Counting sort

Goldsmiths Computing

January 13, 2019

Motivation

- specialised sorting algorithm
- more information than just comparisons available
  - *e.g.* that items are keyed by small integers
  - recall sorting by age
- for when $O(N \log N)$ isn't good enough

Components

1. for each key, (efficiently) determine how many elements of the input are smaller than that element;
2. for each key, directly compute the position of that key in the sorted result;
3. for each element, place it in its final position.

Counting sort

```
function COUNTING-SORT(A,k)
    R ← new array(LENGTH(A))
    C ← new array(k)
    for 0 ≤ j < LENGTH(A) do
        C[A[j]] ← C[A[j]] + 1
    end for
    for 0 < i < k do
        C[i] ← C[i] + C[i-1]
    end for
    for LENGTH(A) > j ≥ 0 do
        R[C[A[j]]-1] ← A[j]
        C[A[j]] ← C[A[j]] – 1
    end for
    return R
end function
```

Complexity analysis

Space

- one temporary array C
- return value array R

$$\Longrightarrow \Theta(N + k)$$

Time

- iterate over input array A
- iterate over temporary array C
- iterate over return value array R

$$\Longrightarrow \Theta(N + k)$$

Work

1. Implement counting sort for arrays of integers between 0 and 100. How will you test your implementation?
2. Questions from CLRS

   Exercises 8.2-1, 8.2-4

   8-2 Sorting in place in linear time