

CS375 Small Assignment 4 Solutions

Due Sep 29th 10PM ET

Consider the following pseudo code for the generation of a list of all subsets of a given array (which is just a reproduction of what we wrote in class):

```
AllSubsets(A[0], ..., n-1):
  Initialize L as [∅] (a list just containing the empty set)
  for i in range(n):
    Initialize nextL as an empty list
    for each set S in L:
      Create a copy S' of S
      Add A[i] to S'
      Add both S and S' to nextL
  L = nextL
  return nextL
```

1. What is the worst case runtime of the above function? (I am looking for a Θ bound here, so this is really two questions in one)
2. What is the best case runtime of the above function? (I am looking for a Θ bound here, so this is really two questions in one)
3. Using the loop invariant method we've discussed in class, show that the above algorithm correctly outputs a list containing all the subsets of A.

It may help to know that that

$$\sum_{i=0}^n 2^i = 2^{n+1} - 1$$

for this homework.

Solutions: Like assignment 3, arguably the first two parts of this were poorly chosen, since the runtime of ALLSUBSETS is actually independent of its input, so its worst and best case runtime are the same. Note that on the i th iteration, as we shall show momentarily L will contain all subsets of

the items in $A[0], \dots, A[i-1]$. In particular, its runtime is:

$$\begin{aligned}
& \sum_{i=0}^{n-1} (\text{The runtime on the inside of the for loop}) \\
&= \sum_{i=0}^{n-1} \sum_{S \text{ in } L} (\text{the time to copy } S \text{ to } S' \text{ and add both to nextL}) \\
&= \sum_{i=0}^{n-1} \sum_{S \text{ in } L} \Theta(|S|) \\
&\leq \sum_{i=0}^{n-1} \sum_{S \text{ in } L} i = \sum_{i=0}^{n-1} i \sum_{S \text{ in } L} 1 \\
&= \sum_{i=0}^{n-1} i|L| \leq \sum_{i=0}^{n-1} i2^i \leq \sum_{i=0}^{n-1} n2^i \\
&= n \sum_{i=0}^{n-1} 2^i = n(2^n - 1) = \Theta(n2^n)
\end{aligned}$$

Note that due to the $\leq$ in there, this was an upperbound. To get a reasonable lowerbound, instead of bounding $|S| \leq i$ for each S in L , we'll just point out that $|S| \geq 1$ for all S in L other than the empty set. This gives us a lowerbound of $\Omega(2^n)$; you can do better, but it gets a bit ugly.

To demonstrate correctness, let's use loop invariants. The loop invariant we shall maintain is that at the start of each iteration of i , L contains all the subsets you can form of the first i elements in A : $A[0], \dots, A[i-1]$. At the start, when $i = 0$, this is true because L contains only the empty set, which is the only subset of nothing (note that if $i = 0$ then presumably we're taking all the subsets of nothing). Now, suppose that our loop invariant is true at the start of any particular iteration i . That is, L contains all the subsets of $A[0], \dots, A[i-1]$. At the end of iteration i , we want to make sure that L now contains all the subsets of $A[0], \dots, A[i-1], A[i]$. Pick any such subset T :

- If T **doesn't** contain $A[i]$, then T is actually just a subset of $A[0], \dots, A[i-1]$, so T was contained in L at the start of the i th iteration. Note that every iteration we add everything to nextL that was already in L (along with the copies with the extra element), so when $S = T$ we will add T to nextL, meaning T will still be in L on the next iteration as well.
- If T does contain $A[i]$, consider the set S that has everything in T **except** for $A[i]$. This is a subset of $A[0], \dots, A[i-1]$, so when we reach S in the inner for loop we will make a copy S' of it, add $A[i]$ to S' (at which point $S' = T$), and then add S' to nextL.

Note that in either case, the set T will be in L at the start of the next iteration, confirming our loop invariant holds for each iteration. In particular, since our loop invariant will hold at the moment $i = n$, we get that our loop finishes with L containing every subset of $A[0], \dots, A[n-1]$.