Cookie assignment Consider the following problem:
You are baby-sitting \( n \) children and have \( m > n \) cookies to divide between them. You must give each child exactly one cookie (of course, you cannot give the same cookie to two different children). Each child has a greed factor \( g_i, 1 \leq i \leq n \) which is the minimum size of a cookie that the child will be content with; and each cookie has a size \( s_j, 1 \leq j \leq m \). Your goal is to maximize the number of content children, i.e., children \( i \) assigned a cookie \( j \) with \( g_i \leq s_j \).

Give a correct greedy algorithm for this problem (10 points), prove that it finds the optimal solution (20 points), and give an efficient implementation (10 points).

Here is the greedy strategy:

Candidate Strategy two: Look at the greediest child. If the largest cookie makes the child content, give the child the largest cookie. Otherwise, give the child the smallest cookie.

For the optimal strategy, Candidate Strategy 2, we will state and prove an exchange argument first for the case when the greediest child is content with the largest cookie. Then we do the same with the case when the greediest child is not content with the largest cookie.

Lemma: Assume \( g_1 \) is the largest greed factor of any child, and that \( s_1 \), the largest size cookie, is at least \( g_1 \). Then there is an optimal assignment that assigns child 1 cookie 1.

Let \( \text{Assign}' \) be any optimal assignment. If \( \text{Assign}' \) assigns child 1 cookie 1, then we are done. If \( \text{Assign}' \) assigns child 1 cookie \( j \neq 1 \), then it either does not give any child cookie 1 or gives cookie 1 to some child \( i > 1 \). In the first case, we let \( \text{Assign} \) give cookie 1 to child 1 and leave the other cookies as they are in \( \text{Assign}' \). Since child 1 is content with cookie 1, and the other children have the same cookies they had before, there are at least as many content children and so \( \text{Assign} \) is also optimal.

In the second case, define \( \text{Assign} \) to give cookie 1 to child 1, cookie \( j \) to child \( i \), and the other children get the same cookie as \( \text{Assign}' \) gives them. \( g_1 \geq g_i \) since child 1 is the greediest child. Thus, if child 1 is content with cookie \( j \), then so is child \( i \), and by assumption child 1 is content with cookie 1. Thus if both children were content in \( \text{Assign}' \), both are content in \( \text{Assign} \), and \( \text{Assign} \) makes the same number of children content as \( \text{Assign}' \). If not, at least child 1 is content in \( \text{Assign} \) so \( \text{Assign} \) makes at least as many children content as \( \text{Assign}' \). Thus, in all cases \( \text{Assign} \) is
an assignment that makes at least as many kids happy as Assign’ and is hence also optimal, and gives 1 to 1.

Lemma: Let $g_1, \ldots, g_n$ be the greed factors of $n$ children, ordered from greediest to least greedy, and let $s_1, \ldots, s_m$ be the sizes of the $m$ cookies, from largest to smallest. Assume $g_1 > s_1$. Let $A'$ be an assignment of cookies to children so that child 1 is not assigned cookie $m$. Then there is an assignment $A$ of cookies to children that assigns cookie $m$ to child 1, so that the number of children $i$ assigned a cookie $j$ with $g_i \leq s_j$ by $A$ is at least the number of such children in $A'$.

Proof: Let $A'$ assign child 1 cookie $j < m$. In the first case, assume that the smallest cookie $m$ is not assigned to any child by $A'$. Let $A$ assign all children except child 1 the same cookie as $A'$ does, but assign child 1 cookie $m$. Then except for child 1, the same set of children are content in $A$ as $A'$. Since $g_1 > s_1 \geq s_j \geq s_1$, child 1 is not content in either $A$ or $A'$. Thus, $A$ and $A'$ have the same number of content children.

In the other case, cookie $m$ is assigned to a child $i > 1$ by $A'$. Then let $A$ assign all children except 1 and $i$ the same cookies as $A'$ does, and assign child 1 cookie $m$ and assign child $i$ cookie $j$. Again, all children except 1 and $i$ are equally content in $A$ and $A'$, and child 1 is not content in either assignment. If child $i$ is content in $A'$, $g_i \leq s_m \leq s_j$, so child $i$ is content in $A$. Thus, $A$ has at least as many content children as $A'$.

Then in any case, there is an optimal solution that gives the greediest child the same cookie $CG$ as the greedy algorithm does. We prove that greedy is optimal by induction on the number of children. If there are no children, any assignment is optimal. Assume the greedy solution is optimal for any set of $n - 1$ children and any set of cookies. Let $OPT$ be an optimal assignment of cookies to children that gives the greediest child $CG$. $OPT$ assigns the remaining children cookies in Cookies − $CG$, and by the induction assumption, greedy is an optimal solution for the $n - 1$ remaining children and $m - 1$ remaining cookies. Therefore, the number of remaining children made content by $OPT$ is at most that for Greedy. But since they both give the greediest child $CG$, the greediest child is either content in both assignments or unhappy in both assignments. Either way, the total number of content children for $OPT$ is at most that in the greedy assignment.

Therefore, by induction on the number of children, greedy is optimal for any set of children and any set of cookies.

To implement the algorithm quickly, let’s sort the children by greed, and the cookies by size, with largest first. Note that we always assign children either the largest or smallest cookies. Thus, the remaining cookies are $C_I, \ldots, C_J$ for some $1 \leq I \leq J \leq m$. Let’s keep counters for the $I$ and $J$ to keep track of this interval.
The algorithm becomes:

CookieGive ($Children[1..n], Cookies[1..m]$).

1. Sort $Children$ by greed, $Cookies$ by size, largest to smallest.
2. $I \leftarrow 1, J \leftarrow m$.
3. FOR $K = 1$ to $N$ do:
   4. IF $s_I \geq g_K$ THEN $Assign[K] = I, I++$
6. Return $Assign$.

The time to sort are $O(n \log n + m \log m)$, and the inner loop is $O(n)$, giving a total time of $O(m \log m)$ since $m \geq n$.

**Implementation** Often, even when greedy algorithms do not find optimal solutions, they are used as heuristics. An independent set in an undirected graph $G$ is a set of nodes $I$ so that no edge has both endpoints in $I$. In other words, if $\{u, v\} \in E$, then either $u \not\in I$ or $v \not\in I$. The maximum independent set problem is, given $G$, find an independent set of the largest possible size.

Implement a greedy algorithm for maximum independent set based on including nodes of smallest degree. Test it on random graphs where each possible edge is in the graph with probability $1/2$. What is the average size of the independent set it finds for graphs of different sizes? (Try $n$ as many powers of 2 as you can.) How do you conjecture the size will grow as a function of $n$?

For larger $n$ even the lowest degree node will be connected to about $1/2$ the other nodes. That means that each time you remove a node, the graph will shrink by about a factor of 2. So you should see that the size of the independent set you find grows around $\log n$. The biggest independent set in a random graph however is about twice this size.