Benjamin Brook
CS 124
Michael Mitzenmacher
April 25, 2016

Programming Assignment 3

Give a dynamic programming solution to the Number Partition problem.

We create a boolean array T of size $b+1$, where each element is initialized to False. The ith element will be set to True if there is a subset of A whose sum is equal to i. If $T[n/2]$ is True, then there is exists a subset whose sum is exactly half the total of A, thereby providing a solution to the Number Partition problem. We begin by setting $T[0]$ to True. Then we take the first element of A, a_1, and set the index $T[a_1]$ to True. Next, we take a_2, and set $T[a_1 + a_2]$ and $T[a_2]$ to True. We continue by taking the a_ith element, and for each index j at which $T[j]$ is True, we set $T[j + a_i]$ to True (it should be noted that we move through T from right to left, to avoid hitting an element in T that was just added on that same iteration. Once we've iterated through each element in A, we are done populating T, and we simply return $T[n/2]$ to determine whether there exists a partition of A where the two subsets have equal sums of their elements. This runs in $O(nb)$ since the algorithm must run through the table T of size $b+1$ for each of n elements in A.

Explain briefly how the Karmarkar-Karp algorithm can be implemented in $O(n \log n)$ steps, assuming the values in A are small enough that arithmetic operations take one step.

We do $\log n$ iterations through the array of length n. This is because the number of non-zero integers is halved with each pass over the array. Therefore, after $\log n$ passes over the array (of size n), there will be only a residual number remaining. So the KK algorithm can be implemented in $O(n \log n)$ time if we assume $O(1)$ for arithmetic operations.