Homework #1

Due Time: 2015/10/12 (Mon.) 12:00
Contact TAs: `ada@csie.ntu.edu.tw`

Instructions and Announcements

- For the first time submitting your ADA 2015 HW, please create a repository for ADA on bitbucket (`https://bitbucket.org`) and share the repository with user ada2015 with read permission.
- You have to login to the judge system (`http://ada2015.csie.org`) to bind the bitbucket repository to your account. For programming problems (those containing "**Programming**" in the problem topic), push your source code to your bitbucket repository. After you push the source code to repository, you can run the judge and get the score. Please refer to the guide of the judge system on its index page for further details.
- For other problems (also known as "hand-written problems"), submit the answers in an electronic copy via the git repository before the due time. Please combine the solutions of all these problems into **only ONE file** in the PDF format, with the file name in the format of "`hw[# of HW].pdf`" (e.g. "`hw1.pdf`"), all in **lowercase**; otherwise, you might only get the score of one of the files (the one that the grading TA chooses) or receive penalty because of incorrect filename.
- Discussions with others are strongly encouraged. However, you should write down your solutions `in your own words`. In addition, for **each and every** problem you have to specify the references (the Internet URL you consulted with or the people you discussed with) on the first page of your solution to that problem. You may get zero point for problems with no specified references.
- **NO LATE SUBMISSION ONE DAY AFTER THE DUE TIME IS ALLOWED.** For all submissions, up to one day of delay is allowed; however, penalty will be applied to the score according to the following rule (the time will be in seconds):

  $$\text{LATE\_SCORE} = \text{ORIGINAL\_SCORE} \times (1 - \text{DELAY\_TIME}/86400)$$

  Note that late submission of partial homework is NOT allowed. The penalty will be applied to the entire homework in the case of late submission.

Problem 1

Assume that $f(x) = f(\lfloor x \rfloor)$ when $x \notin \mathbb{Z}$ and $f(x) = 1$ when $x \leq 1$.

(1) Solve the recurrences.

(a) (3%) $f(n) = 16f(\frac{n}{4}) + 514n$. Use the substitution method to prove that $f(n) = O(n^2)$.

(b) (3%) $f(n) = 27f(\frac{n}{3}) + 40e^3n^3$. Use a recursion tree to prove that $f(n) = O(n^3 \log n)$.

(2) (14%) Partition the following functions into equivalence classes such that $f(n)$ and $g(n)$ are in the same equivalence class iff $f(n) = \Theta(g(n))$.

$$\begin{array}{ccc}
e^5n^3 - 10n^2 + e^{1000} & f(n) = ef(n/2) & f(n) = f(n-1) + n^e \\
f(n) = \sqrt{n}f(\sqrt{n}) + n & f(n) = f(n-1) + \frac{1}{n} & f(n) = f(n-1) + f(n-2) \\
en \lg(\ln n) & \sum_{i=1}^{n} \frac{1}{i} & \frac{n}{\ln n} \\
n \lg n & n \ln n & n! \\
2147483647 & n^{1/\lg n} & (\sqrt{2})^{\ln n} \\
\ln n! & e^{\ln n} & \frac{10n}{e} \\
\lg \ln n & 2^{10000} & (\lg n)^{\ln n} \\
n^{3/2} & n^{\lg \lg n} & n^{\lg \ln n}
\end{array}$$

Then, arrange the equivalence classes into a sequence

$$C_1, C_2, C_3 \ldots$$

such that for any $f(n) \in C_i$ and $g(n) \in C_{i+1}$, $f(n) = o(g(n))$. (If you are unfamiliar with the asymptotic notations, please review chapter 3 of the textbook.)

You only need to give the equivalence class sequence C and each class will be considered wrong if there is any error. Do not forget to put separaters to indicate which functions are in the same equivalence class. You should briefly describe how you come to the final solution, so that you may still get partial score even if your final solution is incorrect.

Note: ln is logarithm of base e and lg is logarithm of base 2.

Hint: Some formulas are not easy to find the bound, you may ask Google!

Hint2: It might be helpful to think about the upper (or lower) bounds of functions when you arrange the functions in order.

Problem 2

(1) Given an array A of size n, we call an element a *majority element* of A if the element appears more than $\lfloor n/2 \rfloor$ times in A. Please design an algorithm to find the *majority element* if it exists; your algorithm should also correctly return `NONE` if the majority element does not exist. Please do not use `sort` to solve this problem.

(a) (5%) Design an algorithm that finds the majority element in $O(n \log n)$ time.
Hint: Split the array A into two arrays A_1 and A_2 of half size. Can you find the majority element of A from that of A_1 and A_2?

(b) (5%) Design an algorithm that finds the majority element in $O(n)$ time.
Hint: Pair up the elements in A to get $n/2$ pairs.

(2) Given k sorted arrays of size n, we want to merge them into a single sorted array of size kn.

(a) (5%) Consider the following algorithm called `SequentialMerge`: First, merge the first two arrays, then merge in the third, then merge in the fourth, and so on. What is the time complexity of this algorithm in terms of k and n? Please briefly explain your answer.

(b) (5%) Design a more efficient algorithm than `SequentialMerge` by using divide and conquer, and show the time complexity of your algorithm in terms of k and n.

Problem 3

Given N black points and N white points on a two dimensional Cartesian plane, with **no two points coincide at the same location** and **no three points on the same line**. Let S be the set of all these $2N$ points.

We can pair each black point with a white point, and each point belongs to exactly one pair, resulting in N pairs. This is called a *match.*

For a match, we can connect each pair of points with a line segment. If some of the line segments intersect, the match is called a *miserable-word match.* Otherwise, it is called a *good-word match.*

(1) (8%) Prove that good-word match always exists.

Hint: For points A, B, C, D, if AB intersects CD, then segment length $AB + CD > AC + BD$. Can the match with minimum total segment length be a miserable-word match?

(2) (15%) For $N \geq 2$, design an algorithm to find a line L in $O(N \log N)$ time, such that

- L passes through exactly one point of S, say x.
- At least one side of L have **the number of black points equal to the number of white points** (not including x).
- Both sides of L should be **nonempty** (not including x).

(Here ”sides” means half-planes divided by L.)

Hint 1: Choose an arbitrary center point x and a line L passing through it. Select one side of L to observe the difference $d = \#black - \#white$. Slowly rotate L by 180 degree. How does d change during the rotation? Can you find some angle with $d = 0$?

Hint 2: Rotate L by another 180 degree. There should be at least two “different” angle with $d = 0$. Can they both violate the third constraint?

(3) (7%) Design an algorithm to find a good-word match in $O(N^2 \log N)$ time.

Hint: Use the result of (2) to divide and conquer.

P.S. If you submit a different algorithm that doesn't use the result of (2), but **correctly** solves subtask (3), you can also get full score for subtasks **(2)** and (3).

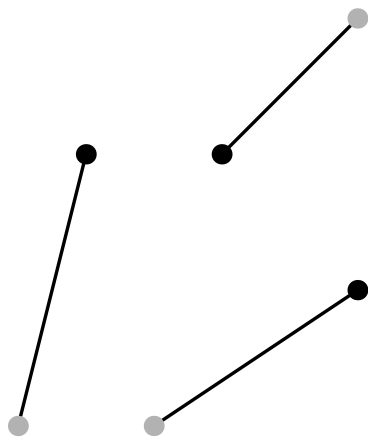
Figure 1: *Good-word* match.

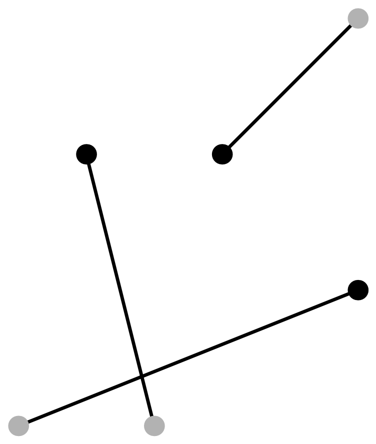
Figure 2: *Miserable-word* match.

Problem 4 - Queue (Programming)

Description

(30% + 6% bonus) In computer science, a queue is a particular kind of abstract data type or collection in which the entities in the collection are kept in order and the principal (or only) operations on the collection are the addition of entities to the rear terminal position, known as enqueue, and removal of entities from the front terminal position, known as dequeue.

We all should have learned how queues work before. But today we are considering a queue in the real world instead of a queue in computer science. There are n people queueing up for the tickets of the ADA course, numbered from 1 to n respectively. Since the ADA course is very popular, some of them may even fight for a better place in the queue.

The fights in the ADA course is very special. When person i is fighting against person j, person i wins if and only if $c(i-j)(i+j)^e \bmod p > \frac{p}{2}$, where c, e, p are numbers predefined by TAs. We say a queue is stable if the first person in the queue can beat the second person in the queue, the second person can beat the third, and so on. Please help us arrange a stable queue.

Input Format

The first line contains an integer T indicating the total number of test cases. Each test case contains four integers n, c, e, p in one line.

- $1 \leq T \leq 10$
- $2 \leq n \leq 200000$
- $1 \leq c < p$
- $0 \leq e \leq 10^{18}$
- $2n < p \leq 2 \times 10^9$
- p is a prime.
- There is at least one stable arrangement.

Output Format

For each test case, please output n integers which form a stable queue from first to last in one line.

Sample Input

```
3
2 1 0 5
3 5 8 13
8 5 3 37
```

Sample Output

```
1 2
1 2 3
7 3 8 6 2 5 4 1
```

Hint

- In the first sample, person 1 beats person 2 since $1(1-2)(1+2)^0 = -1$, and $-1 \bmod 5 = 4 > \frac{5}{2}$.
- There are two bonus tests (3 points each) in this problem. The condition $n \leq 100000$ holds for the first 10 tests.