Intro to Competitive Programming

MCPC Workshop

Outline

What's the Competitive Programming

Briefly speaking, it is to solve programming problems:

What's the Competitive Programming

Briefly speaking, it is to solve programming problems:

- Fast

What's the Competitive Programming

Briefly speaking, it is to solve programming problems:

- Fast
- Correctly

What's the Competitive Programming

Briefly speaking, it is to solve programming problems:

- Fast
- Correctly
- Elegantly

Outline

Get lowest bit

The straightforward way:

```
def lowb(x):
  p = 0
  while x:
    # if p-th digit is 1, return 2^p
    if x & 1: return 1<<p
    x >>= 1 # x = x / 2
    p += 1
  return 0
```

Get lowest bit

The straightforward way:

```
def lowb(x):
  p = 0
  while x:
    # if p-th digit is 1, return 2^p
    if x & 1: return 1<<p
    x >>= 1 # x = x / 2
    p += 1
  return 0
```

In competitive programming:

```
def lowb(x):
  return x & (-x)
```

Prime Sieve

The straightforward way:

```
def primes(n):
  p = []                    # empty prime list at beginning
  for i in range(2,n+1):    # is i a prime?
    f = False
    for j in range(2, i):   # any divisor in [2, i-1]?
      if i % j == 0:
        f = True
        break
    if not f:               # no divisor
      p.append(i)           # yeah, it's a prime!
  return p
```

Prime Sieve

The straightforward way:

```
def primes(n):
  p = []                    # empty prime list at beginning
  for i in range(2,n+1):    # is i a prime?
    f = False
    for j in range(2, i):   # any divisor in [2, i-1]?
      if i % j == 0:
        f = True
        break
    if not f:               # no divisor
      p.append(i)           # yeah, it's a prime!
  return p
```

$2 + 3 + 2 + 5 + \ldots \approx O(\frac{n^2}{logn})$ (https://oeis.org/A088821)

Prime Sieve

More efficient way:

```
def primes(n):
  f = [0] * (n+1)           # set f[0]=0, f[1]=0, ... f[n]=0
  p = []                    # empty prime list at beginning
  for i in range(2, n+1):   # is i a prime?
    if not f[i]:            # not sieved by any value
      p.append(i)           # yeah, it's a prime!

    j = 2                   # sieve:
    while j*i <= n:         # 2*i,
      f[j*i] = True         # 3*i,
      j += 1                # ...
  return p
```

Prime Sieve

More efficient way:

```
def primes(n):
  f = [0] * (n+1)            # set f[0]=0, f[1]=0, ... f[n]=0
  p = []                     # empty prime list at beginning
  for i in range(2, n+1):    # is i a prime?
    if not f[i]:             # not sieved by any value
      p.append(i)            # yeah, it's a prime!

    j = 2                    # sieve:
    while j*i <= n:          # 2*i,
      f[j*i] = True          # 3*i,
      j += 1                 # ...
  return p
```

$\frac{n}{2} + \frac{n}{3} + \ldots + 1 \approx O(nlogn)$ (Harmonic sequence)

Prime Sieve

In competitive programming:

```
def primes(n):
  f = [0] * (n+1)             # set f[0]=0, f[1]=0, ... f[n]=0
  p = []                      # empty prime list at beginning
  for i in range(2, n+1):     # is i a prime?
    if not f[i]:              # not sieved by any value
      p.append(i)             # yeah, it's a prime!

    for j in p:               # let j be a known prime
      if j * i > n: break     # reach the upper bound
      f[j * i] = True         # sieve j * i
      if i % j == 0: break    # guarantee j is the minimum divisor
  return p
```

Prime Sieve

In competitive programming:

```python
def primes(n):
  f = [0] * (n+1)              # set f[0]=0, f[1]=0, ... f[n]=0
  p = []                       # empty prime list at beginning
  for i in range(2, n+1):      # is i a prime?
    if not f[i]:               # not sieved by any value
      p.append(i)              # yeah, it's a prime!

    for j in p:                # let j be a known prime
      if j * i > n: break      # reach the upper bound
      f[j * i] = True          # sieve j * i
      if i % j == 0: break     # guarantee j is the minimum divisor
  return p
```

Each number is only sieved by it's minimum divisor once.

Prime Sieve

In competitive programming:

```
def primes(n):
  f = [0] * (n+1)           # set f[0]=0, f[1]=0, ... f[n]=0
  p = []                    # empty prime list at beginning
  for i in range(2, n+1):   # is i a prime?
    if not f[i]:            # not sieved by any value
      p.append(i)           # yeah, it's a prime!

    for j in p:             # let j be a known prime
      if j * i > n: break   # reach the upper bound
      f[j * i] = True       # sieve j * i
      if i % j == 0: break  # guarantee j is the minimum divisor
  return p
```

Each number is only sieved by it's minimum divisor once.
It's linear!

$A + B$

The straightforward way:

```python
def func(a, b):
  return a + b
```

$A + B$

The straightforward way:

```python
def func(a, b):
  return a + b
```

In competitive programming:

```python
from datetime import datetime
from time import sleep

def func(a, b):
  s = datetime.now()
  sleep(a)
  sleep(b)
  e = datetime.now()
  return e.second - s.second
```

$A + B$

The straightforward way:

```python
def func(a, b):
  return a + b
```

In competitive programming:

```python
from datetime import datetime
from time import sleep

def func(a, b):
  s = datetime.now()
  sleep(a)
  sleep(b)
  e = datetime.now()
  return e.second - s.second
```

~~Creative!~~

Outline

Why

Why do we do Competitive Programming?

An Imagination

- You want to be a top-class programmer 😎.

An Imagination

- You want to be a top-class programmer 😎.
- There are lots of choices: 😱

An Imagination

- You want to be a top-class programmer 😎.
- There are lots of choices: 😱
  - Web full stack, Mobile dev, Database, Big data, Machine learning . . .

An Imagination

- You want to be a top-class programmer 😎.
- There are lots of choices: 😱
  - Web full stack, Mobile dev, Database, Big data, Machine learning . . .
- Oh, you choosed *Web full stack* 😃.

An Imagination

- You want to be a top-class programmer 😎.
- There are lots of choices: 😱
  - Web full stack, Mobile dev, Database, Big data, Machine learning . . .
- Oh, you choosed *Web full stack* 😃.
- What is going to happend next...? 😯

An Imagination

You...

An Imagination

You...

- may find a nice online resource. 😀

An Imagination

You...

- may find a nice online resource. 😀
- follow the instructions. 😀

An Imagination

You...

- may find a nice online resource. 😀
- follow the instructions. 😀
- may need hours to set up environment. 🕖🕘🕒

An Imagination

You...

- may find a nice online resource. 😀
- follow the instructions. 😀
- may need hours to set up environment. 🕖🕗🕘
- finally finish your first demo before sleep. 😐

An Imagination

You...

- may find a nice online resource. 😀
- follow the instructions. 😀
- may need hours to set up environment. 🕖🕗🕘
- finally finish your first demo before sleep. 😐

but what can you still remember in the next day, or next week? 😳

Reflection

Reflection

- We are distracted by those *working skill*s,

Reflection

- We are distracted by those *working skill*s,
- It's not too late to pick up these in future career (≥ 30 years),

Reflection

- We are distracted by those *working skill*s,
- It's not too late to pick up these in future career ($\geq$ 30 years),
- but we only have two to four years in university.

Reflection

- We are distracted by those *working skill*s,
- It's not too late to pick up these in future career ($\geq$ 30 years),
- but we only have two to four years in university.
- Looking for a more efficient way?

Why

Compeititve programming is most efficient way to:

Why

Compeititve programming is most efficient way to:

- improve coding skill

Why

Compeititve programming is most efficient way to:

- improve coding skill
- improve problem solving skill

Why

Compeititve programming is most efficient way to:

- improve coding skill
- improve problem solving skill
- develop insight in computer science

Why

Compeititve programming is most efficient way to:

- improve coding skill
- improve problem solving skill
- develop insight in computer science

There is another story...

Another Story

- You want to be a top-class programmer.😎

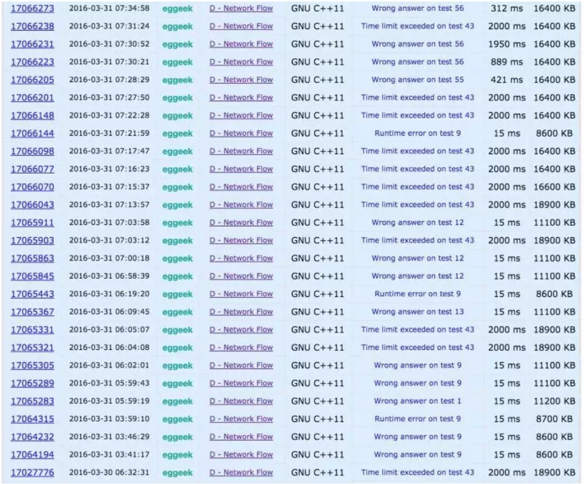
Another Story

- You want to be a top-class programmer.😎
- Someone suggests you to do Competitive Programming.😉

Another Story

- You want to be a top-class programmer.😎
- Someone suggests you to do Competitive Programming.😉
- What is going to happend next...? 😯

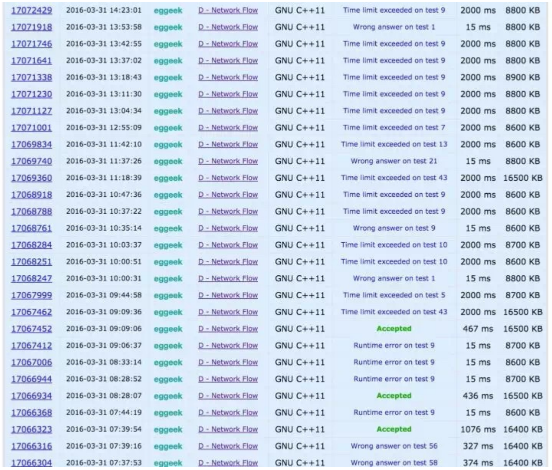
Another Story



17066273	2016-03-31 07:34:58	eggeek	D - Network Flow	GNU C++11	Wrong answer on test 56	312 ms	16400 KB
17066238	2016-03-31 07:31:24	eggeek	D - Network Flow	GNU C++11	Time limit exceeded on test 43	2000 ms	16400 KB
17066231	2016-03-31 07:30:52	eggeek	D - Network Flow	GNU C++11	Wrong answer on test 56	1950 ms	16400 KB
17066223	2016-03-31 07:30:21	eggeek	D - Network Flow	GNU C++11	Wrong answer on test 56	889 ms	16400 KB
17066205	2016-03-31 07:28:29	eggeek	D - Network Flow	GNU C++11	Wrong answer on test 55	421 ms	16400 KB
17066201	2016-03-31 07:27:50	eggeek	D - Network Flow	GNU C++11	Time limit exceeded on test 43	2000 ms	16400 KB
17066148	2016-03-31 07:22:28	eggeek	D - Network Flow	GNU C++11	Time limit exceeded on test 43	2000 ms	16400 KB
17066144	2016-03-31 07:21:59	eggeek	D - Network Flow	GNU C++11	Runtime error on test 9	15 ms	8600 KB
17066098	2016-03-31 07:17:47	eggeek	D - Network Flow	GNU C++11	Time limit exceeded on test 43	2000 ms	16400 KB
17066077	2016-03-31 07:16:23	eggeek	D - Network Flow	GNU C++11	Time limit exceeded on test 43	2000 ms	16400 KB
17066070	2016-03-31 07:15:37	eggeek	D - Network Flow	GNU C++11	Time limit exceeded on test 43	2000 ms	16600 KB
17066043	2016-03-31 07:13:57	eggeek	D - Network Flow	GNU C++11	Time limit exceeded on test 43	2000 ms	18900 KB
17065911	2016-03-31 07:03:58	eggeek	D - Network Flow	GNU C++11	Wrong answer on test 12	15 ms	11100 KB
17065903	2016-03-31 07:03:12	eggeek	D - Network Flow	GNU C++11	Time limit exceeded on test 43	2000 ms	18900 KB
17065863	2016-03-31 07:00:18	eggeek	D - Network Flow	GNU C++11	Wrong answer on test 12	15 ms	11100 KB
17065845	2016-03-31 06:58:39	eggeek	D - Network Flow	GNU C++11	Wrong answer on test 12	15 ms	11100 KB
17065443	2016-03-31 06:19:20	eggeek	D - Network Flow	GNU C++11	Runtime error on test 9	15 ms	8600 KB
17065367	2016-03-31 06:09:45	eggeek	D - Network Flow	GNU C++11	Wrong answer on test 13	15 ms	11100 KB
17065331	2016-03-31 06:05:07	eggeek	D - Network Flow	GNU C++11	Time limit exceeded on test 43	2000 ms	18900 KB
17065321	2016-03-31 06:04:08	eggeek	D - Network Flow	GNU C++11	Time limit exceeded on test 43	2000 ms	18900 KB
17065305	2016-03-31 06:02:01	eggeek	D - Network Flow	GNU C++11	Wrong answer on test 9	15 ms	11100 KB
17065289	2016-03-31 05:59:43	eggeek	D - Network Flow	GNU C++11	Wrong answer on test 9	15 ms	11100 KB
17065283	2016-03-31 05:59:19	eggeek	D - Network Flow	GNU C++11	Wrong answer on test 1	15 ms	11200 KB
17064315	2016-03-31 03:59:10	eggeek	D - Network Flow	GNU C++11	Runtime error on test 9	15 ms	8700 KB
17064232	2016-03-31 03:46:29	eggeek	D - Network Flow	GNU C++11	Wrong answer on test 9	15 ms	8600 KB
17064194	2016-03-31 03:41:17	eggeek	D - Network Flow	GNU C++11	Wrong answer on test 9	15 ms	8600 KB
17027776	2016-03-30 06:32:31	eggeek	D - Network Flow	GNU C++11	Time limit exceeded on test 43	2000 ms	18900 KB

Practice!

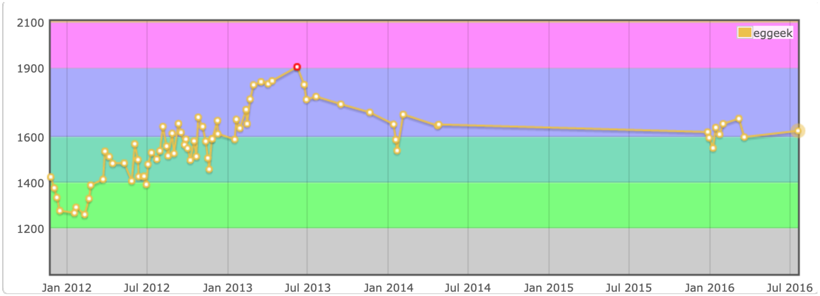
Another Story



17072429	2016-03-31 14:23:01	eggeek	D - Network Flow	GNU C++11	Time limit exceeded on test 9	2000 ms	8800 KB
17071918	2016-03-31 13:53:58	eggeek	D - Network Flow	GNU C++11	Wrong answer on test 1	15 ms	8800 KB
17071746	2016-03-31 13:42:55	eggeek	D - Network Flow	GNU C++11	Time limit exceeded on test 9	2000 ms	8800 KB
17071641	2016-03-31 13:37:02	eggeek	D - Network Flow	GNU C++11	Time limit exceeded on test 9	2000 ms	8800 KB
17071338	2016-03-31 13:18:43	eggeek	D - Network Flow	GNU C++11	Time limit exceeded on test 9	2000 ms	8900 KB
17071230	2016-03-31 13:11:30	eggeek	D - Network Flow	GNU C++11	Time limit exceeded on test 9	2000 ms	8800 KB
17071127	2016-03-31 13:04:34	eggeek	D - Network Flow	GNU C++11	Time limit exceeded on test 9	2000 ms	8800 KB
17071001	2016-03-31 12:55:09	eggeek	D - Network Flow	GNU C++11	Time limit exceeded on test 7	2000 ms	8600 KB
17069834	2016-03-31 11:42:10	eggeek	D - Network Flow	GNU C++11	Time limit exceeded on test 13	2000 ms	8600 KB
17069740	2016-03-31 11:37:26	eggeek	D - Network Flow	GNU C++11	Wrong answer on test 21	15 ms	8800 KB
17069360	2016-03-31 11:18:39	eggeek	D - Network Flow	GNU C++11	Time limit exceeded on test 43	2000 ms	16500 KB
17068918	2016-03-31 10:47:36	eggeek	D - Network Flow	GNU C++11	Time limit exceeded on test 9	2000 ms	8600 KB
17068788	2016-03-31 10:37:22	eggeek	D - Network Flow	GNU C++11	Time limit exceeded on test 9	2000 ms	8600 KB
17068761	2016-03-31 10:35:14	eggeek	D - Network Flow	GNU C++11	Wrong answer on test 9	15 ms	8600 KB
17068284	2016-03-31 10:03:37	eggeek	D - Network Flow	GNU C++11	Time limit exceeded on test 10	2000 ms	8700 KB
17068251	2016-03-31 10:00:51	eggeek	D - Network Flow	GNU C++11	Time limit exceeded on test 10	2000 ms	8600 KB
17068247	2016-03-31 10:00:31	eggeek	D - Network Flow	GNU C++11	Wrong answer on test 1	15 ms	8800 KB
17067999	2016-03-31 09:44:58	eggeek	D - Network Flow	GNU C++11	Time limit exceeded on test 5	2000 ms	8700 KB
17067462	2016-03-31 09:09:36	eggeek	D - Network Flow	GNU C++11	Time limit exceeded on test 43	2000 ms	16500 KB
17067452	2016-03-31 09:09:06	eggeek	D - Network Flow	GNU C++11	Accepted	467 ms	16500 KB
17067412	2016-03-31 09:06:37	eggeek	D - Network Flow	GNU C++11	Runtime error on test 9	15 ms	8700 KB
17067006	2016-03-31 08:33:14	eggeek	D - Network Flow	GNU C++11	Runtime error on test 9	15 ms	8600 KB
17066944	2016-03-31 08:28:52	eggeek	D - Network Flow	GNU C++11	Runtime error on test 9	15 ms	8700 KB
17066934	2016-03-31 08:28:07	eggeek	D - Network Flow	GNU C++11	Accepted	436 ms	16500 KB
17066368	2016-03-31 07:44:19	eggeek	D - Network Flow	GNU C++11	Runtime error on test 9	15 ms	8600 KB
17066323	2016-03-31 07:39:54	eggeek	D - Network Flow	GNU C++11	Accepted	1076 ms	16400 KB
17066316	2016-03-31 07:39:16	eggeek	D - Network Flow	GNU C++11	Wrong answer on test 56	327 ms	16400 KB
17066304	2016-03-31 07:37:53	eggeek	D - Network Flow	GNU C++11	Wrong answer on test 58	374 ms	16400 KB

Keep practicing!

Another Story



Win and lose...

Another Story

Get rewarded!

Another Story

Make friends!

Why do we do Competitive Programming?

Why do we do Competitive Programming?

- IT IS FUN!

Why do we do Competitive Programming?

- IT IS FUN!
- Employment

Why do we do Competitive Programming?

- IT IS FUN!
- Employment
- Academia

Outline

How to start?

- USACO: `https://train.usaco.org/usacogate`
- Codeforces: `http://codeforces.com`
- Atcoder: `http://atcoder.jp`
- Google contests: `https://codingcompetitions.withgoogle.com`
- Facebook Hacker Cup: `https://www.facebook.com/hackercup/contest`

How to join us?

- Weekly training on Saturday.
  - Time: 12:00 to 17:00
  - Location: Lab 147, Rainforest walk 14
- Facebook: `https://www.facebook.com/groups/454114112027992/`
- Mailing list: `https://groups.google.com/forum/#!forum/monashicpc/join`

What will we do?

- Monash Collegiate Programming Contest (MCPC) on **24th August**.
- New Zealand Programming Contest (NZPC) on **7th September**.
- International Collegiate Programming Contest (ICPC) Regional Divisional (TBD).
- ICPC Regional Final (TBD).
- ICPC World Final (TBD, if only...).

Outline

End

Join Us!

Thank you!