You have a network of wireless sensors that you would like to make more reliable, by selecting some number of "backup" sensors for each sensor in the network. Formally, consider the following problem.

Input: Sensors $s_1, s_2, \ldots, s_n$ (each sensor s_i has real-number coordinates (x_i, y_i)), distance parameter $d \in \mathbb{R}^+$, redundancy parameter $r \in \mathbb{Z}^+$, and backup parameter $b \in \mathbb{Z}^+$ with $b \geqslant r$.

Output: Backup sets $B_1, \ldots, B_n$ where $B_j \subseteq \{s_1, \ldots, s_n\} - \{s_j\}$ for each j, every sensor in B_j is within distance d of s_j, every B_j contains at least r elements, and every sensor belongs to at most b backup sets. (If this is not possible, output the special value NIL.)

Give an efficient algorithm to solve this problem, based on network flow techniques. Write a detailed justification that your algorithm is correct (in particular, explain how backup sets and flows correspond to each other) and analyze the worst-case running time of your algorithm. (HINT: Consider using *two* nodes for each sensor.)

Solution on the next page... but you will get the most benefit from this example if you try it for yourself first!

Algorithm:

1. Create a network N with vertices $V = \{s, a_1, \ldots, a_n, b_1, \ldots, b_n, t\}$ and edges
   - $E = \{(s, a_1), \ldots, (s, a_n)\}$ (with $c(s, a_i) = b$)
   - $\cup \{(b_1, t), \ldots, (b_n, t)\}$ (with $c(b_j, t) = r$)
   - $\cup \{(a_i, b_j) : d(s_i, s_j) \leqslant d\}$ (with $c(a_i, b_j) = 1$).

   (Vertices $a_1, \ldots, a_n$ represent sensors; vertices $b_1, \ldots, b_n$ represent backup sets.)
2. Find a maximum integer flow f in network N (using the Edmonds-Karp algorithm, for example).
3. If $|f| < rn$, then return NIL; else, for $j = 1, \ldots, n$, set $B_j = \{s_i : f(a_i, b_j) = 1\}$, and return $B_1, \ldots, B_n$.

Correctness: Every collection of backup sets $B_1, \ldots, B_n$ with no sensor belonging to more than b sets and no backup set of size more than r (but not all B_j necessarily having size r) gives rise to a flow f in N as follows:

- $f(s, a_i)$ = number of backup sets that sensor s_i belongs to (not more than b so edge capacity respected);
- $f(a_i, b_j) = 1$ iff $s_i \in B_j$;
- $f(b_j, t)$ = number of sensors in backup set B_j (not more than r so edge capacity respected).

In this way, flow is conserved at each a_i because the total flow out is exactly equal to the number of backup sets that s_i belongs to, and flow is conserved at each b_j because the total flow in is exactly equal to the number of sensors in backup set B_j. So the maximum flow value $|f|$ is at least as large as the total size of all the backup sets.

Every integer flow in N gives rise to a collection of backup sets $B_1, \ldots, B_n$ with no sensor belonging to more than b sets, as follows:

- $B_j = \{s_i : f(a_i, b_j) = 1\}$;
- no sensor belongs to more than b backup sets because $c(s, a_i) = b$;

For these backup sets, $|B_1| + \cdots + |B_n| = |f|$ because both sides are equal to the number of edges (a_i, b_j) with $f(a_i, b_j) = 1$. This means that the total size of all the backup sets is at least as large as the maximum flow value in N.

Hence, finding a maximum flow in N yields a collection of backup sets with the maximum total size. If this size is equal to rn, then the backup sets $B_1, \ldots, B_n$ defined above satisfy the conditions of the problem; else, there is no collection of backup sets that is large enough.

Runtime: Creating N takes time $\Theta(n^2)$ (every pair of sensors (s_i, s_j) must be examined); solving the maximum flow problem takes time $\mathcal{O}(n^5)$ (using the Edmonds-Karp algorithm, for example); constructing the backup sets takes time $\Theta(n^2)$ (every edge (a_i, b_j) must be examined). The total is $\Theta(n^5)$—dominated by the time to solve the maximum flow problem.