CS 325 Group Project 1

Robert Detjens, Srikar Valluri, Felix Brucker

Scenario

You are a visitor at a political convention with n delegates; each delegate is a member of exactly one political party. It is impossible to tell which political party any delegate belongs to; in particular, you will be summarily ejected from the convention if you ask.

However, you can determine whether any pair of delegates belong to the same party by introducing them to each other. Members of the same political party always greet each other with smiles and friendly handshakes; members of dffierent parties always greet each other with angry stares and insults.

Suppose more than half of the delegates belong to the same political party. Describe an efficient algorithm that finds out the size of the majority party. The efficiency of your algorithms is measured in terms of the number of pairs of delegates that you introduce to each other. We expect that you need about $O(n \log n)$ handshakes.

Description

Our algorithm uses a recursive method to divide and conquer the convention to find the majority in $n \log n$ time.

First, we create a list of all n delegates (so that we can split the delegates up) and call our recursive function on this list.

The recursive function first halves the list and calls itself on each half to find the majority party and size for each half. It is guaranteed that one of the two halves has the same majority as the full array, so one of the two majority parties will be the majority for the whole. Once each half has been computed, it looks through its list of delegates and counts how many delegates are in the two majority parties from either half. It then returns the size of and a delegate from the party with more people.

If the size of the delegate list passed into the recursive function is of length 1, that is our base case. We return the (only) delegate from the list and a majority size of 1, since that is the only possible outcome for this array size.

Runtime Analysis

The runtime of this algorithm is similar to merge sort, as we are recursively divide-and-conquering the array in halves. We split the array into halves $\log n$ times, and for each array iterate over it to find the largest majority. Every delegate must be compared against the two majority parties (linear process: $O(n)$) at each level of recursion ($\log n$ times). This gives us a runtime of $O(n) * \log n$ or $O(n \log n)$.