CS 577-2: Introduction to Algorithms Fall 2020

Practice Problems for Midterm Exam 2

Instructor: Dieter van Melkebeek

The following are some old exam problems and one problem from last year's ICPC regional that you can use as design practice. Don't forget to do the additional problems on each of the assignments!

1. You are given an even number of distinct positive integers $a_1, a_2, \ldots, a_{2n}$, and want to find the maximum value of

$$\sum_{(i,j)\in P} (a_i - a_j),$$

where P ranges over all collections of disjoint pairs (i, j). Design an $O(n)$ algorithm for this problem.

For example, if the given input sequence is $(a_1, a_2, a_3, a_4) = (7, 4, 5, 3)$, then an optimal choice of P is $\{(1, 2), (3, 4)\}$, resulting in the maximum value of $(7 - 4) + (5 - 3) = 5$.

2. For a dinner after a sports event, you want to assign seats such that no two members of the same team sit at the same table. There are n teams of size $s_1, s_2, \ldots, s_n$, respectively, and m tables, of size $t_1, t_2, \ldots, t_m$, respectively.

Design an algorithm that determines whether such a seating arrangement is possible. Your algorithm should run in time $O(N \log N)$, where $N = (\sum_{i=1}^{n} s_i) + (\sum_{j=1}^{m} t_j)$.

3. You collect coupons and aim to have a collection containing each of the n types of coupons that exist. Your current collection consists of exactly n coupons but contains duplicates. You are hoping to achieve your goal by a sequence of exchanges of a coupon of one type for a coupon of another type, but only certain types of exchanges are possible.

For example, suppose that your current collection consists of one coupon of type 1, and 3 coupons of type 2, and that it is possible to exchange a coupon of type 2 for a coupon of type 3, and a coupon of type 3 for a coupon of type 4. In this case, it is possible for you to achieve your goal, namely by exchanging two of your coupons of type 2 for coupons of type 3, and one of the obtained coupons of type 3 for a coupon of type 4.

Develop a polynomial-time algorithm that, given your current collection and the possible exchanges, determines whether your goal is achievable or not.

4. Design an $O(N^3)$ algorithm for the problem "Code Names" of our 2019 ICPC regional contest. See the next two pages for the statement.

Problem B
Code Names

You are given W, a set of N words that are anagrams of each other. There are no duplicate letters in any word. A set of words $S \subseteq W$ is called "swap-free" if there is no way to turn a word $x \in S$ into another word $y \in S$ by swapping only a single pair of (not necessarily adjacent) letters in x. Find the size of the largest swap-free set S chosen from the given set W.

Input

The first line of input contains an integer N ($1 \leq N \leq 500$). Following that are N lines each with a single word. Every word contains only lowercase English letters and no duplicate letters. All N words are unique, have at least one letter, and every word is an anagram of every other word.

Output

Output the size of the largest swap-free set.

Sample Input 1

```
6
abc
acb
cab
cba
bac
bca
```

Sample Output 1

```
3
```

Sample Input 2

```
11
alerts
alters
artels
estral
laster
ratels
salter
slater
staler
stelar
talers
```

Sample Output 2

```
8
```

Sample Input 3	Sample Output 3
6 ates east eats etas sate teas	4