Final Exam:
P and NP... DAG Counter RangeTree// 1.(a)T (b)F (c)F (d)T (e)F (f)T (g)T (h)F (i)T (j)F (k)F (l)F (m)T (n)F (o)T (p)F (q)T (r)T 2. (a)1 (b)4 (c)3 3.
(a)The data structure is basic a AVL-tree with ameution at each subtrees' node(stand for a room) that the number of total rooms, occupied rooms and if this room is occupied. The invariant is that after every operation, the features described above is updated in time.
(b) Say the operation NEW-NODE() return a pointer to a node with:
$rooms = 1, ocupied = False, ocupiedrooms = 0$. We create a blank tree T, create new node and insert it to T repeat for N times, the operation just like AVL-tree, but we need to keep track of $rooms$ correctly for both insert and rotation operation.
(c) We count how many available rooms between 1 to l and 1 to h, we do the minus and get the answer. We do search on the tree and keep track the total number of ocupiedrooms of left-subtree and the root every time we go right, and we finally get the answer.
(d) We do search for l and h on T, once we arrive the node that between l and h, we always check if the root,left-subtree,right-subtree has available rooms, if we found available rooms, change our target to find the position of the empty room, modify the node, back and update the nodes along the way. //(e)We simply search x on T, find x, and along the back way up, decrease the $ocupiedrooms$ for each node on the way. 4.
(a) It's $\frac{m-k}{m}$.
(b) It's It's $p(0) + p(1) \times \frac{m-1}{m} + p(2) \times \frac{m-2}{m} ... + p(\frac{n}{2}) \times \frac{m-\frac{n}{2}}{m}$
(c) $p(0) = 1 \times \frac{m-1}{m} \times \frac{m-2}{m} ... \times \frac{m-2}{m} = \frac{m!}{m^n \times (m-n)!}$
5.
We reduce the problem to solve $x^2 + 4x + 4 = 3$, which is $(x+2)^2 = 3$. So we just d digits of precision of $\sqrt{3}$ and then x could be $\sqrt{3} - 2$ or $-\sqrt{3} - 2$.
6.
(a) We build a graph G to solve the problem. Let each vertex represent each guest, and we do BFS on each vertex, once we complete the distance 2 vertices detect, we add a edge between the origin node and every other node detected. Let the new edges set named E. Randomly pick a vertex v on G and do DFS on v and its adjust vertices, return the
(b) We build a new graph G' and solve the problem. We first all vertices in origin graph to G', then for any vertex v, if any other vertex v' doesn't has a edge connect between them, we add a edge connect them in G'. After build the graph, run the same algorithm in (a) and get the number, if the number bigger than 2, return FALSE, and return TURE otherwise.
7.
(a)3 (b)4 (c)3 (d)2 (e)1
8.
(a)
1 2 2 2 4 1
1 1 3 3 3 5
(b)
$DP(i, b) = MAX$(1, $DP(k, !b) + 1$ for k from 1 to i-1
If (b and ($x_i > x_k$)) or (!b and ($x_i < x_k$)))

(c)
$DP(1, TURE) = 0$, $DP(1, FALSE) = 0$
(d)
$DP(1, TURE), DP(1, FALSE), DP(2, TURE), DP(2, FALSE), DP(3, TURE), DP(3, FALSE)...DP(n$
(e) We simply return $MAX(DP(n, TURE), DP(n, FALSE))$.
(f)There are n subproblems in total, each subproblem run in $O(i)$ times, so the total running time is $O(n^2)$.
9.
We define the subproblem is
$DP(i, j) = \{x_i$, If $i == j$,
$Paren(x_i, o_i, ..., x_j)$, otherwise$\}$.

PAREN(i,j)

1 **if** $i == j$ **return** x_i
2 **if** $DP(i, j)! = None$ **return** $DP(i, j)$
3 **for** $k = i$ **to** j
4 $\quad DP(i, k) =$ MAX$(Paren(i, b)\ O_b\ Paren(b, k))$ **for** $b = i$ **to** $k - 1$
5 **return** $DP(i, j)$

There are $O(n^2)$ subproblems, and each subproblem cost $j - i$(average $O(\frac{n}{2})$), so the total running time is n^3.// 10.
(a) If we just have three balls of different weight, we of course can't distinguish between the lightest and the heaviest ball, and MEDIAN can help us to do that.
(b)

LIGHTER(a, b)

1 $median =$ MEDIAN(a, b, l)
2 **if** $median\ is\ a$
3 $\quad$ **return** TURE
4 **return** FALSE

(c)
The median ball will always not be the heaviest or the lightest ball, so
(1)Randomly choose three ball a,b,c
(2) Using MEDIAN to tell which is the median ball, throw the ball away, pick another ball from the left balls
(3) Repeat unitl there are no left balls
(4) The left two balls are the heaviest and the lightest ball.
(d) We use (c) part method to get the the heaviest and the lightest ball, then use LIGHTEST to know which ball is the lightest ball, after that, we could use LIGHTER so it's easy to use merge-sort to get the answer.
(e) Every LIGHTER call will call MEDIAN, so we will prove that we need at least $\Omega(NlogN)$

calls to LIGHTER. Let's think about comparison model, the decision tree, there are $N!$ result of comparison, so we need at lest $log(N!)$ call to LIGHTER, therefore, we need $\Omega(NlogN)$ calls to MEDIAN.

// Need Be More careful boy!