CS 91 USACO

Bronze Division

Unit 3: Problem Solving Using Algorithms

LECTURE 12: COMPLETE SEARCH – PROBLEM SOLVING SESSION B

DR. ERIC CHOU IEEE SENIOR MEMBER

Objectives

- Practice Problem: Dual Palindromes
- December 2017 Problem 1: Blocked Billboard
- December 2017 Problem 2: The Bovine Shuffle

Practice: Dual Palindromes (dualpal)

SECTION 1

Problem Statement

- A number that reads the same from right to left as when read from left to right is called a palindrome. The number 12321 is a palindrome; the number 77778 is not. Of course, palindromes have neither leading nor trailing zeroes, so 0220 is not a palindrome.
- The number 21 (base 10) is not palindrome in base 10, but the number 21 (base 10) is, in fact, a palindrome in base 2 (10101).

Problem Statement

- Write a program that reads two numbers (expressed in base 10):
  - N (1 <= N <= 15)
  - S (0 < S < 10000)
- and then finds and prints (in base 10) the first N numbers strictly greater than S that are palindromic when written in two or more number bases (2 <= base <= 10).
- Solutions to this problem do not require manipulating integers larger than the standard 32 bits.

INPUT FORMAT (dualpal.in):

- A single line with space separated integers N and S.

SAMPLE INPUT:

```
3 25
```

OUTPUT FORMAT (dualpal.out):

- N lines, each with a base 10 number that is palindromic when expressed in at least two of the bases 2..10. The numbers should be listed in order from smallest to largest.

SAMPLE OUTPUT:

```
26
27
28
```

Nature of the Problem

Read in two number from input file. The first number N is for the number of Dual Palindrome integers to be found. The second number S is the exclusive lower bound for the Dual Palindrome numbers

- A number is considered dual palindrome number if the number is palindrome in at least 2 base system (base from 2 to 10).
- Time complexity: 9 * 10000 (Worst Case) The Time complexity for this problem is bounded.

Dec 2017 Problem 1: Blocked Billboard (billboard)

SECTION 2

Problem Statement

- During long milking sessions, Bessie the cow likes to stare out the window of her barn at two huge rectangular billboards across the street advertising "Farmer Alex's Amazingly Appetizing Alfalfa" and "Farmer Greg's Great Grain". Pictures of these two cow feed products on the billboards look much tastier to Bessie than the grass from her farm.
- One day, as Bessie is staring out the window, she is alarmed to see a huge rectangular truck parking across the street. The side of the truck has an advertisement for "Farmer Smith's Superb Steaks", which Bessie doesn't quite understand, but she is mostly concerned about the truck potentially blocking the view of her two favorite billboards.
- Given the locations of the two billboards and the location of the truck, please calculate the total combined area of both billboards that is still visible. It is possible that the truck obscures neither, both, or only one of the billboards.

INPUT FORMAT (file billboard.in):

- The first line of input contains four space-separated integers: x_1 y_1 x_2 y_2, where (x_1,y_1) and (x_2,y_2) are the coordinates of the lower-left and upper-right corners of the first billboard in Bessie's 2D field of view. The next line contains four more integers, similarly specifying the lower-left and upper-right corners of the second billboard. The third and final line of input contains four integers specifying the lower-left and upper-right corners of the truck. All coordinates are in the range -1000 to +1000. The two billboards are guaranteed not to have any positive area of overlap between themselves.

SAMPLE INPUT:

```
1 2 3 5
6 0 10 4
2 1 8 3
```

OUTPUT FORMAT (file billboard.out):

- Please output the total combined area of both billboards that remains visible.

SAMPLE OUTPUT:

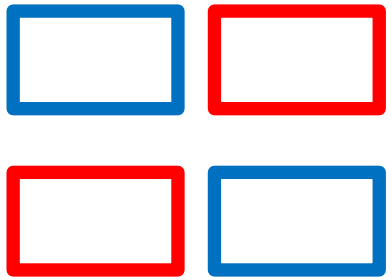
```
17
```

Here, 5 units of area from the first billboard and 12 units of area from the second billboard remain visible.

Nature of Problem

- Basic computational geometry.
- Read in the data as a paralleled array of x[0], y[0] …, x[5], x[5]
- Board 1: (x[0], y[0]), (x[1], y[1])
- Board 2: (x[2], y[2]), (x[3], y[3])
- Board 3: (x[4], y[4]), (x[5], y[5])
- Calculate the area of Board 1 + Board 2 – Overlap(Board 1, Board 3) – Overlap(Board 2, Board 3)

Sorting x1, x2, x3 x4

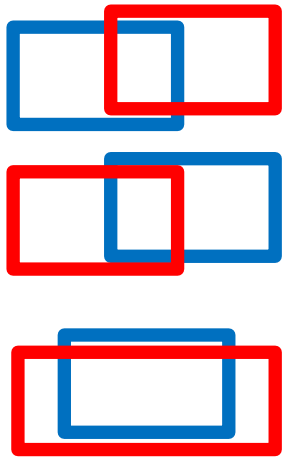


By default, x1 < x2 and x3 < x4.

So, there is an overlap in x-direction if
  x3 < x2 or x1 < x4

They are disjoint if
  x2 < x3 and x4 < x1 ([x1, x2, x3, x4] or [x3, x4, x1, x2])
Otherwise, there is an overlap

Width of the Overlap



Sort the [x1, x2, x3, x4], the width of the overlapped region will be the distance between the middle 2 point among x1, x2, x3, x4.

USACO 2017 December Contest, Bronze
Problem 1. Blocked Billboard

USACO 2018 January Contest, Bronze
Problem 1. Blocked Billboard II

Dec 2017 Problem 2: The Bovine Shuffle (shuffle)

SECTION 3

Problem Statement

- Convinced that happy cows generate more milk, Farmer John has installed a giant disco ball in his barn and plans to teach his cows to dance!
- Looking up popular cow dances, Farmer John decides to teach his cows the "Bovine Shuffle". The Bovine Shuffle consists of his N cows ($1 \le N \le 100$) lining up in a row in some order, then performing three "shuffles" in a row, after which they will be lined up in some possibly different order. To make it easier for his cows to locate themselves, Farmer John marks the locations for his line of cows with positions 1…N, so the first cow in the lineup will be in position 1, the next in position 2, and so on, up to position N.

Problem Statement

- A shuffle is described with N numbers, $a_1 \ldots a_N$, where the cow in position i moves to position a_i during the shuffle (and so, each ai is in the range 1…N). Every cow moves to its new location during the shuffle. Fortunately, all the a_i's are distinct, so no two cows try to move to the same position during a shuffle.
- Farmer John's cows are each assigned distinct 7-digit integer ID numbers. If you are given the ordering of the cows after three shuffles, please determine their initial order.

INPUT FORMAT (shuffle.in):

- The first line of input contains N, the number of cows. The next line contains the N integers $a_1 \ldots a_N$. The final line contains the order of the N cows after three shuffles, with each cow specified by its ID number.

SAMPLE INPUT:

```
5
1 3 4 5 2
1234567 2222222 3333333 4444444 5555555
```

OUTPUT FORMAT (shuffle.out):

- You should write N lines of output, with a single cow ID per line, specifying the order of the cows before the three shuffles.

SAMPLE OUTPUT:

```
1234567
5555555
2222222
3333333
4444444
```

Nature of Problem

- Finding the reverse shuffle. Same technique can be applied to decryption algorithm.
- Undo the shuffle for three time.
- Use x-1 indexing.

```
for (int i=0; i<N; i++){
  a[i] = input.nextInt()-1;
}
for (int i=0; i<N; i++){
  r[a[i]] = i;
}
String[] R3 = new String[N];
for (int i=0; i<N; i++){
   R3[i] = input.next();
}
```