Lab 9 (Techniques)

Collaboration level 0 (no restrictions). Open notes.

---

1. **Finding the singleton:** You are given a sorted array of numbers where every value except one appears exactly twice, and one value appears only once. Design an efficient algorithm for finding which value appears only once. Here are some example inputs to the problem:

   $$1, 1, 2, 2, 3, 4, 4, 5, 5, 6, 6, 7, 7, 8, 8$$

   $$10, 10, 17, 17, 18, 18, 19, 19, 21, 21, 23$$

   $$1, 3, 3, 5, 5, 7, 7, 8, 8, 9, 9, 10, 10$$

   Note: A general solution should not assume anything about the numbers in the array; specifically, they may not be in a small range, and may not be consecutive. Hint: Look at the indices in the array of the pairs that are equal, before the singleton, and after the singleton, and think along the lines of binary search, aiming for $O(\lg n)$.

2. **Unbounded knapsack:** We have n items, each with a value and a positive weight as given by two arrays: Item i has value $v[i]$ and weight $w[i]$. We have a knapsack that holds maximum weight W and we have *infinite copies* of each item.

   Given $W, \{w_1, ..., w_n\}$ and $\{v_1, ..., v_n\}$, we want to find the maximal value that can be loaded into the knapsack.

   (a) A friend proposes the following algorithm: start with the item with the largest cost-per-pound, and pick as many copies as fit in the backpack. Repeat.

   Show your friend this is not correct by coming up with a small example where this algorithm leads to a solution that is not optimal.

   (b) Suppose the optimal solution contains item i. Then the remaining part of the optimal solution must be an optimal solution for .... .

   (c) Come up with a recursive definition. Define your sub-problem and state clearly what its argument(s) represent and what it returns.

   Hint: we have infinite supplies of each item. So we don't need to keep track of which items we already included, only of the remaining space in the knapsack. Consider all the choices, and pick the best.

   (d) Develop a DP solution — either top-down or bottom up. What is the running time of your algorithm ?