CMSC 641 Design & Analysis of Algorithms | Spring 2023
Section 01 | Homework 12

Faisal Rasheed Khan

VB02734

vb02734@umbc.edu

1 Set Cover Example

Consider the following Set Cover example from the textbook. The underlying set X is a set of 12 characters:

X = {a, d, e, h, i, l, n, o, r, s, t, u}

The family F of subsets of X are depicted as words, but they are really just subsets of characters.
The words are:
F = { arid, dash, drain, heard, lost, nose, shun, slate, snare, thread }
For example, shun is really the subset {h, n, s, u}.

1. Which words would be chosen by the Greedy Set Cover algorithm for this instance of Set Cover? Break ties by choosing the word that is earlier in dictionary order.

A. The Greedy Set Cover chooses the words which covers the largest number of letters in X.
The Greedy Algorithm works as select the set which covers largest letters.
Repeat the algorithm till every letter is covered i.e. by selecting subset everytime which has largest letters covered.
The words chosen by Greedy Alogrithm is {thread, lost, shun, drain}

2. Is the solution provided by Greedy Set Cover an optimum solution?

A. The solution provided by Greedy may not be the optimum solution as it is np problem we just verify the answer, there could be more number of solutions for the same set.
To explain further consider Greedy solutions according to the algorithm below:
The algorithm could select={thread, nose, shun, arid, lost}
Or {thread, shun, lost, arid}
So if greedy select the second word nose then it might end up including the words more in subset than the optimum solution.
Therefore, Solution provided by Greedy Set Cover is not always optimum

2 Bin Packing: First-Fit Decreasing

Instructions: Below is a proof that the First-Fit Decreasing (FFD) heuristic uses no more than $3/2 \cdot c*$ bins, where $c*$ is the number bins used in the optimum packing. At various points in the proof, you will be asked to provide an explanation or analysis. The correct answers to these questions are short and direct. If you are writing a lot, you most likely missed the point of the question.

Recall that in the Bin Packing problem we have n items with sizes s1, s2, . . . , sn. We are asked to place the items into bins so that the sum of the sizes of the items in each bin is ≤ 1. The objective is to do so with as few bins as possible.
In the First-Fit Decreasing (FFD) heuristic, we sort the items by size. So, we can assume that
1 ≥ s1 ≥ s2 ≥ s3 ≥ · · · ≥ sn.
We take the items in order of decreasing size and place the current item in the first bin with enough remaining space to hold the current item. If none of the bins have enough space, then we start a new bin. The bins are considered in the order they were started — oldest bin first.
Let c∗ be the number of bins used in the optimum bin packing. Let i0 be the index of the first item that FFD places in bin number c∗ + 1.

Task 1. There is no guarantee that i0 exists. What happens if there is no such i0?
Claim 1: si0 ≤ 1/2.
Proof by contradiction: suppose si0 > 1/2. Then s1, s2, . . . , si0 are all > 1/2. So, there are c∗ + 1 items that are > 1/2 which means the optimum bin packing must use at least c∗ + 1 bins, a contradiction.
QED.

Task 2. How do we know there are c∗ + 1 items > 1/2?

A. From task 1 we have prove that $s_{i_0} \leq \frac{1}{2}$ which is in the c* + 1 th bin, since the elements are inserted in Decreasing order c* bin elements contain $\geq \frac{1}{2}$. If the c* each element = $\frac{1}{2}$ then every 2 elements requires a bin, when they are even then it has $\frac{1}{2}$ elements exactly, if the last i_0 is included the c* + 1 > $\frac{1}{2}$
If c* each element > $\frac{1}{2}$ then each element requires a bin which is c* the next element if its < $\frac{1}{2}$
Then it can be adjusted in the last bin c*, else it will be placed in the new bin c* + 1.
Therefore, c∗ + 1 items > 1/2

Task 3. Why would the optimum bin packing use at least c∗ + 1 bins?

A. From task 1 we have prove that $s_{i_0} \leq \frac{1}{2}$, 1 ≥ s1 ≥ s2 ≥ s3 ≥ · · · ≥ sn. since the elements are inserted in Decreasing order c* bin elements contain $\geq \frac{1}{2}$
If c* each element = $\frac{1}{2}$ then $\frac{1}{2}$ of the elements will fit in c* bin and that extra s_{i_0} irrespective of the value needs 1 bin i.e total bins = c* + 1, and c* + 1 are the atleast no of bins.

Task 4. Why would the optimum bin packing using c∗ + 1 bins be a contradiction?

A. The contradiction is when each element has size $\frac{1}{2}$ and elements are odd then it takes c* bins to fill up by FFD, the last bin has half the place still left, if the $s_{i_0} \leq \frac{1}{2}$ then it is placed in the last bin c*. total bins = c* (but not c* + 1). Therefore, optimum bin packing using c∗ + 1 bins is a contradiction

Let c be the total number of bins used by FFD. We call the bins numbered c∗ + 1 through c the "extra bins." Let t be the number of items that FFD placed in the extra bins and let z1, z2, . . . , zt be their sizes.

Task 5. There are n − i0 items that come after item i0, the first item to be placed in an extra bin. Why isn't t just equal to n − i0 + 1?
Claim 2: t ≤ c∗ − 1.
Proof by Contradiction: Suppose t ≥ c∗ . For 1 ≤ j ≤ c∗ , let bj be the sum of the sizes of the items that FFD placed in bin number j. Then, for 1 ≤ j ≤ c∗ , bj + zj > 1. Thus, $\sum_{j=1}^{c*}(bj\ +\ zj\ ) > \sum_{j=1}^{c*} 1$ = c∗
Since the sum of the sizes of all of the items cannot exceed c∗ , this is a contradiction. QED.

Task 6. Why is bj + zj > 1?

A. Since the bins from starting is filled with decreasing order of size i.e their values are almost nearer to 1 if we try adding z_j then it will exceed 1.

Task 7. Why can't the sum of all the sizes exceed c∗ ?

A. As the elements decrease further (since they are in descending order, then in worst case, the sum of all sizes are exactly c* (as size decreases since we are sorting in descending order). Each element is fit and given bins according to the limit of their sizes. Therefore, at most c* will be sum of sizes. Therefore, it cannot exceed c*

Finally, since each item that FFD placed in an extra bin has size ≤ 1/2 and since there are no more than c∗ − 1 of them, FFD uses no more than ⌈ (c∗ − 1)/2⌉ ≤ c∗/2 extra bins. Thus, FFD uses no more than 3/2 · c∗ bins in total.

Task 8. How do we know that FFD uses no more than ⌈ (c∗ − 1)/2⌉ extra bins?

A. Each item in extra bin has size ≤$\frac{1}{2}$. Then there are c* + 1 bins used for bin size ≥$\frac{1}{2}$. Now adding elements will have extra bins and thier size will be decreasing and will be no more than $\frac{1}{2}$ i.e ≤$\frac{1}{2}$. consider its $\frac{1}{2}$ then at most it takes $\frac{(c*\,-\,1)}{2}$ *bins since size of* $\frac{1}{2}$ *, 2 elements can be placed in one bin*. Therefore, it will not exceed ceil i.e. ⌈ (c∗ − 1)/2⌉ extra bins

Task 9. Why is the total number of bins used by FFD less than or equal to 3/2 · c∗?

A. From task 8 we get no.of extra bins for items is. ⌈ (c∗ − 1)/2⌉ used by FFD. Original bins without extra bins are c* + 1.
Therefore, total no.of bins = (c* + 1) + (c∗ − 1)/2
= $\frac{3}{2}$c* its atmost
So, total number of bins used by FFD less than or equal to $\frac{3}{2}$c*

References:

https://archive.org/details/introduction-to-algorithms-by-thomas-h.-cormen-charles-e.-leiserson-ronald.pdf/page/427/mode/2up

https://archive.org/details/AlgorithmDesign1stEditionByJonKleinbergAndEvaTardos2005PDF/page/n201/mode/2up

21.1-Approximation-Algorithms-post.pdf - Google Drive

21.2-Approximating-TSP-post.pdf - Google Drive

22-Bin-Packing-post.pdf - Google Drive

23.2-Subset-Sum-post.pdf - Google Drive

23.1-Approximating-Set-Cover-v2-post.pdf - Google Drive