Problem Set 3 Solutions

Problem 9. *Consider the following* Even-Split *problem: given a set S of N numbers, $S = \{a_1, \ldots, a_N\}$, partition it into two halves, T and $S - T$, such that the sum of the numbers in T is as close as possible to the sum of the numbers in $S - T$. That is, you want to minimize $|\sum_{i \in T} - \sum_{i \in S-T}|$. Use dyanmic programing to solve this problem. Make your algorithm as efficient as possible. Compute the running time of your algorithm. Then write a computer program to solve the the following instance:*

```
2837, 2742, 1992, 1999, 2108, 2123, 2652, 3214, 3367, 3855,
3912, 4115, 4188, 4723, 5623, 5777, 5909, 6000, 6108, 6756
```

Do you find the answer surprising? Explain.

The problem is very similar to Subset Sum, and the solution we gave for Subset Sum only needs to be modified a little bit: instead of looking to see if array element $A[B]$ is nonzero, where B is our target, we will find the largest nonzero entry in A which is less than or equal to our target. The target is half of the sum of the numbers.

I wrote my program under Microsoft Visual C++ and it had not problem dealing with numbers of this size. Some students told me that Turbo C++ couln't handle an array of size 40,000, so I told them to use only the first 15 numbers (or whatever). Here is that program and its execution.

```
/*
 * Problem 9 - COMP 494 - Phillip Rogaway
 * 20 July 2542
 */

#include <iostream.h>
const int c[]={0,
   2837, 2742, 1992, 1999, 2108,
   2123, 2652, 3214, 3367, 3855,
   3912, 4115, 4188, 4723, 5623 };
//,5777, 5909, 6000, 6108, 6756   // Too big for Turbo C++   Fine under MS Visual C++

const N=15;      // the number of numbers in the list c, excluding 0
const BIG=25000; // a number bigger than the target

void main()
{
   int   i, A[BIG];
   unsigned long long j, target, sum=0;

   for (i=1; i<=N; i++)
      sum += c[i];
   target = sum/2;
   cout << "The sum of the first "<< N << " numbers is " << sum << ",\n"
        << "and so my target is " << target << endl;

   for (i=1; i<BIG; i++)
      A[i]=0;
   A[0]=1;
```

```
    for (i=1; i<=N; i++) {
        for (j=target; j>=c[i]; j--) {
            if (A[j]==0 && j-c[i]>=0 && A[j-c[i]]>0)
                A[j] = i;
        }
    }
    for (i=target; A[i]==0; i--) ;
    cout << "The best solution has a split of " << i << " and "
         << sum-i << "\n";

    sum = 0;  // let's check our work!
    while (i>0) {
        cout << "Take item " << A[i] << ", which is " << c[A[i]] << endl;
        sum += c[A[i]];
        i -= c[A[i]];
    }
    cout << endl << "As a check: the above items sum to " << sum << endl;
}
```

```
The execution looks like this

The sum of the first 15 numbers is 49450,
and so my target is 24725
The best solution has a split of 24725 and 24725
Take item 14, which is 4723
Take item 11, which is 3912
Take item 9, which is 3367
Take item 8, which is 3214
Take item 7, which is 2652
Take item 6, which is 2123
Take item 3, which is 1992
Take item 2, which is 2742

As a check: the above items sum to 24725
```

So we found an exact split. This isn't too surprising. After all, there are $2^{15} = 32768$ subsets of 15 numbers, and we are asking if one of them adds up to 24725. The subsets add up to numbers between 0 and 49450, but most subsets can be expected to sum to something closer to the middle.

Problem 10. *Let $x = x_1 \dots x_N$ and $y = y_1 \dots y_M$ be strings. We say that $z = z_1 \dots z_{N+M}$ is a shuffle of x and y if z can be formed by interspersing the characters from x and y in a way that maintains the left-to-right ordering of the characters from each string. For example, `thhelerloe` is a shuffle of `hello` and `there`. Devise an algorithm that, given strings x, y and z, determines if z is a shuffle of x and y. Analyze your algorithm's running time.*

(Note that I changed n and m to capital letters to be like the examples we did in class.)

For $0 \le n \le N$ and $0 \le m \le M$, let $A[n,m] = \mathsf{True}$ if $z_1 \cdots z_{n+m}$ is a shuffle of $x_1 \cdots x_n$ and $y_1 \cdots y_m$, and False otherwise. Then for each $0 \le m \le M$ we have that $A[0,m] = \mathsf{True}$ iff $y_1 \cdots y_m = z_1 \cdots z_m$. For each $0 \le n \le N$ we have that $A[n,0] = \mathsf{True}$ iff $x_1 \cdots x_n = z_1 \cdots z_n$. For $1 \le n \le N$ and $1 \le m \le M$ note that

$$A[n,m] \quad = \quad (A[n-1,m] \wedge x_n = z_{n+m}) \vee (A[n,m-1] \wedge y_m = z_{n+m}).$$

This is because if $z' = z_1 \cdots z_{n+m}$ is a shuffle of $x' = x_1 \cdots x_n$ and $y' = y_1 \cdots y_m$ then the last character of z' is the last character from one of x' and y' (say x'), and then the rest of z' has to be a shuffle of y' and the rest of x'. The above is easy to calculate by filling in a table in the natural way. Time is $\Theta(NM)$.