ALGORITHM

Fair and Square
Google Code Jam 2013
Qualification Round C

登場人物

98701020 陳聖惟
98701031 呂柏頤
98703011 許宗仁
99701043 劉映君

Why Fair and Square?

- Introduction of Google Code Jam
- Simple Mathematics and suit for expression our idea

Define of Fair and Square

- Palindromic Numbers
  - e.g 121, 9, 67776, 10000000001
- Square Numbers : 9, 16, 225
- Fair and Square Numbers
  - e.g 121, 484, 12321
- Palindrome Square Number $\neq$ Fair and Square
  - e.g 676, 698896

Solving This Problem

- Story : Little John wants to find numbers of all ‘Fair and Square numbers’ of interval [A,B] for T cases.
- Input : Cases T, Interval [A,B].
- Find all ‘Fair and Square Numbers’ in [A,B]
- Output : Case #x: y

Google’s Question

- Small Dataset
  - $1 \leq T \leq 100$, $1 \leq A \leq B \leq 1000$ ... seems easy !?
- First Large Dataset
  - $1 \leq T \leq 10000$, $1 \leq A \leq B \leq 10^{14}$
    ...We could need powerful computer !
- Second Large Dataset
  - $1 \leq T \leq 1000$, $1 \leq A \leq B \leq 10^{100}$
    ...Wow !

Analysis for The Problem

* Question : Directly Search or Table ?
  * The problem of directly search : repeated case
  * For e.g , interval [7,65] contain [9,10]

1. If we compare $[A_i, B_i], \forall i \in T$

$\Rightarrow$ Numbers of compare time $= \sum_{i=1}^{T} C_{A_i - B_i} = T \times C_{\bar{A}-\bar{B}}$

where $C_t =$ Times of Judging Fair and Square numbers

2. Consturct table for $[\inf(A_i), \sup(B_i)], \forall i \in T$

$\Rightarrow$ Numbers of compare time $= T + (C_{\inf(A_i)-\sup(B_i)})$

e.g.
495,580
872,879
9,10
3,4
7,65

* So, we should construct table.

Some Definition

$$a = 10^N a_1 + 10^{N-1} a_2 + \cdots + a_N$$
$$= (a_1 a_2 \cdots a_N)$$
$$(a_1 \cdots a_i) \bullet (b_1 \cdots b_i) = \sum_{k=1}^{i} a_k b_k$$

Idea 1 Tricky Method

* Observe 'PalindromicSquareNumbers' from http://www.fengyuan.com/palindrome.html
* Note : we still delete some numbers which is not 'Fair and Square'.
*

[Algorithm]
1. Import Data & Save to Table
2. def judge_palindrome(i)
3. for k in Table.root :
4. if judge_palindrome (Table.root) :
5. NewTable.append(k)

Analysis for Idea 1

* The total judgement is just the number of palindromic square numbers.
* It is only based on 'Google'.
* But of course, it is assumed that we can find the table which digits up to 100... with correctness.

What is idea 1 talk ?

* It is like joke ! But give us some important view.
  * root : it only appear {0,1,2}. '3' is exception for 1-digit
* We can reducing searching space only for {0,1,2}
* Add rather than Prune, we could use the 'mirroring' of palindrome number.
  So, we could reduce searching space again.

Some Important Thm.

Thm. if a is palindromic number

and $\sum_{i=1}^{n} a_i^2 < 10 \Rightarrow a^2$ is palindromic number

proof : Before do this we need some preparation.

Lemma. Suppose a is palindromic

If $(a_i .. a_1) \bullet (a_1 .. a_i) < 10, \forall i \leq n$

$\Leftrightarrow (a_n .. a_1) \bullet (a_1 .. a_n) < 10$

proof. ($\Rightarrow$) is trivial.

$$(\Leftarrow)\ (a_i .. a_1) \bullet (a_1 .. a_i) < a_i^2 + ... + a_1^2$$
$$< a_1^2 + ... + a_n^2$$
$$< (a_n .. a_1) \bullet (a_1 .. a_n) \because a \text{ is palindromic}$$
$$< 10$$

Conti. Thm.

Proof of *Thm*:

By *Lemma* ∵ The half of the front digit <10 ∴ Not carry.

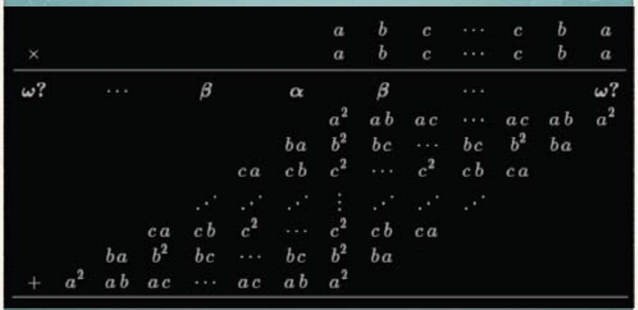
⇒ The half of the later digit do not carry.

For the ith-digit of $a^2 = \begin{cases} (a_i..a_1)\bullet(a_1..a_i),\ i \le n \\ (a_{2n-i-1}..a_1)\bullet(a_1..a_{2n-i-1}),\ n < i \le 2n-1 \end{cases}$

⇒ a^2 is the palindromic number.

Cor.1 In particular, if root $a \neq 3 \Rightarrow a_i \in \{0,1,2\}$

Cor.2 $\sum_{i=1}^{\left\lceil \frac{n}{2} \right\rceil} a_i^2 < 5$



Expression of Thm.

Idea 2 Brute-Force

- This is the worst method.
- Palindrome Numbers :
  $(a_1 a_2 \cdots a_N) = (a_N a_{N-1} \cdots a_1)$
- Squares Numbers :
  $(int\sqrt{(a_1 a_2 \cdots a_N)})^2 = (a_1 a_2 \cdots a_N)$

[Algorithm]

1. def judge_palindrome(i)
2. Table=[python list]
3. for k in range(1,10^N):
4. d=int($k^{0.5}$)
5. if d^2== k
   and judge_palindrome(k)
   and judge_palindrome(d) :
6. then Table.append(k)

Analysis for Idea 2

- Original : from 1 to 10^N , it needs to judge 10^N
  ...which sounds terrible !
- Improvement ? Searching space reduced !
  - Just judge the root of the fair-and-square number.
  - We only need to calculate from $1 \sim 10^{\frac{N}{2}}$.
    => $10^{\left\lceil \frac{N}{2} \right\rceil}$

Some Definition

Note. n is half of whole digit N which $n = \left\lceil \frac{N}{2} \right\rceil$

(∵ We only find the root of the Fair and Square numbers.)

(The reason of '$\left\lceil \frac{N}{2} \right\rceil$' is that the square of roots could not carry.)

Idea 3 Thm. Application

[Algorithm]

1. def Judge_palindrome(i)
2. def mirroing_even
3. def mirroing_odd
4. Table=[]
5. for every digit of k in {0,1,2} :
6. Table.append(mirroing_even(k))
7. Table.append(mirroing_odd(k))
8. If $\sum_{i=1}^{n} a_i^2 < 10$ then hold Table[k]

Analysis for Idea 3

Given front of half digits of $a = (a_1 a_2 \cdots a_{\lceil \frac{n}{2} \rceil - 1} a_{\lceil \frac{n}{2} \rceil})$

$\Rightarrow$ The number of judging $= 3^{\lceil \frac{n}{2} \rceil}$

Improvement of Idea 3

- Now, where can we improve our algorithm ?
  The exactly answer is 'combination' !
- 'Combination' is the famous technique in Discrete Mathematics.

We can put k numbers of $\{0,1,2\}$ into n digits with combination method.
With this approach, we can give the idea 4 !

Practice of Idea 4

- Based on idea 3, we only search the root of whole digit and half front of the digit of root.
- There are some implement of combinations :
  1. if $n == 2k+1$ (which means number of digit is odd):
  2. then discuss putting
  $\{1,1,2\}$, $\{2,1\}$,$\{1,2\}$, $\{2\}$. $\{1,1,1,1,1\}$, $\{1,1,1,1\}$, $\{1,1,1\}$, $\{1,1\}$, $\{1\}$ into digit position.
  3. else : discuss putting
  $\{2\}$, $\{1,1,1,1\}$, $\{1,1,1\}$, $\{1,1\}$, $\{1\}$ into digit position.

Analysis for Idea 4

Denote $\#\{a,b,\cdots,m\}$ is counting numbers of the combination with the front of the Fair and Square numbers which appear one a, b,⋯, m.

$\Rightarrow$ 1. For $n \in$ odd :

$\#\{1,1,2\} = C_1^{\lceil \frac{n}{2} \rceil - 2}$; $\#\{2,1\} = 1$; $\#\{1,2\} = 1$; $\#\{2\} = 1$; $\#\{1,1,1,1,1\} = C_3^{\lceil \frac{n}{2} \rceil - 2}$

$\#\{1,1,1,1\} = C_3^{\lceil \frac{n}{2} \rceil - 1}$; $\#\{1,1,1\} = C_2^{\lceil \frac{n}{2} \rceil - 1}$; $\#\{1,1\} = C_1^{\lceil \frac{n}{2} \rceil - 1}$; $\#\{1\} = 1$

2. For $n \in$ even :

$\#\{2\} = 1$; $\#\{1,1,1,1\} = C_3^{\frac{n}{2}-1}$; $\#\{1,1,1\} = C_2^{\frac{n}{2}-1}$; $\#\{1,1\} = C_1^{\frac{n}{2}-1}$; $\#\{1\} = 1$

Analysis for Idea 4

Summary. We just calculate

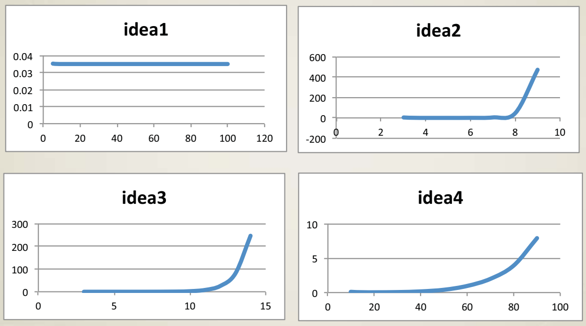
$$\sum_{i=1}^{n} \left\{ C_1^{\lceil \frac{i}{2} \rceil - 2} + C_3^{\lceil \frac{i}{2} \rceil - 2} + 2 \times [C_3^{\lceil \frac{i}{2} \rceil - 1} + C_2^{\lceil \frac{i}{2} \rceil - 1} + C_1^{\lceil \frac{i}{2} \rceil - 1}] \right\} \text{ times,}$$

which exactly $< 3^{\lceil \frac{n}{2} \rceil}$ as $n \to \infty$. $(\because O(n^4) < O(3^{\frac{n}{2}}))$

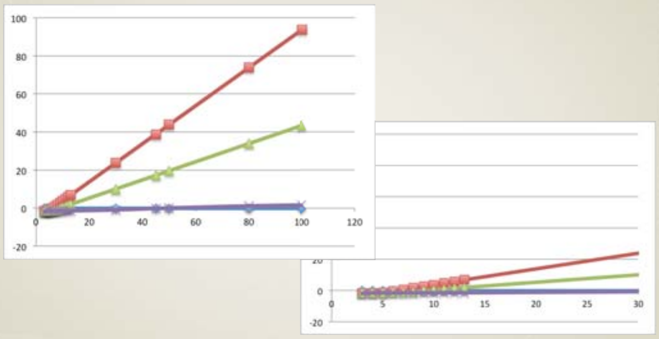
Experiment

idea	1	2	3	4
upper bound digit	45	8	12	100
Time(sec)	0.035	43.177	25.611	11.826

Experiment



Experiment



blue (Idea1), red(Idea2), green(Idea3), purple(Idea4)

Summary and Inspiration

- What is 'Big Data' ? Why so popular ?
- Data Structure and Algorithm is important to complete program.
- The main idea comes from 'Discrete Mathematics', go through Math again, you could get more idea.