CS3230: Assignment for Week 10 Solutions

Due: Sunday, 10th Apr 2022, 11:59 pm SGT.

---

1. Suppose the input formula is ϕ. Construct a string a as follows. Run the decision algorithm on ϕ with the first variable x_1 set to 1. If the decision is YES, set $a_1 = 1$ and otherwise, $a_1 = 0$. Now, run the decision algorithm on ϕ with $x_1 = a_1$ and $x_2 = 1$. If the decision is YES, set $a_2 = 1$ and otherwise $a_2 = 0$. Similarly, for the i-th variable, set $x_1, \ldots, x_{i-1}$ to $a_1, \ldots, a_{i-1}$ and set x_i to 1, and run the decision algorithm on ϕ with these variables hard-coded. If the decision is YES, set $a_i = 1$, and otherwise, $a_i = 0$. It is clear that at the end, a is a satisfying assignment if there exists one, and that the resulting algorithm runs in polynomial time.

2. First, suppose that there is a polynomial-time algorithm $\mathcal{A}$ for Longest Path. To solve Longest Path Length with parameter k, you run $\mathcal{A}$; suppose its answer is ℓ. If $\ell \geq k$, you answer YES; otherwise, you answer NO.

   Next, suppose that there is a polynomial-time algorithm $\mathcal{B}$ for Longest Path Length. To solve Longest Path, you run $\mathcal{B}$ with $k = |V| - 1, |V| - 2, \ldots$, until the answer is YES. (Here, V denotes the set of vertices of the graph.) Since you call $\mathcal{B}$ a polynomial number of times, your algorithm runs in polynomial time. Note that you can speed this up by using binary search, so that you call $\mathcal{B}$ at most $O(\lg |V|)$ times instead of $O(|V|)$ times.

3. (a) **Solution 1:** Clearly, the answer to Partition is YES if and only if there exists a subset of the integers that sum to $M/2$. (In particular, if $M/2$ is not an integer, we can immediately answer NO.) Similarly to Knapsack, we use dynamic programming. For $0 \leq i \leq n$ and $0 \leq j \leq M/2$, let $t[i, j]$ be a boolean variable denoting whether there exists a subset of $a_1, \ldots, a_i$ that sum to j. We have the following recurrence:

$$t[i,j] = \begin{cases} 1 & \text{if } i = j = 0 \\ 0 & \text{if } i = 0 \text{ and } j > 0 \\ t[i-1,j] & \text{if } i > 0 \text{ and } a_i > j \\ \max\left(t[i-1,j], t[i-1,j-a_i]\right) & \text{otherwise} \end{cases}$$

That is, there exists a subset of $a_1, \ldots, a_i$ that sum to j if and only if there exists a subset of $a_1, \ldots, a_{i-1}$ that sum to either j or $j - a_i$. The table has $O(nM)$ entries, and computing each entry takes $O(1)$ time. Hence, the running time is $O(nM)$.

Solution 2: Use the reduction from PARTITION to KNAPSACK from Week 12 Tutorial, which runs in time $O(n \lg M)$, and use the dynamic programming algorithm for KNAPSACK from Lecture 8, which runs in time $O(nM)$.

(b) This is not a polynomial-time algorithm, because the input size is $O(n \lg M)$.