To find $\text{DSS} \left(\{Q_0, Q_1, \ldots, Q_{q-1}\}\right)$ such that $\sum_{i=0}^{q-1} |Q_i| = R_q(n, \rho) = \left\lceil \sqrt{\frac{\rho(n-1)}{q-1}} \right\rceil$ and those subsets yield the multi-set we need, there are two steps of the algorithm:

1. Search all possible partitions of integer $R_q(n, \rho)$ having $q$ parts. For example, $7$ has totally 4 partitions which have 3 parts:
   
   $7 = 5 + 1 + 1$;
   
   $7 = 4 + 2 + 1$;
   
   $7 = 3 + 3 + 1$;
   
   $7 = 3 + 2 + 2$.

Hence if $R_q(n, \rho) = 7$ and $q = 3$, all candidate DSS’s could look like:

$|Q_0| = 5, |Q_1| = 1, |Q_2| = 1$;

$|Q_0| = 4, |Q_1| = 2, |Q_2| = 1$;

$|Q_0| = 3, |Q_1| = 3, |Q_2| = 1$;

$|Q_0| = 3, |Q_1| = 2, |Q_2| = 2$.

Also those subsets have to satisfy $2 \sum_{i=0}^{q-1} |Q_i| \geq \rho(n-1)$

2. DSS should yield the multi-set which contains $i \in \{1, \ldots, n-1\}$ at least $\rho$ time(s). So, the algorithm will check all candidates according to this requirement.

For step 2, it’s easy to accomplish although the complexity of algorithm is really big when $n$ is mild big. And for step 1, we construct a recursive algorithm shown below.

**Algorithm:** GetQP artition($n, q$)

**procedure** RecP artition($n, q, N$)

if $q = 0$

then output([a_0, a_1, ..., a_{q-1}])

else

for $i \leftarrow n - q + 1$ to $\lceil n/q \rceil$

if $N = 0$ or ($N \geq 1$ and $a_N \geq i$) then

$a_{N+1} \leftarrow i$

RecP artition($n-i, q-1, N+1$)

main

RecP artition($n, q, 0$)

In the procedure RecP artition, the values for $a_0, a_1, \ldots, a_N$ have been chosen already. $n$ will be the sum of the values $a_{N+1}, a_{N+2}, \ldots, a_{q-1}$ (which have not been chosen yet). Thus $a_{N+1}$ can take on any value between $n - q + 1$ and $\lceil n/q \rceil$ where $\lceil n/q \rceil$ is the smallest integer not less than $n/q$. If we define $a_{N+1}$ to have the value $i$, then $n - i$ becomes the sum of the remaining integers $a_{N+2}, \ldots, a_{q-1}$.
Also, the value of \( q \) is decreased by one and \( N \) is increased by one, when the procedure \( \text{RecPartition} \) is called recursively. The \( \text{main} \) program requires two input parameters \( n \) and \( q \) to call procedure \( \text{RecPartition} \).

Then we can accomplish the final program to find all possible \( DSS \) with given \( n, q \) and \( \rho \).

**Algorithm:** \( \text{SearchingDSS}(n, q) \)

**procedure** \( \text{RecPartition}(n, q, N) \)

\[
\text{if } q = 0 \\
\quad \text{then if } 2 \sum_{i \neq j} a_i * a_j \geq (n - 1) * \rho \\
\quad \quad \text{then for any } Q_0 \subseteq U \text{ with } |Q_0| = a_0 \\
\quad \quad \quad \text{do for any } Q_1 \subseteq U \setminus Q_0 \text{ with } |Q_1| = a_1 \\
\quad \quad \quad \quad \text{......} \\
\quad \quad \quad \text{do for any } Q_{q-1} \subseteq U \setminus (Q_0 \cup Q_1 \cup \ldots \cup Q_{q-2}) \text{ with } |Q_{q-1}| = a_{q-1} \\
\quad \quad \quad \quad \text{do } M \leftarrow \{ a - b | a \in Q_i, b \in Q_j, i \neq j \} \\
\quad \quad \quad \quad \text{if } M \text{ contains every } i \in \{1, \ldots, n - 1\} \text{ at least } \rho \text{ time(s).} \\
\quad \quad \quad \quad \quad \text{then } \left\{ \begin{array}{l}
\text{output } ([Q_0, Q_1, \ldots, Q_{q-1}]) \\
\text{output } M
\end{array} \right. \\
\quad \quad \quad \quad \quad \text{else } \left\{ \begin{array}{l}
\text{for } i \leftarrow n - q + 1 \text{ to } [n/q] \\
\text{if } N = 0 \text{ or } (N \geq 1 \text{ and } a_N \geq i) \text{ then} \\
N + 1 \leftarrow i \\
\text{RecPartition}(n - i, q - 1, N + 1)
\end{array} \right. \\
\end{array}
\]

**main**

\[
U \leftarrow \{1, 2, \ldots, n\} \\
r \leftarrow \left\lfloor \sqrt{\frac{q(n-1)}{q-1}} \right\rfloor \\
\text{RecPartition}(r, q, 0)
\]

This algorithm searches all \( DSS \)'s, and as mentioned above, the complexity ascends extremely when \( n \) grows up.