1 Specifications

This program performs a modified radix sort that performs a minimal amount of comparisons (and thereby reducing the effects of branch misprediction) in exchange for a naive allocation scheme. It also (ab)uses bitwise operations to minimize space. The order is **NOT** the one defined by *strcmp*; rather, it is defined as the slightly more intuitive AaBbCcDd....Zz order.

2 Program Model

As listed in these sources, this radix sort performs several precomputations in order to avoid useless comparisons due to differing lengths.

In order to avoid checking for different lengths, this sort first sorts the list of words by length, and then goes backwards to avoid unnecessary comparisons.

In order to avoid checking empty buckets, a list of bits is kept that each corresponds to a letter for each length. The buckets are then tallied using a naive multiplication technique.

Popcount is implemented using a broadword 64-bit algorithm.

Thanks to the quirks of ASCII, the order is encoded in an essentially O(1) operation.

3 Analysis

As in the previously mentioned sources, the big O running time for this program is linear w.r.t. the number of words (assuming the basic operation is a placement into a bucket and that several key non-constant operations are actually constant).

If we do not count for loop comparisons, then this program performs 0 comparisons during the bucket sort.