Cupsort

Luc Haller

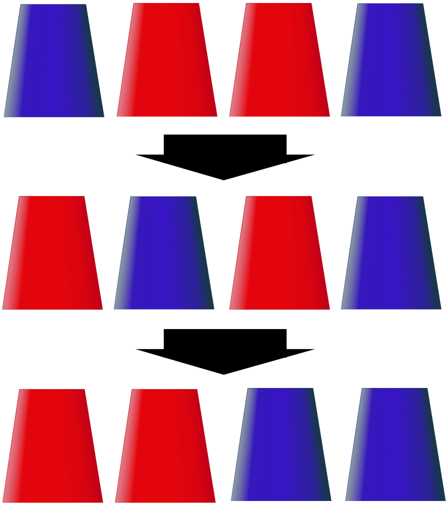
Swiss Olympiad in Informatics

January 7, 2017

Task story

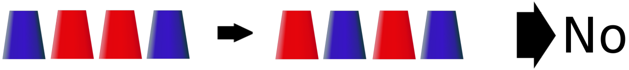
- Sort cups standing in a row by color.
- How many times do we need to swap two adjacent cups?

Example



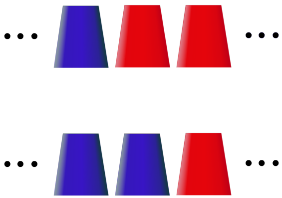
Subtask 1: Problem statement

- Only two colors: red and blue (in the input: 0 and 1)
- Want to put all the red cups left of the blue cups.
- Only yes/no: Can we sort the cups with only 0 or 1 swap?
- Example:



Solution

- Idea:
  - Skip to the first blue cup
  - Iff there is a red cup farther than directly behind it, it is impossible to sort with one swap.



- $\mathcal{O}(n)$ running time, input size is also $\mathcal{O}(n)$

Subtask 2

- Same as before, but want to know how many swaps we need.
- Observation: We need to swap every blue cup with every red cup that comes after it.
- These are all swaps: no need to swap cups of the same color.

Solution

- Counter for how many blue cups we have seen already, add this to the result at every red cup we encounter.
- Again $\mathcal{O}(n)$ running time

Subtask 3

- Now, there are more than two colours, but all cups have distinct colours.
- Idea: Use a suitable sorting algorithm, modifying it to count the number of swaps that it makes.
- Note: The number of "swaps" is mathematically speaking the number of inversions in the list.

Easy sorting algorithm: Bubble Sort

- Compare each number with the one after it, swap if first one is larger.
- Repeat until sorted.
- $\mathcal{O}(n^2)$ running time
- Note: The number of inversions is also $\mathcal{O}(n^2)$, so if we want to be faster, we have to sometimes simulate multiple swaps with one step of our algorithm.

Faster sorting algorithm: Merge Sort, $\mathcal{O}(n \log n)$

Idea:

- Split list into two halves, sort these recursively.
- Merge the two now sorted halves, which we can do in linear time using the knowledge that they are sorted.
- The inversion counting happens in the merging step: Each element of the second list needs to be swapped with all larger elements of the first half.

Subtasks 4 and 5

- For Subtask 3, Bubble Sort was fast enough.
- Subtask 4: Multiple cups may have the same colour.
- Only difference in solution compared to Subtask 3: Take care not to swap cups of the same colour.
- Subtask 5: More cups than before: Need an $\mathcal{O}(n \log n)$ solution to solve it fast enough.
- Also, cup groups instead of single cups: Numbers to be sorted have weights. The number of cup swaps to swap two groups is the product of the group sizes.

Summary

- More detailed write up in solution booklet.
- Implementations (C++ code) in solution booklet.
- Other solutions were possible, particularly for Subtask 1 there are many different ad hoc approaches, and for Subtask 5 there is an alternative $\mathcal{O}(n \log n)$ solution (in solution booklet).