1. (•) ***I understand the course policies.***

   Describe and analyze an algorithm to nest the boxes so that the number of visible boxes is as small as possible.

Solution: We assume one box can only contain *exactly* one another box. We reduce this problem to bipartite matching problem, and then use Ford-Fulkerson algorithm to solve it. We construct a bipartite graph G with vertex set $U \cup W$ as follows:

- Each box is a node in both U and W.
- For a pair of node $u \in U$ and $v \in W$, if u can be rotated so that it can be contained in v, we add an edge $e = (u, v)$.

Call two boxes are nested if one contains another. Let N be the maximum number of nested boxes. Then, the number of visible boxes can be simply computed as $n - N$.

Theorem 1. *The maximum number of nested boxes equals to $|M|$, where $|M|$ is the value of the size of the maximum matching in G.*

Proof: If we know the maximum number of nested boxes and how the boxes contain each other, we can transfer to a maximum matching in G as follows: if u contains v, then we select $e = (u, v)$ in G, where $u \in U$ and $v \in W$. Each time we place a box into another, the number of visible boxes is decreased by one. Let $|M|$ be the final number of edges selected. It is maximized according to the above procedure. Otherwise, there will be a better way to place the boxes. Since each box can contain exactly one another box, this is a valid matching. Moreover, the number of the nested boxes and the size of the maximum matching is $|M|$.

Conversely, if we know a maximum matching, we can find how to place the boxes into each other and then get the smallest number of visible boxes. If we match u with v, then we place box u into v. Finally, the size of the maximum matching is $|M|$, and equals to the number of nested boxes. □

There are at most six permutation of width, height and depth. Hence for a pair of boxes, we can determine whether they can contain one another in constant time. Let there be n boxes. Then, there are $2n$ nodes and at most $O(n^2)$ edges in G. Hence, the maximum flow has value at most $O(n)$, and the Ford-Fulkerson algorithm runs in $O(n^3)$ time.

■

2. (•) ***I understand the course policies.***

   Describe an efficient algorithm that either rounds A in this fashion, or reports correctly that no such rounding is possible.

   Solution: test ■

3. (•) ***I understand the course policies.***

   Describe and analyze an algorithm to compute a donation schedule, describing how much money each voter should send to each candidate on each day, that guarantees that every candidate gets enough money to win their election. The schedule must obey both Federal laws and individual voters' budget constraints. If no such schedule exists, your algorithm should report that fact.

   Solution: test ■

4. (•) ***I understand the course policies.***

   (a) Prove that this greedy strategy does not always compute an optimal solution.

   Solution: test ■

   (b) Describe and analyze an efficient algorithm to compute the smallest set of monotone paths that covers every marked cell. The input to your algorithm is an array $M[1..n, 1..n]$ of booleans, where $M[i, j]$ = TRUE if and only if cell (i, j) is marked.

   Solution: test ■

5. (•) ***I understand the course policies.***

   (a) Prove that if Paul uses a deterministic strategy, and Sally knows his strategy, then Sally can guarantee that she wins.

   (b) Describe a deterministic strategy for Sally that guarantees that she wins when $r \leq M$, no matter what strategy Paul uses.

   (c) Prove that if Sally uses a deterministic strategy, and Paul knows her strategy then Paul can guarantee that he wins when $r < M$.

   (d) Describe a randomized strategy for Sally that guarantees that she wins with probability at least $\min r/M, 1$, no matter what strategy Paul uses.

   (e) Describe a randomized strategy for Paul that guarantees that he loses with probability at most $\min r/M, 1$, no matter what strategy Sally uses.