Counting Sort

Yufei Tao

Department of Computer Science and Engineering
Chinese University of Hong Kong

We already know that sorting n integers can be done in $O(n \log n)$ time. Today, we will see a variant of the sorting problem where the integers come from a small domain.

Sorting in a Small Domain)

Problem Input:

A set S of n integers is given in an array of length n. Every integer is in the range of $[1, U]$. It holds that $U \geq n$.

Goal:

Produce an array that stores the integers of S in ascending order.

Counting Sort

Step 1: Let A be the array storing S. Create an array B of length U. Initialize B by setting all its cells to 0.

Step 2: Carry out the following for every $i \in [1, n]$: set $B[A[i]] = 1$.

Step 3: Generate the sorted order as follows:

for $x = 1$ to U

  if $B[x] = 1$ **then** append integer x to A.

Example

At the beginning

13	2	8	4	11	12																								

Initialize array B (assuming $U = 16$)

13	2	8	4	11	12		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							

$\longleftarrow A \longrightarrow$ $\longleftarrow B \longrightarrow$

Setting n cells of B to 1

13	2	8	4	11	12		0	1	0	1	0	0	0	1	0	0	1	1	1	0	0	0							

$\longleftarrow A \longrightarrow$ $\longleftarrow B \longrightarrow$

Final sorted list

2	4	8	11	12	13		0	1	0	1	0	0	0	1	0	0	1	1	1	0	0	0							

$\longleftarrow A \longrightarrow$ $\longleftarrow B \longrightarrow$

Analysis of Counting Sort

Steps 1 and 3 take $O(U)$ time.
Step 2 takes $O(n)$ time.

Therefore, the overall running time of counting sort is $O(n + U) = O(U)$.

For small $U = O(n)$ (e.g., $1000n$), the counting sort runs in $O(n)$ time.

It is important to note that counting sort does **not** improve merge sort in general! $O(n + U)$ is **incomparable** to $O(n \log n)$. When $U = O(n)$, counting sort is faster, but when $U = \Omega(n^2)$, merge sort is faster.