A Good Problem – Solution Outline

David Wu

Problem Restatement

- We start with an array $a = [0, 0, \ldots, 0]$ of length n.
- We want to transform a into target array b.
- Allowed operations:
  1. **Type 1:** `1 x` $\rightarrow$ add 1 to all elements equal to x.
  2. **Type 2:** `2 i` $\rightarrow$ add 1 to the element at index i.
- Constraint: total operations ≤ 20000, with $n \leq 1000$, $b_i \leq n$.

Algorithm Idea: Divide and Conquer

- Consider the value range $[l, r]$ with midpoint $mid = \lfloor (l + r)/2 \rfloor$.
- Invariant before calling `solve(l,r)`: every element $a[i]$ with target $b[i] \in [l, r]$ currently equals l.
- Steps in `solve(l,r)`: if l==r return
  1. For all i with $b[i] \in (mid, r]$: perform `2 i` (lift them from l to $l + 1$).
  2. For $j = l + 1, \ldots, mid$: perform `1 j` (globally push elements in step 1 from $l + 1$ to $mid + 1$).
  3. Recurse on $[l, mid]$ and $[mid + 1, r]$.

Correctness

Invariant Preservation

- Left half $[l, mid]$: elements remain at value $l \Rightarrow$ invariant holds for recursive call.
- Right half $(mid, r]$: after one `2 i` and $(mid - l)$ global operations, they reach $mid + 1$. $\Rightarrow$ invariant holds for recursive call.

Termination

When $l = r$, all elements with target $b[i] = l$ are already equal to l, so no further operations are needed.

Complexity Analysis

- Recursion depth $= O(\log n)$ (value range $[0, n]$ is split by halves).
- At each level:
  - Type 2 operations: each element is updated at most once per level $\Rightarrow O(n)$.
  - Type 1 operations: all elements in b is at most n $\Rightarrow$ *at most $n/2$ operations* $\Rightarrow O(n)$.
- Total operations:

$$O(n) \text{ per level } \times O(\log n) \text{ levels } = O(n \log n).$$

Conclusion

- Divide-and-conquer on the value range ensures correctness.
- Each recursive step maintains the invariant.
- Total number of operations is $O(n \log n)$, well within the 20000 bound for $n \leq 1000$.