ProgSet 3

CS 124: Data Structures and Algorithms

Due: Thursday, April 18, 2024 **Denny Cao and Ossimi Ziv**

---

§1 Number Partition

Input	: A sequence of n numbers $A = \{a_1, a_2, \ldots, a_n\}$
Output	: A sequence of n numbers $S = \{s_1, s_2, \ldots, s_n\}$ of signs $s_i \in \{+1, -1\}$ such that the residual sum of the numbers in A is minimized.

Computational Problem: Number Partition

Claim 1.1 — `Number Partition` can be solved in pseudo-polynomial time.

Proof. Suppose the sequence of terms in A sum up to some number b. Then each of the numbers in A has at most $\log b$ bits. We will show there exists a dynamic programming algorithm that solves the `Number Partition` problem that takes time polynomial nb:

- **Subproblems:** Let $D[i, j]$ be whether it is possible for $A[0, i]$ to sum up to j.
- **Recurrence:** The recurrence relation is given by:

$$D[i, j] = D[i-1, j + a_i] \vee D[i-1, j - a_i]$$

  Our recurrence is correct because if $D[i-1, j + a_i]$ is true, then we can subtract a_i from the sum to get j. Similarly, if $D[i-1, j - a_i]$ is true, then we can add a_i to the sum to get j. In doing so, we obtain every possible sum of the first i elements of A.

- **Topological Order:** We solve the subproblems in increasing order of i and j.
- **Base Case:** $D[0, 0] =$ True and $D[i, j] =$ False for all other i, j.
- **Original:** The original problem is to find the smallest j such that $D[n, j]$ is true.
- **Time Complexity:** The time complexity of this algorithm is $O(nb)$. This is because for each subproblem we check if b sums are possible (The maximum sum is b). When checking if a sum is possible, we take $O(1)$ time to check 2 previous subproblems. Thus, the total time complexity is $O(nb)$ to fill the table. Iterating to find the smallest j such that $D[n, j]$ is true takes $O(b)$ time. Therefore, the total time complexity is $O(nb) + O(b) = O(nb)$.

Therefore, the `Number Partition` problem can be solved in pseudo-polynomial time. □