Критерии отбора победителей и призеров первого этапа

Количество баллов, набранных при решении задач одной попытки, суммируется. Если участник решал задачи, как в первой, так и во второй попытке, то выбирается попытка с большей суммой баллов. Призерам первого отборочного этапа необходимо было набрать 16 баллов (для 9 класса) и 16 баллов (для 10-11 класса). Победители первого отборочного этапа должны были набрать 29 баллов.

2. ВТОРОЙ ОТБОРОЧНЫЙ ЭТАП

Второй отборочный этап проводится в онлайн формате. Работы оцениваются автоматически средствами системы тестирования. Продолжительность второго этапа составляет 6 недель. Данный этап включает в себя задачу анализа данных и алгоритмическую задачу. Они носят междисциплинарный характер и воссоздают специфику инженерной задачи заключительного этапа в более простой форме. Решение каждой задачи дает определенное количество баллов, которые зачисляются в полном объеме за правильное решение задачи. На данном этапе можно получить суммарно от 0 до 13 баллов. При этом за первую задачу максимум можно получить 5 баллов, а за вторую — 10 (теоритически возможный, но практически недостижимый предел). Объем и сложность задач подобраны таким образом, чтобы затруднить достижение максимального балла единолично. Это необходимо, чтобы обеспечить командную работу участников и распределение обязанностей внутри команды. Участники не ограничиваются в выборе языка программирования и количестве попыток решения. Задачи по программированию выкладываются двумя партиями: в начале второго этапа и в середине. Команды могут выполнять задачи в любом порядке.

Задачи второго этапа

7.1. Алгоритмы

Задача 7.1.1. (3 балла)

Две страны находятся в напряженных дипломатических отношениях. При этом в каждой стране есть по S послов второй страны. Каждая из стран в качестве дипломатического шага может выслать некоторое количество послов.

Согласно дипломатическому протоколу сколько-бы послов не выслала одна из стран, вторая сразу высылает такое-же количество послов. Таким образом послов в обоих странах всегда одинаково.

После того как одна из стран инициировала высылку послов, и вторая выслала такое-же количество, вторая страна может пойти на эскалацию и выслать еще послов. После этого первая страна так-же высылает такое-же количество послов, и после чего может опять эскалировать конфликт.

Первая страна может выслать при эскалации количество послов равное 0, N_1 или N_2 ... или N_k (всего k вариантов числа высылки при эскалации), вторая страна может выслать M_1, M_2, ... M_p (всего p вариантов высылки при эскалации). При ответном действии страна всегда сначала высылает такое же количество послов что

и другая, и только потом при эскалации выбирает значение из списка. В конфликте проигрывает страна, которая не может эскалировать конфликт.

Вопрос: При каких числах послов (от 1 до 40000) может победить первая страна при оптимальной стратегии соперницы? (возможен вариант, что конфликт не закончится никогда, значит первая страна не побеждает)

Вход: списки значений по сколько послов может высылать первая и вторая страна
Выход: список значений от 1 до 40000 включительно при которых выигрывает первая страна

Примеры

Пусть в каждой стране по 7 послов Первая старана может выслать 2 или 4 посла, а Вторая страна 3 или 4.

1. первая страна высылает 2х послов, вторая отвечает и в каждой стране остается по 5 послов
2. вторая страна высылает 4х послов, в каждой стране остается по 1 послу
3. первая страна не может выслать, она проиграла

Решение

```
def whoWin(pl, pl_ab, wins, stLen ):
    if (stLen,pl) in wins.keys():
        return wins[(stLen,pl)]
    for myTry in pl_ab[pl]:
        if myTry > stLen:
            continue
        nRes = stLen-myTry
        if (nRes,1-pl) in wins.keys() and  wins[(nRes,1-pl)] == pl:
            return pl
    for myTry in pl_ab[pl]:
        if myTry > stLen:
            continue
        nRes = stLen-myTry
        if (nRes,1-pl) not in wins.keys() and whoWin(1-pl,pl_ab,wins, nRes) == pl:
            return pl
    return 1-pl

def getAnswer(pl_abT):
    pls = [0,1]
    winsF = {(0,0):1, (0,1):0}
    for pl in pls:
        for iS in range(min(pl_abT[1-pl])-1):
            winsF[(iS,1-pl)] = pl
            for iM in pl_abT[pl]:
                winsF[(iS+iM,pl)] = pl
    res = list()
    for i in winsF.items():
        if i[0][1] == 0 and i[1]==0:
            res.append(i[0][0])
    #print(res)
    for i in range(maxPosl+1):
        if  (i,0) not in winsF.keys():
```

```
            winsF[(i,0)] = whoWin(0,pl_abT,winsF, i)
            if  winsF[(i,0)]==0:
                res.append(i)
        if  (i,1) not in winsF.keys():
            winsF[(i,1)] = whoWin(1,pl_abT,winsF, i)
    res.sort()
    return res
```

Код проверки и генерации правильного решения

```
def solve(dataset):
    ab = ast.literal_eval(dataset)
    ab = [ list(set(i)) for i in ab]
    #print(ab)
    return str(getAnswer(ab))


def check(reply, clue):
    return str(reply) == str(clue)
```

7.2. Анализ данных

Задача 7.2.1. (10 баллов)

Данная задача является задачей разработки алгоритма машинного обучения.

В качестве входный данных предоставляется информация о абитуриентах. В качестве результата предсказания необходимо определить среднюю оценку по каждому из онлайн курсов, которые пройдет абитуриент. При этом надо учитывать, что большинство абитуриентов проходит 1 онлайн курс, а если абитуриент не проходил онлайн курс, то средняя оценка 0. Оценки за курсы это столбцы содержащие avg (или Avg) в названии. Победители определяются по сравнению решений по метрике MSE (нормированной сумме квадратов разностей предсказанного значения и верного)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(\overline{Y_i} - Y_i\right)^2$$

В качестве результата берется отношение среднего квадратичного отклонения представленных данных к среднему квадратичному отклонению константы 0 (как если бы на вся матрица ответов была заполнена нулями).

Количество набранных баллов определяется как $10*(1-\frac{MSE}{MSEZ})$, где MSE-средняя квадратичная ошибка решения, а $MSEZ$ - средняя квадратичная ошибка нулевого решения (всем устанавливается значение ноль). Таким образом отправка нулевого решения (или хуже нулевого) дает 0 баллов. Максимальный бал, если угадано все за данную задачу - 10 баллов.

Набор данных

- Обучающий дата сет:
  http://files.jet.su/d/NTI_BD2018_2_2_train
- Тестовый сет:
  http://files.jet.su/d/NTI_urfy_test_2

- Пример результата:
  http://files.jet.su/d/NTI_URFU_EX_RESULT

Чтобы отправить решение нажмите скачать данные, там будет текст, на него можно не обращать внимание технические особенности платформы, тестовые данные тут:
http://files.jet.su/d/NTI_urfy_test_2

Расшифровка данных:
https://docs.google.com/spreadsheets/d/1BvjFG1fTJaEmfwgHK8-VttW1MN5Q5Y7OohzYUigs9Kk/editgid=944357857

Решение

```
import os
from os import listdir
from os.path import isfile, join
import pandas as pd
import sklearn as sk
from sklearn.metrics import mean_squared_error
from  sklearn.model_selection import *
from sklearn.ensemble import RandomForestRegressor
from sklearn import preprocessing as pp
import numpy as np

finalDftr = pd.read_csv("urfuTrain.csv")
finalDftr = finalDftr.drop(["index",'brs_id'],axis=1)
ycols = list()
for c in finalDftr.columns:
    if "vg" in c:
        ycols.append(c)


finalDftr.fillna(0)
x = finalDftr.drop(ycols, axis=1)
y = finalDftr[ycols]
for i,c in enumerate(x.columns):
    if (x[c].dtype != 'object'):
        continue
    le = pp.LabelEncoder()
    le.fit(x[c].astype('str'))
    x[c]=le.transform(x[c].astype('str'))
xtrD, xtt,ytr,ytt = train_test_split(x,y, test_size= 0.8)

cblist = list()
for c in ytr.columns:
    yctr=ytr[c]
    cb = RandomForestRegressor()
    cb.fit(xtrD,yctr)
    cblist.append(cb)
ytp = pd.DataFrame()
for i,c in enumerate(ytr.columns):
    ytcp = pd.DataFrame(pd.Series(cblist[i].predict(xtt),name=c))
    ytp=pd.concat([ytp,ytcp], axis=1)

mean_squared_error(ytp,ytt)
```

Код проверки и генерации правильного решения

```python
import csv
import random
import requests
urlT = 'https://stepik.org/media/attachments/lesson/61999/xTestFinal__1_.csv'
import codecs

def solve(dataset):
    request = requests.get(urlR)
    csvMy = request.text

    return csvMy

def check(reply, clue):
    clueL = clue.splitlines()[1:]
    replyL = reply.splitlines()[1:]

    sumErrZ = 0
    sumErr = 0
    countC = 0
    for i,line in enumerate(clueL):
        splitedClue = line.split(",")[1:]
        if i>=len(replyL):
            splitedReply = list()
        else:
            splitedReply = replyL[i].split(",")[1:]
        for k, clueV in enumerate(splitedClue):
            if k>=len(splitedReply):
                repV=0
            else:
                repV=splitedReply[k]
            sumErr+=(float(clueV)-float(repV))**2
            sumErrZ+=(float(clueV))**2
            countC+=1
    err = sumErr/countC
    errZ = sumErrZ/countC
    if err == 0:
        return True
    res = 1-err/float(errZ)
    if res<0:
        res=0
    balls = 10*res
    resT = "В перовой задаче набрали "+str(balls)+" баллов"

    return res, resT
```

Критерии определения призеров и победителей второго этапа

Количество баллов, набранных при решении всех задач суммируется. Призерам второго отборочного этапа было необходимо набрать 35 баллов (для участников из 9 класса) и 45 (для участников из 10-11 класса). Победители второго отборочного этапа должны были набрать 63 балла и выше.