Homework 9

Robbie McKinstry, Jack McQuown, Cyrus Ramavarapu

22 September 2016

Problem 10:

Problem 11:

Problem 17:

To develop a dynamic programming algorithm to determine if the set of gems $\mathcal{G}$ can be partitioned such that the two partitions $\mathcal{P}$ and $\mathcal{Q}$ have the same value and the same number of rubies and emeralds all possible partitions can be initially considered. Let the total value of all the gems be $\mathcal{L}$ and the total number of gems be n.

In considering every possible partition, a new gem is added to either $\mathcal{P}$ or $\mathcal{Q}$. This gem, labeled g_i, will have a value v_i. Additionally, g_i will be either an *emerald* or a *ruby*. Since the value between the two partitions has to be equal and the quantity of the gem type also has to be the same, these values need to be tracked.

Tracking these values as the tree of all possible partitions is generated gives the following pruing rules.

1. If the value of either partition becomes greater than $\mathcal{L}/2$ the tree can be pruned. This is because if a partition has a value greater than half of the max value, the other partition can never catch up.

2. If the number of emeralds or rubies in either partition becomes greater than n, the tree can be pruned. This is because the other partition will never be able to get enough gems of the necessary type to restablish balance.

3. If two branches have the same value, same number of rubies, and the same number of emeralds in both partitions, one branch can be arbitrarily pruned.

Using these pruning rules gives the following polynomial time algorithm.

Function: *Gem Partition*
Input: $\mathcal{G}$
Globals: $A[\ ][\ ][\ ][\ ][\ ][\ ][\ ][\ ][\ ][\ ]$

```
/* iterate through possible gems                                    */
for l = 1 to n do
    /* iterate through values for P                                 */
    for h = 1 to L/2 do
        /* iterate through rubies for P                             */
        for j = 1 to n/2 do
            /* iterate through emeralds for P                       */
            for k to n/2 do
                /* iterate through values for Q                     */
                for x = 1 to L/2 do
                    /* iterate through rubies for Q                 */
                    for y = 1 to n/2 do
                        /* iterate through emeralds for Q           */
                        for z to n/2 do
                            if A[l,h,j,k,x,y,z] is defined then
                                if g_l is a ruby then
                                    A[l+1, h+v_l, j+1, k, x, y, z] = 1
                                    A[l+1, h, j, k, x+v_l, y+1, z] = 1
                                else
                                    A[l+1, h+v_l, j, k+1, x, y, z] = 1
                                    A[l+1, h, j, k, x+v_l, y, z+1] = 1

/* check if partition is possible                                   */
for i = 1 to L/2 do
    for j = 1 to n/2 do
        for k = 1 to n/2 do
            if A[n,i,j,k,i,j,k] is defined then
                if A[n,i,j,k,i,j,k] == 1 then
                    Return: 1
Return: 0
```

The answer to if a partition is possible will be found by searching the space at the n^{th} level where the values for partitions $\mathcal{P}$ and $\mathcal{Q}$ are the same while also having the same number of rubies and emeralds.

This algorithm runs in $\mathcal{O}\left(n^5\mathcal{L}^2\right)$ time. This is a $poly(n + \mathcal{L})$ if $poly(x) = x^7$.