Question 8.4 from CTCI: Power Set

Benjamin De Brasi

October 2, 2017

Question

Power Set: Write a method to return all subsets of a set.

Explanation and Algorithm

Here are two ways to solve this problem:

The first is using the Base Case and Build approach. Imagine that we are trying to find all subsets of a set like $S = \{a_1 > a_2 > ... > a_n\}$. We can start with the Base Case.

Base Case: n = 0. There is just one subset of the empty set: {}.

Case: n = 1. There are two subsets of the set $\{a_1\}$: {} , $\{a_l\}$.

Case: n = 2. There are four subsets of the set $\{a_1, a_2\}$: {} $\{a_1\}$ $\{a_2\}$ $\{a_1, a_2\}$

Case: n = 3. Now here's where things get interesting. We want to find a way of generating the solution for n on the prior solutions.

What is the difference between the solution for n 3 and the solution for n 2? Let's look at this more deeply:

P(2) = {} $\{a_1\}$ $\{a_2\}$ $\{a_1, a_2\}$

P(3) = {} $\{a_1\}$ $\{a_2\}$ $\{a_3\}$ $\{a_1, a_2\}$ $\{a_1, a_3\}$ $\{a_2, a_3\}$ $\{a_1, a_2, a_3\}$

Essentially, P(3) = P(2) + every a new copy of P(2) with each of its' subsets adding in a_3.

Here's some psuedocode for this idea:

foo(set,index);

set = [1, 2, 3], index = 0

masterlist = []

recurse incrementing till index = 3

return []

masterlist = [[]]

Go through each element in master list adding the element the index is on to all elements in the masterlist. To do this, without erasing the subsets you already have, you use a tmplist

tmplist = []

tmplist = [3]

masterlist.add(tmplist)

masterlist = [[], [3]]

This list then gets returned and the next recursion function on the stack executes, this time with

masterlist = [[], [3]] and index =2

tmp = [2]

tmp = [2,3]

each one adding to masterlist seperately making masterlist

masterlist = [[], [3], [2], [2,3]]

The same thing will occur for index 1 making

masterlist = [[], [3], [2], [2,3], [1], [1,3], [1,2], [1,2,3]]

Ths main logic is: use the same list which you will return the answer as a way of visiting all previously made subsets, creating a copy and adding the new current element you've reached and then adding that to the masterlist.

The second way involves a neat trick using binary numbers.

The first step is to understand how many subsets the power set of any given set will have. The easiest way to think about it is to note that within every subset of the power set an element in the origin set has two states it can be in; It can either be in that particular subset or not in that particular subset. If there are n elements that means there 2^n subsets total and thus 2^n elements in a power set of a set of size n.

In the same way that 1 or 0 means true or false for a boolean, here 1 and 0 mean include or exclude respectively. Getting a binary number that has as many digits as there are elements in the set means you've effectively mapped the inclusion and exclusion state of every element to the number. Just as critical, is the fact that incremental from 0 to 2^n means that you will get every combination of inclusions and exclusion states possible and thus the power set.

Hints

Recursive Approach:

1. How can you build all subsets of $\{a, b, c\}$ from the subsets of $\{a, b\}$?
2. Anything that is a subset of a, b is also a subset of $\{a, b, c\}$. Which sets are subsets of $\{a, b, c\}$ but not $\{a, b\}$?
3. Subsets that contain c will be subsets $\{a, b, c\}$ but not $\{a, b\}$. Can you build these subsets from the subsets of $\{a, b\}$?
4. You can build the remaining subsets by adding c to all the subsets of $\{a, b\}$.

Combinatorial Approach:

1. You can solve this problem by mapping each subset to a binary number. The ith bit could represent a "boolean" flag for whether an element is in the set.

Code

```
\*Answer 1 *\

ArrayList<ArrayList<Integer>> getSubsets(ArrayList<Integer> set, int
    index){
  Arraylist<ArrayList<Integer>> allsubsets;
```

```
   if(set.size() == index){
      allsubsets = new ArrayList<ArrayList<Integer>>();
      allsubsets.add(new ArrayList<ArrayList<Integer>>());
   } else {
      allsubsets = getSubsets(set, index + 1);
      int item = set.get(index);
      ArrayList<ArrayList<Integer>> moresubsets = new
          ArrayList<ArrayList<Integer>>();

      for(ArrayList<Integer> subset : allsubsets){
         ArrayList<Integer> newsubset = new ArrayList<Integer>();
         newsubset.addAll(subset);
         newsubset.add(item);
         moresubsets.add(newsubset);
      }
      allsubsets.addAll(moresubset);
   }
   return allsubsets;
}
```

```
\*Answer 2 *\

ArrayList<ArrayList<Integer>> getSubsets2(ArrayList<Integer> set){
   Arraylist<ArrayList<Integer>> allsubsets = new
       ArrayList<ArrayList<Integer>>();
      int max = 1 << set.size();
      for(int k = 0; k < max; k++){
         ArrayList<Integer> subset = convertIntToSet(k,set);
         allsubsets.add(subset);
      }
   return allsubsets;
}

ArrayList<ArrayList<Integer>> getSubsets2(int x, ArrayList<Integer> set){
   Arraylist<ArrayList<Integer>> subset = new
       ArrayList<ArrayList<Integer>>();
   int index = 0;
   for(int k = x; k > 0; k >>= 1){
      if((k&1) == 1){
         subset.add(set.get(index));
      }
      index++;
   }
   return subset;
}
```

Run Time Analysis

Time and space complexity is $O(n * 2^n)$

Sources

Question, answer and other material taken from Cracking the Coding Interview 4th edition by Gayle Laakmann McDowell.