Homework 12, CSE 232

Due October 29 Note: On most of the problem sets through the semester, I'll put a horizontal line with "Optional" under it. Any problems below this section are encouraged - I think they're interesting and will help you learn the subject - but not necessary to complete in order to get credit for the homework.

Problem 1

This week's problem is Google Code Jam Round 1B 2012 - Equal Sums. Here's the problem statement:

> I have a set of positive integers S. Can you find two non-empty, distinct subsets with the same sum?
>
> Note: A subset is a set that contains only elements from S, and two subsets are distinct if they do not have exactly the same elements.

Input

> The first line of the input gives the number of test cases, T. T test cases follow, one per line. Each test case begins with N, the number of positive integers in S. It is followed by N distinct positive integers, all on the same line.

Output

> For each test case, first output one line containing "Case #x:", where x is the case number (starting from 1).
>
> If there are two different subsets of S that have the same sum, then output these subsets, one per line. Each line should contain the numbers in one subset, separated by spaces. If it is impossible, then you should output the string "Impossible" on a single line. If there are multiple ways of choosing two subsets with the same sum, any choice is acceptable.

Limits

> No two numbers in S will be equal.
>
> $1 \leq T \leq 10$.

Small dataset

> N is exactly equal to 20.
>
> Each number in S will be a positive integer less than 10^5.

Large dataset

> N is exactly equal to 500.
>
> Each number in S will be a positive integer less than 10^{12}.
>
> *a) For the Small input, what is the smallest and largest possible sum of a subset of S? How about for the Large input?*

For the Small input, the smallest subset sum would be of a single element $sum\{1\} = 1$, while the largest subset sum would be of all of the largest possible integers $sum\{99999, 99998, 99997 \ldots\} = 1999790$. For the Large input, the smallest sum is still 1, but the largest sum is approximately 500×10^{12}.

b) Describe how you could write a dynamic programming solution to the Small input by relating larger subset sums to smaller ones. Why won't this approach work for the Large subset?

We can construct an array of size 1999790 that contains bitmasks for subsets that sum to a particular integer. We initialize the array with Nones, then fill it in as follows: the subset S_x for some sum x is either $\{x\}$ if $x \in S$ or $S_{x-n} + \{n\}$ if S_{x-n} exists for some $n \in S$. We can simply loop through each possible subset sum, filling it in if we can, and if we ever try to fill in a sum that already exists, then we report those two subsets. Unfortunately, for the Large input, we would need an array of size 5×10^{14}, which is too large to work with.

c) What are the total possible number of subsets for the Large inputs? How about only subsets of size 6? How does this suggest a simple randomized approach to solving the problem? Implement a randomized algorithm to the Large input, test it, and attach it to your homework.

There are 2^{500} possible subsets for the Large input, much larger than the largest possible subset sum, meaning (by the pigeonhole principle) there *must* be multiple ways of reaching some sum, and because there are so many subsets, paired sums are very likely to be found fairly quickly. Even when only choosing subsets of size 6, we have $\binom{500}{6} \approx 2 \times 10^{13}$ total subsets, which is close enough to the total number of sums that we are still very likely to be able to find matching subsets of size 6! Thus, we'll simply keep generating random size 6 subsets, hashing them in a dictionary keyed to their sums. When we find another random subset that has the same sum as one we've already generated, we'll return both of them.

```
In []: import sys
       import random

       def solve_prob(nums):
           d = {}
           N = len(nums)
           while True:
               S = (sorted(random.sample(nums, 5)))
               total = sum(S)
               if total in d and d[total] != S:
                   return d[total], S
               d[total] = S

       infile = open("%s" % sys.argv[1], 'r')
       outfile = open("%s.out" % sys.argv[1][:-3], 'w')
       cases = int(infile.readline().strip())
       for i in range(cases):
           nums = [int(num) for num in infile.readline().split()][1:]
           set1, set2 = solve_prob(nums)
           output = "\n%s\n%s" % (' '.join(map(str, set1)), ' '.join(map(str, set2)))
           outfile.write("Case #%i: %s\n" % (i+1, output))
           print "Case #%i: %s" % (i+1, output)
           print sum(set1), sum(set2)
       infile.close()
       outfile.close()
```