ADS HW3 Report

404261476 資工四甲 楊培澤

Problem : Eat or not to Eat

Solution :

This problem is the most difficult one among the four assignment, and I have no better idea than brute force. Count the GCD of the all cows' period and find out the least milk that produced by the cows, and if there are more than one cow, pass and go on to the next stage.

Discussion :

Is brute force the only way for this problem?

Code :

```python
def gcd(a, b):
    while b:
        a, b = b, a%b
    return a

if __name__ == '__main__':
    # dir
    input_dir = './data/Eat or Not to Eat.in'
    output_dir = './data/Eat or Not to Eat.out'

    # load all data
    data = []
    tmp = [line.strip('\r').strip('\n').split(' ') for line in open(input_dir, 'r').readlines()]
    # tmp = [
    #     ['2'],
    #     ['4'],
    #     ['4', '7', '1', '2', '9'],
    #     ['1', '2'],
    #     ['2', '7', '1'],
    #     ['1', '2'],
    #     ['4'],
    #     ['4', '7', '1', '2', '9'],
    #     ['1', '2'],
    #     ['2', '7', '1'],
    #     ['1', '2']
    # ]
    case = int(tmp[0][0])
    # print(case)
```

```
result = []
c_count = 1
for c in range(case):
    data = []
    full_length = int(tmp[c_count][0])
    for i in range(full_length):
        data.append(tmp[c_count + i + 1])
    c_count += full_length + 1

    # doc
    gcd_n = 1
    ans1 = full_length
    ans2 = 0
    p_count = 0
    milk = [[0 for i in range(1005)] for j in range(1005)]
    eaten = [False for i in range(1005)]
    length = [0 for i in range(1005)]

    # get gcd of all data
    for i in range(1, len(data)+1):
        length[i] = int(data[i-1][0])
        for j in range(length[i]):
            milk[i][j] = int(data[i-1][j+1])
        gcd_result = gcd(gcd_n, length[i])
        gcd_n = gcd_n * length[i] / gcd_result
    # print('gcd_n', gcd_n)
    # print_result(milk, 4)
```

```
while True:
    stop_condition = False
    for i in range(int(gcd_n)):
        count = 0
        cow = 0
        min_num = 1000
        # print()
        for j in range(1, full_length+1):
            # print(i, j)
            if eaten[j] == False:
                if milk[j][i%length[j]] < min_num:
                    min_num = milk[j][i%length[j]]
                    cow = j
                    count = 1
                elif milk[j][i%length[j]] == min_num:
                    count += 1
        if count == 1:
            stop_condition = True
            eaten[cow] = 1
            ans1 -= 1
            ans2 = p_count * gcd_n + i + 1
        # print(count, stop_condition)
    p_count += 1
    if stop_condition == False:
        break
# print(ans1, int(ans2))
result.append([ans1, int(ans2)])
```

```
# check accuracy
error = []
ground_truth = open(output_dir, 'r').readlines()
if len(result) != len(ground_truth):
    print('Result count inconsist')
else:
    count = 0
    for i in range(len(result)):
        word_list = ground_truth[i].strip('\r').strip('\n').split(' ')
        if result[i][0] == int(word_list[0]) and result[i][1] == int(word_list[1]):
            count += 1
        else:
            error.append({
                'line': i+1,
                'result1': result[i][0],
                'ground_truth1': int(word_list[0]),
                'result2': result[i][1],
                'ground_truth2': int(word_list[1])
            })
    if count == len(result):
        print('Match output file')
    else:
        print('Some error in result')
        print(error)

# make output file
filename = ''.join(['./', 'output.txt'])
with open(filename, 'w') as f:
    for i in range(len(result)-1):
        f.write("%s %s\n" % (result[i][0], result[i][1]))
    f.write("%s %s" % (result[len(result)-1][0], result[len(result)-1][1]))
print('Output predict file...')
```