CS 170 DIS 04

Released on 2018-02-14

1 Worst-case Instances for Greedy Set-Cover

Recall the set cover problem:

Input: A set of elements B and sets $S_1, \dots, S_m \subseteq B$.

Output: A selection of the S_i whose union is B (i.e. that contain every element of B).

Cost: Number of sets picked.

The natural strategy to solve this problem is a greedy approach: At every step, pick the set that covers the most uncovered elements of B. In the book, we proved that this greedy strategy over-estimates the optimal number of sets by a factor of at most $O(\log n)$, where $n = |B|$. In this problem we will prove that this bound is tight.

Show that for any integer n that is a power of 2, there is an instance of the set cover problem (i.e. a collection of sets $S_1, \dots, S_m$) with the following properties:

i. There are n elements in the base set B.

ii. The optimal cover uses just two sets.

iii. The greedy algorithm picks at least $O(\log n)$ sets.

Solution: Consider the base set $U = \{1, 2, 3, \dots, 2^k\}$ for some $k \geq 2$. Let $T_1 = \{1, 3, 5, \dots, 2^k - 1\}$ and $T_2 = \{2, 4, 6, \dots, 2^k\}$. These two sets comprise an optimal cover. We add sets $S_1, \dots, S_{k-1}$ by defining $l_i = 2 + \sum_{j=1}^{i} 2^{k-j}$ (take $l_0 = 0$) and letting $S_i = \{l_{i-1} + 1, \dots, l_i\}$. We think of S_i as covering a tiny bit more than a $1/2^i$-fraction of the universe, and in particular S_1 is larger than both T_1 and T_2.

Since S_1 contains $2^{k-1} + 2$ elements, it will be picked first. After the algorithm has picked $\{S_1, S_2, \dots, S_i\}$, each of T_1 and T_2 covers $2^{k-i-1} - 1$ new elements while S_{i+1} covers 2^{k-i-1} new elements. Hence, the algorithm picks the cover $S_1, \dots, S_{k-1}$ containing $k - 1 = \log n - 1$ sets.

An example of the above algorithm is with, say, $n = 64$. Let $T_1 = \{1, 3 \dots, 63\}$, and $T_2 = \{2, 4, \dots, 64\}$. Let picking T_1 and T_2 be the optimal answer. Now let $S_1 = \{1, 2, \dots, 32, 33, 34\}$, or 2 elements more than $64/2$. Let S_2 contain the next $64/4 = 16$ elements, let S_3 contain the next $64/8 = 8$ elements, S_4 contain the next $64/16 = 4$ elements, and S_5 contain the remaining 2 elements. This works because $64 = 32 + 16 + 8 + 4 + 2 + 2$. We are just regrouping the terms as follows $64 = (32 + 2) + 16 + 8 + 4 + 2$ and creating sets out of them.

Our greedy algorithm will pick S_1, S_2, S_3, S_4, S_5 instead of T_1, T_2. Our greedy algorithm ended up picking $\log(k) - 1$ sets, since $k = 6$ for $n = 64$. The first two paragraphs of the solutions explain how this is generalized for all k.

2 Money Changing.

Fix a set of positive integers called *denominations* $x_1, x_2, \ldots, x_n$ (think of them as the integers 1, 5, 10, and 25). The problem you want to solve for these denominations is the following: Given an integer A, express it as

$$A = \sum_{i=1}^{n} a_i x_i$$

for some nonnegative integers $a_1, \ldots, a_n \geq 0$.

1. Under which conditions on the denominations x_i are you able to do this for all integers $A > 0$?

2. Given any set of denominations x_i, not necessarily satisfying the conditions in the first part, describe the set of *sufficiently large* integers A that you can express as $A = \sum_{i=1}^{n} a_i x_i$ with nonnegative a_i. In other words you should prove a statement like "If A exceeds X, then we can write $A = \sum_{i=1}^{n} a_i x_i$ for $a_i \geq 0$ if and only if A has the following simple property...." You do not have to give X explicitly, but prove it exists.

3. Suppose that you want, given A, to find the nonnegative a_i's that satisfy $A = \sum_{i=1}^{n} a_i x_i$, and such that the sum of all a_i's is minimal —that is, you use the smallest possible number of coins. Define a *greedy algorithm* for this problem.

4. Show that the greedy algorithm finds the optimum a_i's in the case of the denominations 1, 5, 10, and 25, and for any amount A.

5. Give an example of a denomination where the greedy algorithm fails to find the optimum a_i's for some A. Do you know of an actual country where such a set of denominations exists?

6. How far from the optimum number of coins can the output of the greedy algorithm be, as a function of the denominations?

Solution:

1. A can be expressed as a linear combination of the x_i if and only if $x_i = 1$ for some i. If one of your denominations x_i is 1, you will certainly be able to express every integer A as $\sum_{i=1}^{n} a_i x_i$ for some nonnegative integers $a_1, \cdots, a_n$. Conversely, in order to express $A = 1$ as a linear combination, you must have $x_i = 1$ for some i.

2. A necessary condition that $A = \sum_{i=1}^{n} a_i x_i$ is that $g = gcd(x_1, ..., x_n)$ divides A. This condition is also sufficient if we allow negative a_i (eg $1 = 1 \cdot 6 + (-1) \cdot 5$). But if we restrict $a_i \geq 0$, then $g|A$ turns out to be both necessary and sufficient for $A \geq X$ for some (large) X. In other words, if g divides A **and** $A \geq X$, then we can create change A using our coins x_i.

   Here is a proof.

   From the extended Euclidean algorithm we know we can write $g = \sum_{i=1}^{n} g_i x_i$ with some possibly negative g_i. We realize that if all the g_i were positive, we would be done.

So, we let $G = \sum_{i=1}^{n} |g_i| x_i$. Additionally, let $x_{min} = \min_i x_i$ and $k = x_{min}/g$, and $X = kG$.

First, note that g divides X. Why? Because remember that g is the $gcd(x_i)$, and G is a weighted sum of the x_is. g divides each x_i, and so it also divides G. Since g divides G, and $X = kG$, g divides X.

So, the claim is that as long as $A \geq X$ and $g|A$, we can represent $A = \sum_i a_i x_i$ for positive a_i.

Let's consider X. $g|X$, and $X \geq X$. So, can X be written as $\sum_i a_i x_i$? Yes, because $X = kG$, k is positive, and $G = \sum_i a_i x_i$ for positive a_i.

Next, let's consider $X + g$. $g|(X + g)$ and $(X + g) \geq X$. So, can $X + g$ be written as $\sum_i a_i x_i$? Yes, because $X + g = kG + g$. This is equal to $k(\sum_{i=1}^{n} |g_i| x_i) + \sum_{i=1}^{n} g_i x_i = \sum_{i=1}^{n} (k|g_i| + g_i) x_i$. If g_i is positive, $k|g_i| + g_i$ is also positive. If g_i is negative, $k|g_i| + g_i = (k - 1)|g_i|$, which is still positive.

Similar arguments can be made for $X + 2g, X + 3g, \ldots, X + (k - 1)g$.

Now, let's consider $X + kg$. Note that $X + kg = X + x_{min}$, because $k = x_{min}/g$. We know that X can be created with positive coins, and x_{min} is simply 1 coin of the smallest denomination, therefore $X + x_{min}$ can be created with positive coins.

Next, let's consider $X + (k+1)g$. Note that $X + (k+1)g = X + kg + g = X + g + x_{min}$. We know that $X + g$ can be created, and so then can $X + (k + 1)g$.

This argument can therefore be extended for any $X + mg$, where m is a natural number.

Note that the coefficients are not necessarily unique (all the x_i could be identical), but we have shown that there is at least one set of nonnegative coefficients for all multiple of g at least equal to X.

3. Order your denominations such that $x_1 > x_2 > \cdots > x_n$. Then the *greedy algorithm* for this problem would be: Given A, let a_1 be the largest integer such that $a_1 x_1 \leq A$. If $A - a_1 x_1 > 0$, let a_2 be the largest integer such that $a_2 x_2 \leq A - a_1 x_1$. If you have nothing left over after doing this for $i = 1, \cdots, n$, then $A = \sum_{i=1}^{n} a_i x_i$.

4. Since 1 divides 5 and 5 divides 10, it is clear that if we have a case in which the greedy algorithm would not find the optimal solution, it must involve 25, *i.e.* A must be greater than 25.

   Note that $x_4 = 1$ cent, $x_3 = 5$ cent, and so on.

   Assume the greedy algorithm does not find the optimal solution for A, $A > 25$.

   Then $A = \sum_{i=1}^{4} a_i x_i = \sum_{i=1}^{4} b_i x_i$ and $\sum_{i=1}^{4} a_i > \sum_{i=1}^{4} b_i$, where the a_i were determined by the greedy algorithm and the b_i are optimal in that $\sum_{i=1}^{4} b_i$ is minimal.

   W.l.o.g. $a_4 = b_4$ [since $a_4 \leq 4$ any change of the number of 1 cent coins must occur in 5 unit steps to give the same sum-this is obviously worse than changing b_3]. Also, since the other denominations are $5, 10, 25$, the number of 1 cent coins that the optimal algorithm takes must be $A \bmod 5$, which is the number of 1 cent coins our greedy algorithm takes too. In addition to that note that $a_3 \leq 1$.

By the above considerations we must have $a_1 > b_1$. Why? Because our greedy algorithm can certainly not pick *less* 25-cent coins than the optimal algorithm. The first thing our greedy algorithm does is pick as many 25-cent coins as possible! Also, a_1 is not *equal* to b_1, because if it were, then we know that our greedy algorithm correctly picks the optimal set of coins until $A = 24$ anyway (since 1 divides 5 and 5 divides 10.)

So, let $x := a_1 - b_1$. Note that x is a positive number.

For a_2, b_2 have three cases to consider: $a_2 = b_2$, $a_2 > b_2$ and $a_2 < b_2$.

Let's set $y := a_2 - b_2$.

Now, remember that $a_1x_1 + a_2x_2 + a_3x_3 + a_4x_3 = b_1x_1 + b_2x_2 + b_3x_3 + b_4x_4$. We can rewrite this as $b_3 = 5x + 2y + a_3$, using the actual values of x_i, the fact that $a_4 = b_4$, and our definitions of x and y.

Thus the number of coins changes by $\sum_{i=1}^4 b_i - \sum_{i=1}^4 a_i = 4x + y$. If we can show that this number is positive, this is a contradiction and we are done, since we expected $\sum_{i=1}^4 a_i > \sum_{i=1}^4 b_i$.

In cases 1 and 2, x and y are ≥ 0. Therefore $4x + y$ is clearly positive.

In case 3, y is negative. But, as we have to ensure that $b_3 = 5x + 2y + a_3$ is ≥ 0 and we know that a_3 is at most 1, we have $y \geq -\frac{5}{2}x - \frac{1}{2}$. Hence $4x + y \geq \frac{3}{2}x - \frac{1}{2}$ and it is again positive.

5. A couple of real world examples:

- The United States of America 1875 – 1878 had 25 cent, 20 cent, 10 cent and 5 cent coins (and no 40 cent coins). To get 40 cents, the *greedy* algorithm gives 25 — 10 — 5, *i.e.* three coins, whereas the minimum is two coins (20 — 20)
- Prior to the change to the decimal system, Britain and many of her colonies had the following system:

$$\begin{array}{lcccl} & & 1 \text{ shilling} & = & 12 \text{ pence} \\ 1 \text{ florin} & = & 2 \text{ shillings} & = & 24 \text{ pence} \\ 1 \text{ half–crown} & = & 2.5 \text{ shillings} & = & 30 \text{ pence} \end{array}$$

  So to get 36 pence, the *greedy* algorithm would take a half–crown and six pennies (*i.e.* seven coins), whereas one florin and one shilling (two coins) would be the minimal solution
- Cyprus in 1901 had 18 Piastres, 9 Piastres, 4.5 Piastres and 3 Piastres Silver coins and 1 Piastre, 0.5 Piastre and 0.25 Piastre Bronze coins.
  To get 6 Piastres, the *greedy* algorithm would take 4.5, 1 and 0.5 Piastre coins (three coins), whereas the minimum would be two 3 Piastre coins
- Persia under Muzaffar–ed–din Shah (1896 – 1907) had the following coins: 2 Tomans (= 400 Shahi), 1 Toman (= 200 Shahi), 0.5 Toman (= 100 Shahi), 4 Kran (= 80 Shahi), 0.25 Toman (= 50 Shahi), 2 Kran (= 40 Shahi), 1 Kran (= 20 Shahi), 0.5 Kran (= 10 Shahi), 0.25 Kran (= 5 Shahi), 3 Shahi, 2 Shahi and 1 Shahi.
  To get the sum of 160 Shahi, the *greedy* algorithm would take a 100 Shahi, a 50 Shahi and a 10 Shahi coin (three coins), whereas the minimum would be two 80 Shahi coins

6. As the counterexamples show, the fact that the greedy algorithm goes wrong is connected with the difference between two successive denominations. To represent the furthest possible distance Δ_{max} of the output of the greedy algorithm $G(A)$ from the optimum number of coins $M(A)$, let's consider one more pathological example that would guide us to our conclusion.

   Example 1. Denominations $D_1 = \{1, 7, 8\}$; $A = 2 \cdot x_2 = 2 \cdot 7 = 14$; and $G(14) = 1_8 + 6_1 = 7$ coins. The optimum solution $M(14) = 2_7 = 2$ coins, thus the difference of outputs is $\Delta(14) = 5$. Now supposing we vary $x_3 = 8$ from 8 to 13, then the corresponding $\Delta(14)$ takes the values $5, 4, 3, 2, 1, 0$ respectively. The last 0 indicates that the greedy algorithm yields the optimum solution in this last case.

   Now keeping the original denominations $D_1 = \{1, 7, 8\}$ but varying A from 13 to 20 maintains the optimality of the greedy solution, until for $A = 3 \cdot x_2 = 3 \cdot 7 = 21$, $G(21) = 2_8 + 5_1 = 7$ coins, whereas the optimum solution $M(21) = 3_7 = 3$ coins. Thus the difference of outputs, is $\Delta(21) = 7 - 3 = 4$. Varying the last denomination x_3 from 8 to 21, the corresponding $\Delta(21)$ varies from 4 to 0.

   A few observations so far:

   - The locally maximum $\Delta(A)$ occur at A such that A is a multiple of x_2, and $2x_2 \leq A \leq x_2^2$.
   - Δ is maximized when $x_n = x_{n-1} + 1$, $x_{n-1} > 2$.
   - Denominations x_i such that $x_i > A$ do not contribute to the growth of Δ as they cannot be used in the *change* for A. Thus w.l.o.g. we may assume that Δ is maximized when x_{n-1} and x_n are the ones with the pathological relationship.
   - Had there been any denomination between 1 and x_{n-1}, it would only have decreased the value of $G(A)$, by *catching* some of the remainder from A/x_n, thus decreasing the value of Δ. Therefore for the worst possible case, i.e. the maximum Δ, we may assume only three denominations: 1, x_2, $x_2 + 1$ with $x_2 > 2$.
   - At the local maximas, i.e. the multiples of x_2 from $2x_2$ to x_2^2, $\Delta(A) = x_2 - A/x_2$.

   From the last observation it follows that the worst performance of the greedy algorithm happens for $A = 2x_2$, $x_2 > 2$, where $\Delta(A) = x_2 - 2$.

3 Midterm Discussion