●●●● Very Frequent 0/16

Simulation

Author: Darren Yao
Contributors: Allen Li, Siyong Huang, Juheon Rhee

Directly simulating the problem statement.

Language: Python Edit This Page

TABLE OF CONTENTS

RESOURCES

IUSACO	5 - Simulation	This module is based on Chapter 5 of Darren Yao's book

Since there's no formal algorithm involved, the intent of the problem is to assess competence with one's programming language of choice and knowledge of built-in data structures. At least in USACO Bronze, when a problem statement says to find the end result of some process, or to find when something occurs, it's usually sufficient to simulate the process naively*.

Example 1

Shell Game
Bronze - Easy

Focus Problem – try your best to solve this problem before continuing!

Solution

We can simulate the process. Store an array that keeps track of which shell is at which location, and Bessie's swapping can be simulated by swapping elements in the array. Then, we can count how many times Elsie guesses each shell, and the maximum points she can earn is the maximum amount of times a shell is guessed.

Copy PYTHON

```
read = open("shell.in")

n = int(read.readline())

# shell_at_pos[i] stores the label of the shell located at position i
```

```
# The shells can be placed arbitrarily at the start.
shell_at_pos = [i for i in range(3)]

# counter[i] stores the number of times the shell with label i was picked
counter = [0 for _ in range(3)]

for _ in range(n):
    # Zero indexing: offset all positions by 1
    a, b, g = [int(i) - 1 for i in read.readline().split()]

    # Perform Bessie's swapping operation
    shell_at_pos[a], shell_at_pos[b] = shell_at_pos[b], shell_at_pos[a]

    # Count the number of times Elsie guesses each particular shell
    counter[shell_at_pos[g]] += 1

print(max(counter), file=open("shell.out", "w"))
```

Example 2

Mixing Milk

Bronze - Easy

Focus Problem – try your best to solve this problem before continuing!

Solution

We can simulate the process of pouring buckets. The amount of milk poured from bucket i to bucket j is the smaller of the amount of milk in bucket i (which is m_i) and the remaining space in bucket j (which is $c_j - m_j$). We can just handle all of these operations in order, using an array c to store the maximum capacities of each bucket, and an array m to store the current milk level in each bucket, which we update during the process. Example code is below.

Copy PYTHON

```
N = 3  # The number of buckets (which is 3)
TURN_NUM = 100

# capacity[i] is the maximum capacity of bucket i
capacity = [0 for _ in range(N)]
# milk[i] is the current amount of milk in bucket i
milk = [0 for _ in range(N)]
with open("mixmilk.in") as read:
    for i in range(N):
        capacity[i], milk[i] = map(int, read.readline().split())

for i in range(TURN_NUM):
    bucket1 = i % N
    bucket2 = (i + 1) % N

    """
    The amount of milk to pour is the minimum of the remaining milk
    in bucket 1 and the available capacity of bucket 2
    """
    amt = min(milk[bucket1], capacity[bucket2] - milk[bucket2])

    milk[bucket1] -= amt
    milk[bucket2] += amt

with open("mixmilk.out", "w") as out:
    for m in milk:
        print(m, file=out)
```