**Question No. 1**

Solution: To solve the given problem adeptly using dynamic programming state and recurrence relation is defined as follows:

State:

State dp [i][j] is defined as the maximum total hatred score for seating the first I guests where the host is in the position j. Here, I ranges from 0 to 2n, and j ranges from 0 to n.

Recursion Relation:

For this dynamic programming approach the recurrence relation is defined as follows:

Here, iteration over the previous seating arrangements (k < j) is being done to find the optimal arrangement for the current existing state. And adding the hatred scores between the current existing guest and the two adjacent guests.

Base Cases:

For i=0, dp [0][1] = 0 for all j.

For j=0, dp [i][0] = ∞ for all I > 0, as the host must be seated somewhere.

Algorithm

|  |
| --- |
| 1. **Initialize the dp table with appropriate sizes.** 2. **Iterate over the number of guests (i) from 1 to 2n.** 3. **Iterate over the host position (j) from 1 to 2n.** 4. **Compute the minimum hatred score for each state dp[i][j] using the recurrence relationship.** 5. **Return the minimum score from dp[2n][j] for all j as the answer.** |

The Analysis of Time Complexity

Initialize the dp table: 0(n^2)

Compute all the cells of ted p table: 0(n)

Total time complexity: 0(n^3)

Bonus O(n3)-time dynamic programming

Our recurrence relation can be modified such that it takes into account all of the possible places for the host in each state. This will allow us to acquire the bonus.

)

The modified recursion relation is shown above that allows considering all the positions for the host at each step which leads to a time complexity of 0(n^3).

**Question No. 2:**

**Solution**

1. Bonus

To show that b ≤ M and a ≤ M with probability at least 1 Lemma 1 can be used.According to the algorithm, a and b are the elements of set S with respective ranks.

Probability that the rank of b is greater than or equal to is at most .

Similarly, the probability that the rank of a is less than or equal to is at most

So, the probability that b is greater than or equal to M and a is less than or equal to the M is at least 1 – 2. For the largest values of n this probability tends to be 1.

b.

The probability that a random element selected from T does not have a T-rank in [k, k + r] is equal to the probability that the rank of this element is either less than k or greater than or equal to k + r. the probability of this event is at most b using lemma 1.

1. a and b are selected randomly from S so the probability that none of the element is S has a T-rank in [k, k + r] is the product of individual probabilities. It is bounded above b (since a and b both are selected randomly).

C

Given that r = [5 ], the interval [ri, r(i + 1)] contains at most r elements for an non negative integer i. Using the result from part (b), the probability that none of the elements in S has a T-rank in this interval is at most . Hence, the probability that at least one element in S has a T-rank in this interval is at least 1 -

d.

From the result of part (a) and (c), we can get that with high probability both a≤M and S’ contains at most 1000 (ln n elements.

e.

From the conditions that have been established in part (d), the algorithm runs in linear time and returns the median of T with probability at least 1 - .