

CS 473 (ug): Combinatorial Algorithms, Fall 2005

Head-Banging Session 8

Name:

NetID:

You determine how much extra credit the people you work with get for each session. Each person is to be rated on two criteria: participation and helpfulness. A person should get participation points if he/she is actively involved in working with the group and contributes ideas and suggestions, *regardless of whether they are correct*. A person should get ‘helpfulness’ points if he/she solves the problem quickly and guides the group to a solution or helps other people who are having trouble. **Simply stating ‘X is the solution’ is NOT being helpful**; it defeats the purpose of the head-banging sessions.

If you are working in a group of 4, you have a total of 17 points *for each category* to be distributed among the members of your group. That is, the total number of participation points you award your 3 team members should be less than or equal to 17. (Similarly for helpfulness points.) If you are working in a smaller group, ask the TA how many points you have.

Don’t forget to turn the evaluation sheet in

NetID	Helpfulness	Participation
Total ≤ 17		

Remember to detach this page and hand it in.

1. **Problem 7.20: Measuring Balloons** Your friends are involved in a large-scale atmospheric science experiment. They need to get good measurements on a set S of n different conditions in the atmosphere (such as the ozone level at various places), and they have a set of m balloons that they plan to send up to make these measurements. Each balloon can make at most two measurements.

Unfortunately, not all balloons are capable of measuring all conditions, so for each balloon $i = 1, 2, \dots, m$, they have a set S_i of conditions that balloon i can measure. Finally, to make the results more reliable, they plan to take each measurement from at least k different balloons. (Note that a single balloon should not measure the same condition twice.) They are having trouble figuring out which conditions to measure on which balloon.

(a) Give a polynomial-time algorithm that takes the input to an instance of this problem (the n conditions, the sets S_i for each of the m balloons, and the parameter k), and decides whether there is a way to measure each condition by k different balloons, while each balloon measures at most two conditions.

(b) You show your friends a solution computed by your algorithm from (a), and to your surprise they reply, "This won't do at all - one of the conditions is only being measured by balloons from a single subcontractor." You hadn't heard anything about subcontractors before; it turns out there's an extra wrinkle they forgot to mention. . . .

Each of the balloons is produced by one of three different subcontractors involved in the experiment. A requirement of the experiment is that there be no condition for which all k measurements come from balloons produced by a single subcontractor.

Explain how to modify your polynomial-time algorithm from part (a) into a new algorithm that decides whether there exists a solution satisfying all the conditions from (a), plus the new requirements about sub-contractors.

2. **Problem 7.21: Testing Wireless Networks** You're helping to organize a class on campus that has decided to give its students wireless laptops for the semester. Thus there is a collection of n wireless laptops; there is also a collection of n wireless *access points*, to which a laptop can connect when it is in range.

The laptops are currently scattered across campus; laptop l is within the range of a set S_l of access points. We will assume that each laptop is within range of at least one access point (so the sets S_l are all non-empty); we will also assume that every access point has at least one laptop in range of it.

To make sure that all the wireless connectivity software is working correctly, you need to try having laptops make contact with access points in such a way that each laptop and each access point is involved in at least one connection. Thus we will say that a *Test Set* T is a collection of ordered pairs of the form (l, p) for a laptop l and an access point p , with the properties that:

- (a) If $(l, p) \in T$, then l is within range of p . (That is, $p \in S_l$.)
- (b) Each laptop appears in at least one ordered pair in T .
- (c) Each access point appears in at least one ordered pair in T .

Example. Suppose that $n = 3$; laptop 1 is within range of access points 1 and 2, laptop 2 is within range of access point 2, and laptop 3 is within range of access points 2 and 3. Then, the set of pairs (*laptop 1, access point 1*), (*laptop 2, access point 2*), (*laptop 3, access point 3*) form a test set of size 3.

This way, by trying out all the connections specified by the pairs in T , we can be sure that each laptop and each access point have correctly functioning software.

The problem is: Given the sets S_l for each laptop (i.e., which laptops are within range of which access points), and a number k , decide whether there is a test set of size at most k .

(a) Give an example of an instance of this problem for which there is no test set of size n . (Recall that we assume each laptop is within range of at least one access point, and each access point has at least one laptop within range of it.)

(b) Give a polynomial-time algorithm that takes the input to an instance of this problem (including the parameter k), and decides whether there is a test set of size at most k .

3. **Problem 7.31: Packing Boxes** Some of your friends are interning at the small high-tech company WebExodus. A running joke among the employees there is that the back room has less space devoted to high-end servers than it does to empty boxes of computer equipment, piled up in case something needs to be shipped back to the supplier for maintenance.

A few days ago, a large shipment of computer monitors arrived, each in its own large box; since there are many different kinds of monitors in the shipment, the boxes do not all have the same dimensions. A bunch of people spent some time in the morning trying to figure out how to store all these things, realizing of course that less space would be taken up if some of these boxes could be *nested* inside others.

Suppose each box is a rectangular pallellepiped with side lengths equal to (i_1, i_2, i_3) ; and suppose each side length is strictly between half a meter and one meter. Geometrically, you know what it means for one box to nest inside another: It's possible if you can rotate the smaller so that it fits inside the larger in each dimension. Formally, we can say that box i with dimensions (i_1, i_2, i_3) *nests* inside box j with dimensions (j_1, j_2, j_3) if there is a permutation a, b, c of the dimensions $\{1, 2, 3\}$ such that $i_a < j_1, i_b < j_2, i_c < j_3$. Of course, nesting is recursive: If i nests in j , and j nests in k , then by putting i inside j inside k , only box k is visible. We say that a *nesting arrangement* of a set of n boxes is a sequence of operations in which a box i is put inside another box j in which it nests; if there were already boxes nested inside i , these end up inside j as well. (Also notice the following: Since the side lengths of i are more than half a meter each, and since the side lengths of j are less than a meter each, box i will take up more than half of each dimension of j , and so after i is put inside j , nothing else can be put inside j .) We say that a box k is *visible* in a nesting arrangement if the sequence of operations does not result in its ever being put inside another box.

Here is the problem faced by the people at WebExodus: Since only the visible boxes are taking up any space, how should a nesting arrangement be chosen to minimize the *number* of visible boxes? Give a polynomial-time algorithm to solve this problem.