**2-sum.** Given an array a*a* of *n* 64-bit integers and a target value *T*, determine whether there are two distinct integers *i* and *j* such that *ai*​+*aj*​=*T*. Your algorithm should run in linear time in the worst case.

**American flag sort.** Given an array of *n* objects with integer keys between 0 and *R*−1, design a linear-time algorithm to rearrange them in ascending order. Use extra space at most proportional to R.

**Cyclic rotations.** Two strings *s* and *t* are *cyclic rotations* of one another if they have the same length and s*s* consists of a suffix of *t* followed by a prefix of *t*. For example, "suffixsort" and "sortsuffix" are cyclic rotations.

Given *n* distinct strings, each of length *L*, design an algorithm to determine whether there exists a pair of distinct strings that are cyclic rotations of one another. For example, the following list of *n*=12 strings of length *L*=10 contains exactly one pair of strings ("suffixsort" and "sortsuffix") that are cyclic rotations of one another.

1

2

3

algorithms polynomial sortsuffix boyermoore

structures minimumcut suffixsort stackstack

binaryheap digraphdfs stringsort digraphbfs





The order of growth of the running time should be *nL*2 (or better) in the worst case. Assume that the alphabet size *R* is a small constant.

*Signing bonus*. Do it in *NnL* time in the worst case.