HW4 Report

CSE 373

Minqi Shi

111548035

I used backtracking algorithm in this homework to exhaustively find all possible solutions and determine the best one. To get a working program is easy but an efficient one is hard. The hardest part I think is pruning, trying to avoid as much unnecessary backtracking as possible.

I first read in the data and store them in two list. (Just a remainder, to test different files, you need to manually change the filePath variable in Line 15. Sorry for the inconvenience. Thank you!) I have two wrapper classes that store the elements and the subsets. An Element class has a value representing the actual value and a list of Subsets which contain this element. A Subset class contains an ID and a list of integers that in the subset. I also have a MinSet class which is just MSC to represent the result. The MinSet class has a list of Subsets (as the result) and a Boolean array representing if a certain element is already covered in the subsets. While reading in the data, if I encounter a set that only has only one element and the element is already covered in the subsets added before, simply ignore this subset and continue. Additionally, I have a static variable finalResult to store the possible solution.

Before backtracking, I search if there are any element that only exist in one subset. If exist, that subset has to be added to the partial result in order to cover everything. In the meantime, I go through that subset and delete all the elements in that subset from the element list, because we

don’t have to consider finding the subset to include them in the result anymore. Then I sort the element list by the number of subsets a certain element in, in increasing order, since I want the to go through the elements that has less choice of Subsets than those having more. I also sort the subset list by the elements a certain subset has, in decreasing order.

If there’re more subsets than elements, I perform a backtracking search by the element list. First I check the partial solution is valid and better than the current solution I have; if so, replace the current solution and return; then if the partial solution has equal or more subsets than the current solution, there is no need to continue, so return. I get the first elements in the element list and make a for loop to go through all the subsets the element is in. To preserve the original element list and subset list, I make a deep copy of both and use the copies to in the for loop. In the for loop, I add the i^{th} subset to the partial result and remove all the elements that’s in the i^{th} subset from the element list and then call backtracking search again, passing the elements list and the partial result in.

If there’re more elements than subsets, and the elements are more than 200, I add in all the subsets, from the biggest to the smallest that don’t have any overlap with each other, with a tolerance of 1 elements (It means if the subset about to add has one element that is in the elements covered so far, we still consider there is no overlap). I then use the partial solution I got to get rid of the elements already covered. By doing that, I avoid lots of elements that need doing backtracking on. Then I perform a backtracking search by the reduced element list. After a final result is found, I check if there are any redundant subset in the result before finishing up.

The time for running the samples are either less than 5 seconds or more than a minute. For the k files, I have all the way to s-k-50-100. For rg files. I can run up to s-rg-245-50 and s-rg-245-50 seems to give me correct answer; but for s-rg-197-45 and s-rg-155-40 the program gives incorrect answers if I filled in with none-overlaps subsets first, while the running time exceed 1 minute if I directly apply a backtracking search by element list, although correct answer can be found that way. So, I didn't include 197-45 and 155-40 in the report.

s-k File Name	MSC		s-rg File name	MSC
20-30	6		8-10	4
20-35	6		31-15	9
30-50	9		40-20	10
30-55	9		63-25	16
35-65	10		109-35	22
40-60	14		118-30	20
40-80	9		245-50	35
50-95	12			
50-100	9			

And for s-X-12-6 the MSC is 3.

(Because most of the output for the above file come out almost instantly, I here omit the running time for each one. For s-rg-245-50, it is about 5 or 6 seconds on my computer)