Competitive Programming and Big-O Notation

Anson Ho

Flow

1. Introduction to competitive programming

2. Skills and advices for competitive programming

3. Big-O Notation

Programming

- Algorithm
  - idea, outline
  - flowchart
- Program
  - codes in specific programming languages
- Target
  - problem solving

Competitive Programming

- Usually for competitions
- More limits on
  - runtime
  - memory
  - resources
    - time
    - software
    - hardware
  - available programming languages

Online judge

- HKOI online judge
- Codeforces
- Codechef
- TopCoder
- POJ
- HDU online judge
- AtCoder

- May have contests regularly

Contest

- Annual contest
  - IOI
  - NOI
  - APIO
  - ACM ICPC
  - CCC
- Open contest
  - Facebook Hacker Cup
  - Google Code Jam

Contest structure

- Individual/ team
- Length
- Full feedback?
- Pretest?
- Partial score (subtask)?
- Batch score/ score per test
- Time penalty?
- Hack?

Programming languages

- Common
  - C, C++, Java, Pascal, Python, …
- HKOI (16/17)
  - C, C++, Pascal

Programming languages

- IOI 16
  - C++, Pascal, Java
  - C and Pascal will be removed
  - Python will be added
- NOI 16
  - C, C++, Pascal
  - C and Pascal will be removed in 2020

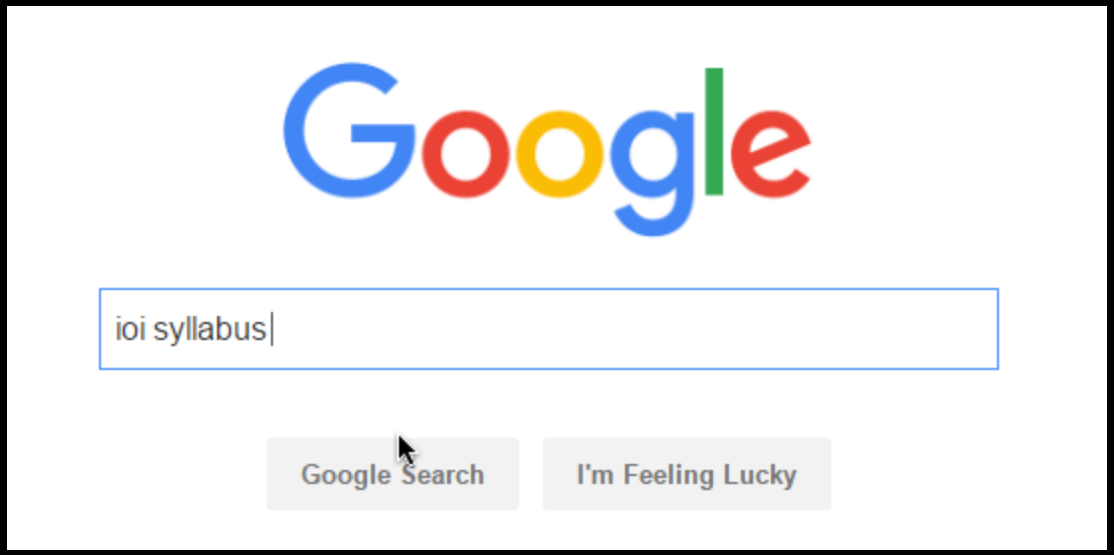
Environment

- Onsite/ online
- Operating system
- Compiler
- IDE/ text editor

- Beware of the differences between the computer for coding and the computer for judging

Topics

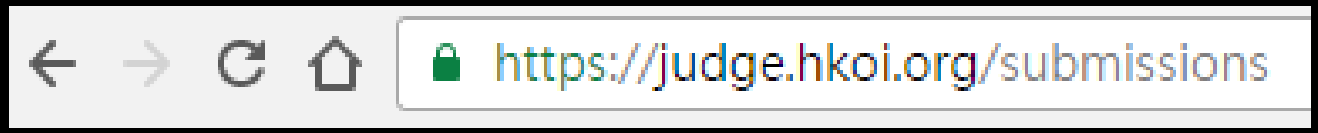
- Basic



Problem type

- Input and output
- Interactive
- Output only
- (Other)

Common verdict (HKOI)



Verdict (HKOI)

- Accepted
- Compilation Error
- Wrong Answer
- Time Limit Exceeded
- Runtime Error
- Partial Score

Verdict (other)

- Wrong Output Format
- Presentation Error
- Memory Limit Exceeded
- Idleness Limit Exceeded
- Skipped
- Hacked

Take a break

Before the start of the contest

- The information about the contest
  - Will the problems arrange in the order of difficulties?
  - What types of problems will be most likely to appear?
- What is your target?
- Get familiar with your coding environment

When the contest starts

- Briefly read all of the problems first
  - if “first blood” is not your target
- Observe the scoreboard for “hints”
  - if exists

When the contest starts

- Try to focus on one problem
  - “multithread” is hard for human
    - at least it is hard for me
- If … then break
  - accept that some problems cannot be solved in a short period of time

Read a problem

- Extract the useful information
  - although background stories are interesting
- Constraints, sample tests are obviously useful
  - understand the sample tests before coding

Read a problem

- Ask for clarification if necessary
  - the range of a variable is missing
  - ambiguity in the statement
- Think before ask
  - don’t ask what is the corresponding output for …
  - in most of the time, the reply is no comment

Before submit

- Check if your program works on all sample tests
  - create some tests if you think the sample tests are not enough
  - esp. corner cases

Before submit

- Check if the program has silly bugs
  - presence of debug messages
  - insufficient array size
  - forget that array index is 0-based
  - some steps may cause overflow
  - absence of necessary initialization
  - wrong output format
    - esp. when there is no auto-formatting
  - standard IO/ file IO?

Before submit

- For contest without feedback, write a slow but correct program to verify your main solution with some small cases (automatically)

Example

- Count the number of factors of N
- $1 <= N <= 10^{12}$

Example

- Main solution

- Prime factorization of N
- $N = p1^{k1}p2^{k2}\ldots pm^{km}$

- $Ans = (k1 + 1)(k2 + 1)\ldots(km + 1)$

Example

- TLE but correct solution

- For i from 1 to N

```
    if N is divisible by i
        Ans += 1
```

Debug

- Usage of IDE
  - breakpoint
  - variable watcher
- Debug message
- “Binary search” on bugs

Example

- Program exited unexpectedly
- 100 lines
- insert a debug message at line 50

```
if the debug message is printed
   the program exits in the second half
else
   the program exits in the first half
```

Advices

- Maintain good coding style
  - reader-friendly spacing
  - avoid long expression
  - use short variable name
    - e.g. cnt instead of count
    - avoid meaningless names for some meaningful variables
  - decrease the time cost on debugging
  - esp. for team contest

Advices

- No need to write comment
  - except you have something to remind yourself
  - e.g. TODO list
- Avoid unnecessary floating point
  - e.g. store fraction by 2 integers

Advices

- Practice makes perfect
  - HKOI online judge
  - Codeforces
  - …
  - solve problems in problem archive
  - participate in contests
- Attend HKOI training

Advices

- Read Codeforces blogs
- Read others’ code
- Don’t google solution/ view editorial immediately

Take a break

Time complexity

- A measure of the runtime (number of operations) of the program

- Highly depends on the algorithm
- Also depends on the implementation

- vs time limit

Time complexity

- Focus
  - average case
  - worst case

Space complexity

- A measure of memory usage of the program

- Highly depends on the algorithm
- Also depends on the implementation

- vs memory limit
- (usually not a limiting factor)

Big-O notation

- Briefly estimate the complexities of algorithm
- Shortcut to avoid TLE
- Save your valuable time

Big-O notation

- A “upper bound” of the function

- f(n) = O(g(n))
  - if there exists **constants** n_0 , c
  - such that for all n > n_0
  - we have f(n) <= c * g(n)

Big-O notation

- We also have other kinds of “bound”

Notation	Name	Intuition	Informal definition: for sufficiently large n...	Formal Definition
$f(n) = O(g(n))$	Big O; Big Oh; Big Omicron[12]	$\|f\|$ is bounded above by g (up to constant factor) asymptotically	$\|f(n)\| \leq k \cdot g(n)$ for some positive k	$\exists k > 0\ \exists n_0\ \forall n > n_0\ \|f(n)\| \leq k \cdot g(n)$
$f(n) = \Omega(g(n))$	Big Omega	**Two definitions :** Number theory: $\|f\|$ is not dominated by g asymptotically Complexity theory: f is bounded below by g asymptotically	Number theory: $\|f(n)\| \geq k \cdot g(n)$ for infinitely many values of n and for some positive k Complexity theory: $f(n) \geq k \cdot g(n)$ for some positive k	Number theory: $\exists k > 0\ \forall n_0\ \exists n > n_0\ \|f(n)\| \geq k \cdot g(n)$ Complexity theory: $\exists k > 0\ \exists n_0\ \forall n > n_0\ f(n) \geq k \cdot g(n)$
$f(n) = \Theta(g(n))$	Big Theta	f is bounded both above and below by g asymptotically	$k_1 \cdot g(n) \leq f(n) \leq k_2 \cdot g(n)$ for some positive k_1, k_2	$\exists k_1 > 0\ \exists k_2 > 0\ \exists n_0\ \forall n > n_0$ $k_1 \cdot g(n) \leq f(n) \leq k_2 \cdot g(n)$
$f(n) = o(g(n))$	Small O; Small Oh	f is dominated by g asymptotically	$\|f(n)\| \leq k \cdot \|g(n)\|$, for every fixed positive number k	$\forall k > 0\ \exists n_0\ \forall n > n_0\ \|f(n)\| \leq k \cdot \|g(n)\|$
$f(n) = \omega(g(n))$	Small Omega	f dominates g asymptotically	$\|f(n)\| \geq k \cdot \|g(n)\|$, for every fixed positive number k	$\forall k > 0\ \exists n_0\ \forall n > n_0\ \|f(n)\| \geq k \cdot \|g(n)\|$
$f(n) \sim g(n)$	On the order of	f is equal to g asymptotically	$f(n)/g(n) \to 1$	$\forall \varepsilon > 0\ \exists n_0\ \forall n > n_0\ \left\|\frac{f(n)}{g(n)} - 1\right\| < \varepsilon$

https://en.wikipedia.org/wiki/Big_O_notation#Family_of_Bachmann.E2.80.93Landau_notations

Examples

- Linear search
- $O(n)$

- Bubble sort
- $O(n^2)$

Examples

- O(logn)
- Binary search

- O(nlogn)
- Merge sort

Examples

- Exhaustion of length n 01-string
- $O(n2^n)$

- Exhaustion of length n permutation
- $O(n * n!)$

Properties

O(f(n)) + O(g(n))

= O(f(n) + g(n))

= O(f(n))

if f(n) is “greater” then g(n)

O(c * f(n))

=O(f(n))

Multivariable

- $O(nmk)$
- $O(n^2m + nm^2)$

Remarks

- The relation between
  - number of operations
  - runtime of program
- The machine performance

Remarks

- $f(n) <= c * g(n)$
- $O(n)$ is always better than $O(n^2)$?
- $f_1(n) = 1000n$
- $f_2(n) = 0.5n^2$

Remarks

- Lower the c
- “Constant optimization”
  - avoid c++ map for offline query
  - use if statement instead of unnecessary mod operator

- Useless in most competitions
- Useful in some competitions

Thank you