CSCI-GA.1170-001/002 Fundamental Algorithms November 19, 2015

Solutions to Problem 2 of Homework 8 (12 points)

Name: GOWTHAM GOLI (N17656180) *Due: Tuesday, November 17*

(a) (8 points) You are given n integers $a_1, \ldots, a_n \geq 0$, and a target $T \geq 0$. Design an $O(nT)$ algorithm to determine is there exists a subset of the a_i's that sum to T. For example, if $n = 5$ and $a_1 = 3$, $a_2 = 5$, $a_3 = 2$, $a_4 = 11$, $a_5 = 3$, then the answer is YES for $T = 10$ (e.g., $3 + 5 + 2 = 10$), but NO for $T = 9$.

Solution:

Define a 2d-array S of size nT such that $S[i, T]$ is *true* if there is a subset of $a_k's$ in the set of integers $\{a_1, \ldots, a_i\}$ such that their sum evaluates to T and $S[i, T]$ is false otherwise.

- $S[i, 0] = true,\ \forall i$. We can choose the empty set $\emptyset$ for the sum to evaluate to 0
- $S[0, T] = false,\ \forall T$. The sum of 0 number of elements can never evaluate to T
- There can be a subset of $a_1, \ldots, a_i$ adding up to T if there is a subset of $a_1, \ldots, a_{i-1}$ adding up to T or if there is a subset of $a_1, \ldots, a_{i-1}$ adding up to $T - a_i$

Therefore from the above three cases we have the following recurrence equation

$$S[i, T] = \begin{cases} true & \text{if } T = 0 \\ false & \text{if } i = 0 \\ S[i-1, T] \vee S[i-1, T - a_i] & \text{otherwise} \end{cases}$$

□

The first row and the first column of the matrix are filled using the two base cases and then the entire matrix is constructed iteratively. Time taken to fill each of entry matrix takes only $O(1)$ time as it involves taking *or* between two booleans. Therefore the total time taken to fill S is $O(nT)$ and in the end $S[n, T]$ is returned.

(b) (4 points) Solve part (a) using only T *bits* of extra memory (in addition to the a_i's themselves).

Solution:

Define a boolean array B of size $T + 1$ and initialize all it's entries to *false*.

$B[k]$ is YES if there exists a subset of a_i's whose sum evaluates to k. Therefore $B[0] = 0$, since the empty set evaluates to sum 0.

If $a_i \leq T$ then $B[a_i] = true$ i.e choose the single element a_i as the subset, so the sum evaluates to a_i

Now for all a_i, starting from 0 traverse up to T and if $B[j]$ is true then set $B[j + a[i]]$ to *true* i.e if there is a subset of elements such that their sum evaluates to j then add $a[i]$ to these subset so that the sum then evaluates to $j + a[i]$

```
1  Algorithm: SubsetSum(n, T)
2  B ← NewArray(T + 1)
3  for i ← 0 to T do
4  |  B[i] ← false
5  end
6  for i ← 1 to n do
7  |  if a[i] ≤ T then
8  |  |  B[a[i]] ← true
9  |  end
10 end
11 for i ← 1 to n do
12 |  for j ← 0 to T do
13 |  |  if B[j] ← true then
14 |  |  |  B[j + a[i]] ← true
15 |  |  end
16 |  end
17 end
18 Return B[T]
```

Algorithm 1: Dynamic Programming Algorithm to determine if there exists a subset of a_i's whose sum evaluates to T

□