPOINTS}**

**1.第 1 个问题**

**3-SUM in quadratic time. Design an algorithm for the 3-SUM problem that takes time proportional to n^2*n*2 in the worst case. You may assume that you can sort the n*n* integers in time proportional to n^2*n*2or better.**

***Note: these interview questions are ungraded and purely for your own enrichment. To get a hint, submit a solution.***

1

**Correct**

*Hint:*given an integer x and a sorted array a[] of n*n* distinct integers, design a linear-time algorithm to determine if there exists two distinct indices i and j such that a[i]+a[j]==x.

**1/1 分**

**2.第 2 个问题**

**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*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).**

1

**Correct**

*Hints*: Standard version. First, find the maximum integer using ∼1lg*n* compares—this divides the array into the increasing and decreasing pieces.

Signing bonus. Do it without finding the maximum integer.

**1/1 分**

**3.第 3 个问题**

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

* **Version 0: 1 egg, \le T≤*T* tosses.**
* **Version 1:**∼1lg*n***eggs and**∼1lg*n***tosses.**
* **Version 2:**∼lg*T***eggs and**∼2lg*T***tosses.**
* **Version 3: 22 eggs and**∼2*n*√**tosses.**
* **Version 4: 22 eggs and \le c \sqrt T≤*cT*​ tosses for some fixed constant c*c*.**

1

**Correct**

*Hints:*

* Version 0: sequential search.
* Version 1: binary search.
* Version 2: find an interval containing T*T* of size \le 2T≤2*T*, then do binary search.
* Version 3: find an interval of size \sqrt n*n*​, then do sequential search. Note: can be improved to ∼2*n*−−√ tosses.
* Version 4: 1+2+3+…+*t*∼12*t*2. Aim for c = 2 \sqrt{2}*c*=22​.