Solving Google Code Jam problems with PyPy

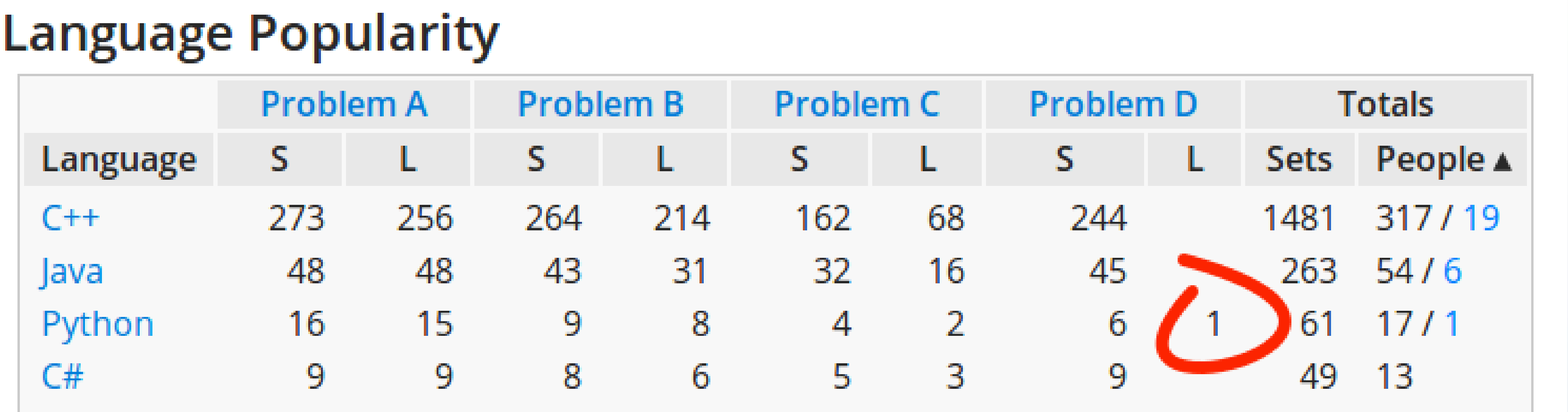
Alessandro Amici <a.amici@bopen.eu>

B-Open Solutions – http://bopen.eu

EuroPython 2013

Python wins competitions already!

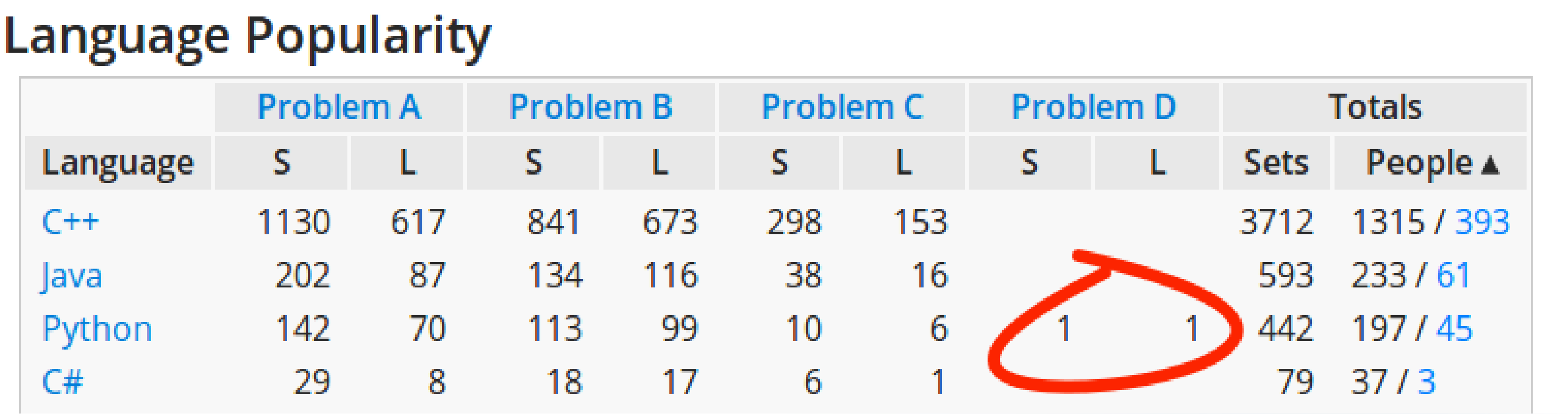
Google Code Jam 2011 – Round 3 – user: linguo



Language Popularity

	Problem A		Problem B		Problem C		Problem D		Totals	
Language	S	L	S	L	S	L	S	L	Sets	People ▲
C++	273	256	264	214	162	68	244		1481	317 / 19
Java	48	48	43	31	32	16	45		263	54 / 6
Python	16	15	9	8	4	2	6	1	61	17 / 1
C#	9	9	8	6	5	3	9		49	13

Google Code Jam 2013 – Round 2 – user: bmerry



Language Popularity

	Problem A		Problem B		Problem C		Problem D		Totals	
Language	S	L	S	L	S	L	S	L	Sets	People ▲
C++	1130	617	841	673	298	153			3712	1315 / 393
Java	202	87	134	116	38	16			593	233 / 61
Python	142	70	113	99	10	6	1	1	442	197 / 45
C#	29	8	18	17	6	1			79	37 / 3

Coding Competitions 101

What does it look like?

- Google Code Jam – 2 ½ hours, ranking by score and time
  - Problem statement including limits with test input dataset and output
  - Model, code, test, debug, tune for performance...
  - Download the input dataset and start the clock
  - Run your code on your computer and get an output
  - Upload the output within 4 minutes
  - The online judge declares it correct or incorrect
  - Score points or try again with a different dataset

Coding Competitions 201

Contraints and assets

- Constraints → Execution time and used memory
  - CPU (speed and cores), RAM (size), Storage (size and speed)
- Assets → Time
  - Modeling time
  - Coding time
  - Testing time
  - Debugging time
  - Performance-tuning time

PyPy competition limitations

A few libraries are not ported to CFFI yet:

- NumPy – http://www.numpy.org/ (NumPyPy is a partial reimplementation)
- SciPy – Scientific computing library – http://www.scipy.org/
- Gmpy2 – Numerical library – http://code.google.com/p/gmpy/

Small tasks don't perform well:

- Fast tasks suffer from the warm-up slowdown
- Small memory tasks suffer the bigger memory footprint of PyPy

PyPy competition setup

Setup a clean virtualenv with the latest PyPy realease with:

- IPython – http://ipython.org/
- PyFlake – https://pypi.python.org/pypi/pyflakes
- NumPyPy – in the PyPy distribution
- NetworkX – Graph library – http://networkx.github.io/
- PIL – Image processing – http://www.pythonware.com/products/pil/

Own library of algorithms:

- PriorityDictionary – partially ordered dict – http://goo.gl/aWg6r
- Dijkstra Shortest Path Algorithm – http://goo.gl/pQaLo
- GCD, LCM, binom, isqrt, etc...

Store Credit

Problem Statement:

https://code.google.com/codejam/contest/351101/dashboard#s=p0

```
from sys import stdin

T = int(stdin.next())
for t in xrange(T):
    C = int(stdin.next())
    I = int(stdin.next())
    P = map(int, stdin.next().split())
    for i, p in enumerate(P):
        if C-p in P[i+1:]:
            break
    print 'Case #%d: %d %d' % (t+1, i+1, i+1 + P[i+1:].index(C-p)+1)
```

T9 Spelling

Problem Statment:

https://code.google.com/codejam/contest/351101/dashboard#s=p2

```
from sys import stdin
T9R = {'2': 'abc', '3': 'def', '4': 'ghi', '5': 'jkl',
    '6': 'mno', '7': 'pqrs', '8': 'tuv', '9': 'wxyz', '0': ' ',
}
T9 = {} # building the T9 mapping because I'm lazy
for k, v in T9R.items():
    for i, c in enumerate(v):
        T9[c] = k * (i+1)
T = int(stdin.next())
for t in xrange(T):
    M = stdin.next().strip('\n') # keeps leading and trailing spaces!!
    KP = T9[M[0]]
    for c in M[1:]:
        kp = T9[c]
        if kp[0] == KP[-1]:
            KP += ' '
        KP += kp
    print 'Case #%d: %s' % (t+1, KP)
```

Reverse Word

Problem Statement:

https://code.google.com/codejam/contest/351101/dashboard#s=p1

```
from sys import stdin
T = int(stdin.readline())
for t in xrange(T):
    print 'Case #%d: %s' % (t+1, ' '.join(reversed(stdin.readline().split())))
```

Snapper Chain - GCJ Qualification Round 2010

Problem statement:

https://code.google.com/codejam/contest/351101/dashboard#s=p0

Complexity Analysis:

- First we need to check how the most stupid solution scales and if we stand a chance to attack the large input with it: O(T * N * K) ≈ 10^5 * 30 * 10^8 ≈ 10^14. No chance, we need to come up with something.
- What is the computational upper limit of your machine, right? In the most optimistic case and on a good machine you can crunch of the order of 2*10^9 operations per second for 8 minutes so your upper bound is roughly 2*10^9 * 480 ≈ 10^12. But in the real world you better keep thinking until the numer of loop iterations required to solve all test cases gets close to 10^10.

Snapper Chain - GCJ Qualification Round 2010

Tricks:

- There are at most 30 different chains you need to fully solve not T so your problem is really of order O(N*N*K) ≈ 10^11
- try lo leverage binary representation and binary operations to compute the snapper chain state as these operations are really fast and especially considering that the longer chain fits comfortably into a 32 bit int. The algorithm can be written as add and xor of integers
- each of the K iteration looks just like adding 1 to the previous state!!

```
from sys import stdin
T = int(stdin.next())
for t in xrange(1, T+1):
    N, K = map(int, stdin.next().split())
    s = 2 ** N
    print 'Case #%d: %s' % (t, 'ON' if (K % s) == (s - 1)  else 'OFF')
```

Tide Goes In, Tide Goes Out - Code Jam 2012 - Round 1B

https://code.google.com/codejam/contest/1836486/dashboard#s=p1

```
from sys import stdin
import heapq as hp
T = int(stdin.next())
for tc in range(1, T+1):
    H, N, M = map(int, stdin.next().split())
    CH = [map(int, stdin.next().split()) for i in range(N)]
    CL = [map(int, stdin.next().split()) for i in range(N)]
    T = [[2**31]*M for i in xrange(N)]
    T[0][0] = 0.
    F = [(T[0][0], 0, 0)]
    while len(F):
        t, j, i = hp.heappop(F)
        if j==N-1 and i==M-1:
            break
        for jj, ii in [(j-1,i), (j,i-1), (j+1,i), (j,i+1)]:
            if not (0<=jj<N and 0<=ii<M): continue
            if min(CH[j][i],CH[jj][ii]) - max(CL[j][i],CL[jj][ii]) < 50: continue
            ts = max(t, (H + 50 - CH[jj][ii])/10.)
            if ts > 0.:
                ts += 1. if (H-10*ts-CL[j][i]) >= 20 else 10.
            if ts < T[jj][ii]:
                T[jj][ii] = ts
                hp.heappush(F, (ts, jj, ii))
    print 'Case #%s: %s' % (tc, T[-1][-1])
```

PyPy competition limitations

A few libraries are not ported to CFFI yet:

- NumPy – http://www.numpy.org/ (NumPyPy is a partial reimplementation)
- SciPy – Scientific computing library – http://www.scipy.org/
- Gmpy2 – Numerical library – http://code.google.com/p/gmpy/

Small tasks don't perform well:

- Fast tasks suffer from the warm-up slowdown
- Small memory tasks suffer the bigger memory footprint of PyPy

URLs

```
# setup script
http://pastebin.com/KGR8i5wW
# Store Credit solution
http://pastebin.com/s7jb6TB0
# EP gcj
http://goo.gl/3q7zN
# gcj stats
http://www.go-hero.net/jam
```

Lessons learned

PyPy advantages over Python

- Modeling time
  - can code at low level when needed, much like C++
- Coding time and Testing time
  - simple code usually runs fast enough
- Performance-tuning time
  - can skip several optimization techniques
  - usually good speed and memory performance for heavy tasks
- Debugging time
  - simple code + less optimization == easier debugging

Thanks

Alessandro Amici <a.amici@bopen.eu>

B-Open Solutions – http://bopen.eu