Input: Collection C of sets $S_1, ..., S_m$, with weight w_i for each set S_i.
Output: Subcollection $C' \subseteq C$ such that $\cup_{S \in C'} S = \sum_{S \in C} S$, and $\sum_{S_i \in C'} w_i$ is as small as possible.

It is useful to think of the weight of a set as its cost; then the problem requires to select (buy) sets whose total cost is as small as possible but such that all elements are obtained (covered). Thus the natural greedy idea is: Pick the set that gives the maximum number of elements for the money that you spend to buy it, i.e. its cost.

GreedySetcover$(C, U)\{$

1. Mark all elements of U "uncovered".
2. C' = null.
3. While some elements remain uncovered $\{$

   For each $S_j \notin C'$ compute the per element covering cost $u_j = w_j/$number of uncovered elements in S_j.

   Pick the set S_k that has minimum u_k. $C' = C' \cup \{S_k\}$.

   Mark the uncovered elements in S_k covered.

   $\}$
4. Return C'

$\}$

Example instance: $U = \{0, \ldots, 2^n - 1\}$. The sets are $S_i = \{2^i, \ldots, 2^{i+1} - 1\}$, for $i = 0, \ldots, n-1$. We also have set S_n consisting of all the even numbers, and S_{n+1} consisting of all the odd numbers from the universe. The weights $w_i = 1$ for all $i = 0, \ldots, n-1$, and $w_n = w_{n+1} = 1 + \epsilon$.
Execution: $u_i = w_i/2^i = 1/2^i$ for $i = 0, \ldots, n-1$. $u_n = u_{n+1} = (1+\epsilon)/2^{n-1}$. Thus S_{n-1} will be selected. In each subsequent iteration $S_{n-2}, \ldots, S_0$ will get selected, and finally S_n. So the total cost will be $n + 1 + \epsilon$.

Note that the optimal cost is $2 + 2\epsilon$, obtained by selecting S_n, S_{n+1}. So it is not a great algorithm, since the ratio of the cost of the greedy algorithm to the cost of the Optimal algorithm is about $(n+1)/2 = (1 + \log_2 N)/2$ where N denotes the number of elements to be covered.

1 Analysis

Let OPT denote the cost paid by the optimal algorithm, which picks a collection C^*. We can say that the optimal algorithm pays OPT to "buy" the $|U|$ elements, where $U = \cup_{S \in C} S$. In each iteration the greedy algorithm buys some elements by paying an amount w_k, which minimizes the per unit cost *at that iteration*. We can lower bound the number of elements bought by greedy by comparing the unit cost paid by greedy to the per unit cost $OPT/|U|$ paid by the optimal algorithm.

For convenience, let us renumber the sets in C so that the set picked by the greedy algorithm in iteration i is numbered i; those never picked are numbered arbitrarily. Let U_i denote the set of uncovered elements at the end of iteration i of the greedy algorithm. Let $U_0 = U$.

Lemma 1 $|U_i| \leq |U_{i-1}|(1 - w_i/OPT)$

Proof: Let r_j denote the number of elements in each S_j that remain uncovered just before iteration i. Let C^* denote the optimal collection. We know that

$$\sum_{S_j \in C^*, j \geq i} w_j \leq OPT \quad \text{and} \quad \sum_{S_j \in C^*, j \geq i} r_j \geq |U_{i-1}|$$

Greedy picks the set with the minimum unit cost, which we have numbered S_i, i.e. $w_i/r_i \leq w_j/r_j$ for all $j \geq i$. But then for the collection C^* we have

$$\frac{w_k}{r_k} \leq \frac{\sum_{S_j \in C^*, j \geq i} w_j}{\sum_{S_j \in C^*, j \geq i} r_j} \leq \frac{OPT}{|U_{i-1}|}$$

Thus $r_k \geq w_k |U_{i-1}|/OPT$. But now $|U_i| = |U_{i-1}| - r_k \leq |U_{i-1}|(1 - w_k/OPT)$. ■

Lemma 2 *The greedy solution has weight at most* $OPT \cdot (1 + \ln |U|)$.

Proof: Note that $1 - x \leq e^{-x}$ with equality only when $x = 0$. Thus from the above Lemma we have $|U_i| < |U_{i-1}| e^{-w_i/OPT}$. But we write $|U_{i-1}|$ similarly. So proceeding in this manner we have $|U_i| < |U| e^{-W_i/OPT}$ where $W_i = \sum_{j=1}^{i} w_j$.

The algorithm terminates when $|U_i| = 0$, which will surely happen as soon as $|U| e^{-W_i/OPT} < 1$, i.e. as soon as $W_i > OPT \cdot \ln |U|$. Since in the ith step only OPT weight could have been added, we know that at termination $W_i < OPT \cdot (1 + \ln |U|)$. But W_i is the weight the greedy algorithm will need. ■