

Input: Collection C of sets S_1, \dots, 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. $C' = \text{null}$.

3. While some elements remain uncovered {

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

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: $U = \{0, \dots, 2^n - 1\}$. The sets are $S_i = \{2^i, \dots, 2^{i+1} - 1\}$, for $i = 0, \dots, n - 1$. 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 $i = 0, \dots, n - 1$, and $w_n = w_{n+1} = 1 + \epsilon$.

Execution: $u_i = w_i/2^i = 1/2^i$ for $i = 0, \dots, n - 1$. $u_n = u_{n+1} = (1 + \epsilon)/2^{n-1}$. Thus S_{n-1} will be selected. In each subsequent iteration S_{n-2}, \dots, 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

The crux of the analysis is to estimate the unit price paid in each iteration. While this can be estimated not absolutely, it can be estimated in terms of the total cost OPT paid by the optimal algorithm to pick the optimal subcollection C^* .

Let U_i denote the set of uncovered elements at the end of iteration i of the greedy algorithm. Let $U = U_0 = \cup_{S \in C} S$.

Lemma 1 $|U_i| \leq |U_{i-1}|(1 - w_k/OPT)$, where S_k is picked in iteration i .

Proof: We know that U is covered by the sets in C^* at total cost OPT . Some of the sets in C^* might get picked in the first $i - 1$ iterations. Let D^* denote the sets from C^* that do not get picked. Then we know that the sets in D^* can cover the elements U_{i-1} which remain uncovered after iteration $i - 1$. Let $D^* = \{S_1^*, \dots, S_p^*\}$. Letting $w(S)$ denote the weight of S we have

$$\sum_j w(S_j^*) \leq OPT$$

because $S_j^* \in C^*$. Let r_j^* denote the number of elements of S_j^* not covered by iteration i . Because the sets S_j^* together can cover U_{i-1} we have

$$\sum_j r_j^* \geq |U_{i-1}|$$

Thus it follows that there must exist some S_j^* for which $w(S_j^*)/r_j^* \leq OPT/|U_{i-1}|$. But the left hand side is simply the unit price of the set S_j^* . Since the algorithm picks the set with the minimum unit price, the set S_k it picked must have unit price possibly even smaller. Thus $u_k = w_k/r_k \leq OPT/|U_{i-1}|$. But then $r_k \geq w_k|U_{i-1}|/OPT$.

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 $OPT \cdot (1 + \ln |U|)$.*

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 not picked are numbered arbitrarily. Also, let $N_i = |U_i|$. Then the above Lemma says $N_i \leq N_{i-1}(1 - w_i/OPT)$.

Next note that $1 - x \leq e^{-x}$ with equality only when $x = 0$. Thus we have $N_i < N_{i-1}e^{-w_i/OPT}$. But we write N_{i-1} similarly. So proceeding in this manner we have $N_i < N_0e^{-W_i/OPT}$ where $W_i = \sum_{j=1}^i w_j$.

The algorithm terminates when $N_i = 0$, which will surely happen as soon as $N_0e^{-W_i/OPT} < 1$, i.e. as soon as $W_i > OPT \cdot \ln N_0$. Since in the i th step only OPT weight could have been added, we know that at termination $W_i < OPT \cdot (1 + \ln N_0)$. But W_i is the weight the greedy algorithm will need. ■