Заключительный этап

Задача по искусственному интеллекту

Предскажи пол

Условия:

VK продвигает сервисы, товары и услуги разнообразными способами. Одним из основных является реклама. Качественная реклама требует хорошего понимания аудитории. Вам предложен датасет. Необходимо предсказать пол человека.

Представьте, что вы работаете в бизнес-юните VK Реклама компании VK.

Приближаются праздники. Крупная компания, занимающаяся продажей подарочных наборов, хочет запустить рекламу подарков в соцсетях ВКонтакте, Одноклассники и Дзен.

Вам нужно разработать модель машинного обучения, которая будет предсказывать пол пользователя соцсетей. Это необходимо, чтобы реклама была эффективной: демонстрировалась пользователям соответствующего гендера.

Компания VK заинтересована в получении этого крупного заказа. Сотрудник, который представит модель машинного обучения, быстро и точно предсказывающую пол пользователя, получит вознаграждение.

В вашем распоряжении есть табличный датасет, на котором вы можете тренировать модели машинного обучения. У вас есть 4 часа и 10 попыток, чтобы разработать свой вариант модели машинного обучения.

Описание датасета

Таблицы

train_labels.csv:
- «user_id» - id пользователя;
- «target» - пол пользователя (1 / 0).

train.csv, test.csv;
- «request_ts» - server timestamp of request;
- «user_id» - id пользователя (см. п.1);
- «referer» - url, где показывается реклама. В данном случае захэшировано 2 части url:
1) domain - домен урла;
2) path - все что после domain. Например, https://a758bf6/1432d3f1, а 758bf6 - domain, 1432d3f1 - path.
- «geo_id» - id geo;
- «user_agent» - строка user_agent.

referer_vectors.csv:
- «component0» - … - «component9» - числа, которые несут в себе информацию о url. Их нельзя как-либо интерпретировать;
- «referer» - url, где показывается реклама (см. п.2).

geo_info.csv:
- «geo_id» - id geo (см. п.2);
- «country_id» - id страны;

- «region_id» - id региона;
- «timezone» - часовой пояс для geo.

Требования

Для каждого пользователя (user) из файла test_users.csv необходимо предсказать пол. Их рекламные запросы лежат в файле test.csv.

Формат вывода

Формат вывода соответствует train_labels.csv.

Решение

Данный `doc` представляет правильное решение финального задания олимпиады, направленной на предсказание пола человека.

In [1]:
```
# imports
import os
import sys
import re
from tqdm import tqdm
from datetime import datetime

import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from catboost import CatBoostClassifier
from sklearn.metrics import accuracy_score, f1_score, roc_auc_score
```

In [ ]:

In [ ]:

In [ ]:

Загружаем данные

Загружаем `train` и `test` рекламные запросы.

In [2]:
```
train = pd.read_csv(os.path.join('data', 'data_from_allcups',
'train.csv'), sep=';')
test = pd.read_csv(os.path.join('data', 'data_from_allcups', 'test.csv
'), sep=';')
```

In [3]:
```
train.head()
```

Out[3]:

	request_ts	user_id	referer	geo_id	user_agent
0	1701011363	fb858e8e0a2bec074450eaf94b627fd3	https://9b48ee5/	4799	{'browser': 'Chrome Mobile', 'browser_version'...
1	1700986581	46a5f128fd569c764a92c2eaa788095e	https://9b48ee5/	8257	{'browser': 'Chrome Mobile', 'browser_version'...
2	1701011071	5a74e9ac53ffb21a20cce117c0ad77ba	https://9634fd0/1409e548	3150	{'browser': 'Yandex Browser', 'browser_version...
3	1700992803	af735816ca19115431ae3d89518c8c91	https://9b48ee5/	2740	{'browser': 'Chrome Mobile', 'browser_version'...
4	1701021666	364f0ae0a3f29a685c4fb5bae6033b9a	https://9b48ee5/	4863	{'browser': 'Yandex Browser', 'browser_version...

In [4]
```
test.head()
```

Out[4]:

	request_ts	user_id	referer	geo_id	user_agent
0	1700993094	c2802dadd33d8ae09bb366bdd41212ea	https://9b48ee5/	8816	{'browser': 'Chrome Mobile', 'browser_version'...
1	1701005579	e5b1988db74527ec092f28b0bbfdaac9	https://9b48ee5/	3663	{'browser': 'Chrome', 'browser_version': '116....
2	1700969752	6ef1eedbdb72554e53e69782066065c5	https://72879b4/12411b9e	2336	{'browser': 'Chrome', 'browser_version': '114....
3	1700991608	7e057293ecae62985a327b7af51858ea	https://9b48ee5/	9652	{'browser': 'Chrome Mobile', 'browser_version'...
4	1701019815	a27bd7ce8828497823fa8d5d05e7bbf7	https://9b48ee5/	3871	{'browser': 'Chrome Mobile', 'browser_version'...

In [ ]:

Загружаем `train` и `test` разметку пользователей.

In [5]:
```
train_labels = pd.read_csv(os.path.join('data', 'data_from_allcups',
'train_labels.csv'), sep=';')
test_labels = pd.read_csv(os.path.join('data', 'data_from_allcups', '
test_labels.csv'), sep=';')
test_users = pd.read_csv(os.path.join('data', 'data_from_allcups', 't
est_users.csv'), sep=';')
```

In [6]:
```
train_labels.head()
```

Out[6]:

	user_id	target
0	fb858e8e0a2bec074450eaf94b627fd3	0
1	46a5f128fd569c764a92c2eaa788095e	0
2	5a74e9ac53ffb21a20cce117c0ad77ba	0
3	af735816ca19115431ae3d89518c8c91	0
4	364f0ae0a3f29a685c4fb5bae6033b9a	0

In [7]:
```
test_labels.head()
```

Out[7]:

	user_id	target
0	c2802dadd33d8ae09bb366bdd41212ea	0
1	e5b1988db74527ec092f28b0bbfdaac9	0
2	6ef1eedbdb72554e53e69782066065c5	0
3	7e057293ecae62985a327b7af51858ea	0
4	a27bd7ce8828497823fa8d5d05e7bbf7	0

In [8]:
```
test_users.head()
```

Out[8]:

	user_id
0	c2802dadd33d8ae09bb366bdd41212ea
1	e5b1988db74527ec092f28b0bbfdaac9
2	6ef1eedbdb72554e53e69782066065c5
3	7e057293ecae62985a327b7af51858ea
4	a27bd7ce8828497823fa8d5d05e7bbf7

In [ ]:

Загружаем `geo_info`.

In [9]:
```
geo_info = pd.read_csv(os.path.join('data', 'data_from_allcups', 'geo_info.csv'), sep=';')
```

In [10]:
```
geo_info.head()
```

Out[10]:

	geo_id	country_id	region_id	timezone_id
0	6447	c31b4e	470e75	f6155e
1	8730	a0a6e9	NaN	d816ca
2	7769	e878d4	NaN	ec4385
3	7330	c31b4e	23f9c2	f6155e
4	600	c31b4e	6dbc37	e56e80

In [ ]:

Загружаем данные `referer_vectors`.

In [11]:
```
referer_vectors_frame = pd.read_csv(os.path.join('data', 'data_from_allcups', 'referer_vectors.csv'), sep=';')
```

In [12]:
```
referer_vectors_frame.head()
```

Out[12]:

	component0	component1	component2	component3	component4	component5	component6	component7	component8	component9	referer
0	16708	-3741	11395	-1597	-3212	6269	5610	-15351	13779	14102	https://a6899a4/15652e67
1	11731	4045	22213	-1184	-8992	9381	-3496	-3120	-899	16817	https://9b48ee5/
2	10551	2947	12282	-470	16222	4472	-3316	9606	4197	18948	https://7a4c700/161af7e3
3	12816	20498	-10110	7731	-569	12035	3014	6398	11439	-271	https://9653126/159bc361
4	3710	11096	11333	14673	8030	1852	10554	11625	4306	13210	https://72879b4/125c29e6

In [ ]:

In [ ]:

In [ ]:

Делаем преобразования данных и генерируем новые фичи

Создаем фичу `request_hour`

In [13]:
```
train['request_hour'] = train.request_ts.apply(lambda elem: datetime.fromtimestamp(elem).hour).astype('str')
test['request_hour'] = test.request_ts.apply(lambda elem: datetime.fromtimestamp(elem).hour).astype('str')
```

In [14]:
```
train.loc[:, ['request_hour']].head()
```

Out[14]:

	request_hour
0	18
1	11
2	18
3	13
4	21

In [ ]:

Создаем фичу `domain`. Чем она может быть полезна? Например, на `domain` - `woman.ru` чаще заходят девушки.

In [15]:
```
train['domain'] = train.referer.apply(lambda elem: elem.replace('https://', '').split('/')[0])
test['domain'] = test.referer.apply(lambda elem: elem.replace('https://', '').split('/')[0])
```

In [16]:
```
train.loc[:, ['domain']].head()
```

Out[16]:

	domain
0	9b48ee5
1	9b48ee5
2	9634fd0
3	9b48ee5
4	9b48ee5

In [ ]:

Генерируем фичи `browser` и `os` из колонки `user_agent`.

In [17]:
```
train_unique_user_agent = set(train.loc[~train.user_agent.isna(), 'user_agent'].unique())
test_unique_user_agent = set(test.loc[~test.user_agent.isna(), 'user_agent'].unique())
unique_user_agent = train_unique_user_agent | test_unique_user_agent
parser_user_agent = dict(map(lambda elem: (elem, eval(elem)), unique_user_agent))

train['browser'] = train.user_agent.apply(lambda elem: parser_user_agent.get(elem, {}).get('browser', 'none'))
train['os'] = train.user_agent.apply(lambda elem: parser_user_agent.get(elem, {}).get('os', 'none'))

test['browser'] = test.user_agent.apply(lambda elem: parser_user_agent.get(elem, {}).get('browser', 'none'))
test['os'] = test.user_agent.apply(lambda elem: parser_user_agent.get(elem, {}).get('os', 'none'))
```

In [18]:
```
train.loc[:, ['browser', 'os']].head()
```

Out[18]:

	browser	os
0	Chrome Mobile	Android
1	Chrome Mobile	Android
2	Yandex Browser	Android
3	Chrome Mobile	Android
4	Yandex Browser	Android

In [ ]:

Делаем `join` таблиц.

In [19]:
```
train = train.merge(train_labels, on='user_id', how='inner')
train = train.merge(geo_info, on='geo_id', how='left')
```

```
train = train.merge(referer_vectors_frame, on='referer', how='left')

test = test.merge(test_labels, on='user_id', how='inner')
test = test.merge(geo_info, on='geo_id', how='left')
test = test.merge(referer_vectors_frame, on='referer', how='left')
```

In [ ]:

Проверяем есть ли `null` значения в колонках, на которых будем обучать модель машинного обучения.

In [20]:
```
for col in train.columns:
 temp_count = train.loc[train.loc[:, col].isna()].shape[0]
 if temp_count > 0:
  print(f"{col}: {temp_count}")
```
```
user_agent: 1
region_id: 50620
```
In [21]:
```
for col in test.columns:
 temp_count = test.loc[test.loc[:, col].isna()].shape[0]
 if temp_count > 0:
  print(f"{col}: {temp_count}")
```
```
region_id: 8928
```
In [ ]:

Заполняем `nan` значения в колонки `region_id`.

In [22]:
```
train = train.fillna({'region_id': 'none'})
test = test.fillna({'region_id': 'none'})
```

In [ ]:

In [23]:
```
train.shape
```

Out[23]:
```
(601290, 23)
```
In [24]:
```
train.user_id.nunique()
```
Out[24]:
```
500000
```
In [25]:
```
train.loc[0, :]
```
Out[25]:
```
request_ts            1701011363
user_id       fb858e8e0a2bec074450eaf94b627fd3
referer           https://9b48ee5/
geo_id                  4799
user_agent  {'browser': 'Chrome Mobile', 'browser_version'...
request_hour              18
domain               9b48ee5
browser              Chrome Mobile
os                Android
target                 0
country_id               c31b4e
region_id               470e75
```

```
timezone_id            f6155e
component0             11731
component1             4045
component2             22213
component3             -1184
component4             -8992
component5             9381
component6             -3496
component7             -3120
component8             -899
component9             16817
Name: 0, dtype: object
```

```
In [ ]:
```

```
In [26]: test.shape
Out[26]: (105109, 23)
```

```
In [27]: test.user_id.nunique()
Out[27]: 85000
```

```
In [28]: test.loc[0, :]
Out[28]: request_ts             1700993094
         user_id       c2802dadd33d8ae09bb366bdd41212ea
         referer            https://9b48ee5/
         geo_id                8816
         user_agent  {'browser': 'Chrome Mobile', 'browser_version'...
         request_hour             13
         domain               9b48ee5
         browser              Chrome Mobile
         os                Android
         target                 0
         country_id              c31b4e
         region_id              36e3f3
         timezone_id             f6155e
         component0              11731
         component1              4045
         component2              22213
         component3              -1184
         component4              -8992
         component5              9381
         component6              -3496
         component7              -3120
         component8              -899
         component9              16817
         Name: 0, dtype: object
```

```
In [ ]:
In [ ]:
In [ ]:
```

Создаем модель машинного обучения

Опредяем фичи, которые пойдут в модель.

In [29]:
```
cat_features = ['request_hour', 'domain', 'browser', 'os', 'country_
id', 'region_id']
embedding_regex = re.compile('component')
embedding_cols = list(filter(embedding_regex.match, train.columns))
```

In [ ]:

Делаем `train_test_split`, чтобы избежать переобучения.

In [30]:
```
X = train.loc[:, ['user_id'] + cat_features + embedding_cols]
y = train.loc[:, 'target'].values
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.
2, random_state=42)

X_test = test.loc[:, ['user_id'] + cat_features + embedding_cols]
```

In [ ]:

В качестве архитектуры был выбран `CatBoostClassifier`, так как в основном модели архитектуры `boosting` лучше всего себя показывают на табличных данных.

Почему CatBoostClassifier?

- Из коробки предоставляет эффективный механизм обработки категориальных фичей.

Но в целом можно были бы выбрать любой другой алгоритм `boosting` или классификации. Но тогда необходимо было бы делать дополнительную обработку категориальных фичей.

In [31]:
```
model = CatBoostClassifier(
 loss_function='Logloss',
 iterations=1000,
 thread_count=-1,
 boosting_type='Plain',
 early_stopping_rounds=15,
 auto_class_weights='Balanced',
 random_seed=42
)
```

In [32]:
```
model.fit(
 X_train.loc[:, cat_features + embedding_cols],
 y_train,
 cat_features=cat_features,
 eval_set=(X_val.loc[:, cat_features + embedding_cols], y_val),
 verbose=100
)
Learning rate set to 0.145682
0:      learn: 0.6356408       test: 0.6344862best: 0.6344862 (0)
        total: 184ms    remaining: 3m 3s
```

```
100:    learn: 0.4138089       test: 0.4022551best: 0.4022551 (100)
        total: 13.5s    remaining: 2m
200:    learn: 0.4017066       test: 0.3908834best: 0.3908834 (200)
        total: 26.6s    remaining: 1m 45s
300:    learn: 0.3953927       test: 0.3855973best: 0.3855973 (300)
        total: 39.4s    remaining: 1m 31s
400:    learn: 0.3915297       test: 0.3831783best: 0.3831783 (400)
        total: 51.9s    remaining: 1m 17s
500:    learn: 0.3884363       test: 0.3813544best: 0.3813544 (500)
        total: 1m 4s    remaining: 1m 4s
600:    learn: 0.3859545       test: 0.3800051best: 0.3800051 (600)
        total: 1m 17s   remaining: 51.5s
700:    learn: 0.3840300       test: 0.3792664best: 0.3792664 (700)
        total: 1m 30s   remaining: 38.4s
800:    learn: 0.3820074       test: 0.3783316best: 0.3783316 (800)
        total: 1m 43s   remaining: 25.8s
900:    learn: 0.3803848       test: 0.3778145best: 0.3778145 (900)
        total: 1m 56s   remaining: 12.8s
999:    learn: 0.3790082       test: 0.3774107best: 0.3774106 (996)
        total: 2m 8s    remaining: 0us

bestTest = 0.3774105528
bestIteration = 996

Shrink model to first 997 iterations.
```

Out[32]:
```
<catboost.core.CatBoostClassifier at 0x7f02b637e3e0>
```

In [ ]:

In [ ]:

In [ ]:

Оцениваем результат модели

Метрики на `train sample` из обучающего датасета.

In [33]:
```
y_train_pred_proba = model.predict_proba(X_train.loc[:, cat_features
+ embedding_cols])[:, 1]
y_train_pred = np.where(y_train_pred_proba > 0.5, 1, 0)

train_score_frame = X_train.loc[:, ['user_id']]
train_score_frame['target'] = y_train
train_score_frame['pred_proba'] = y_train_pred_proba
train_score_frame['pred'] = y_train_pred

train_score_frame = train_score_frame.groupby(['user_id'], as_index=
False).aggregate({
 'target': 'max',
 'pred_proba': 'mean',
 'pred': 'max'
```

```
})
```

In [34]:
```
print(f"train_accuracy: {np.round(accuracy_score(train_score_frame.target, train_score_frame.pred), 4)}")
print(f"train_f1: {np.round(f1_score(train_score_frame.target, train_score_frame.pred), 4)}")
print(f"train_roc_auc: {np.round(roc_auc_score(train_score_frame.target, train_score_frame.pred_proba), 4)}")
```

```
train_accuracy: 0.9039
train_f1: 0.9007
train_roc_auc: 0.9607
```

In [ ]:

Метрики на `val sample` из обучающего датасета.

In [35]:
```
y_val_pred_proba = model.predict_proba(X_val.loc[:, cat_features + embedding_cols])[:, 1]
y_val_pred = np.where(y_val_pred_proba > 0.5, 1, 0)

val_score_frame = X_val.loc[:, ['user_id']]
val_score_frame['target'] = y_val
val_score_frame['pred_proba'] = y_val_pred_proba
val_score_frame['pred'] = y_val_pred

val_score_frame = val_score_frame.groupby(['user_id'], as_index=False).aggregate({
 'target': 'max',
 'pred_proba': 'mean',
 'pred': 'max'
})
```

In [36]:
```
print(f"val_accuracy: {np.round(accuracy_score(val_score_frame.target, val_score_frame.pred), 4)}")
print(f"val_f1: {np.round(f1_score(val_score_frame.target, val_score_frame.pred), 4)}")
print(f"val_roc_auc: {np.round(roc_auc_score(val_score_frame.target, val_score_frame.pred_proba), 4)}")
```

```
val_accuracy: 0.8618
val_f1: 0.8562
val_roc_auc: 0.9104
```

In [ ]:

Метрики на `test` множестве.

In [37]:
```
y_test_pred_proba = model.predict_proba(X_test.loc[:, cat_features + embedding_cols])[:, 1]
y_test_pred = np.where(y_test_pred_proba > 0.5, 1, 0)

test_score_frame = X_test.loc[:, ['user_id']]
```

```
test_score_frame['pred_proba'] = y_test_pred_proba
test_score_frame['pred'] = y_test_pred

test_score_frame = test_score_frame.groupby(['user_id'], as_index=False).aggregate({
 'pred_proba': 'mean',
 'pred': 'max'
})
test_score_frame = test_score_frame.merge(test_labels, on='user_id', how='left')
```

In [38]:
```
test_score_frame.head()
```

Out[38]:

	user_id	pred_proba	pred	target
0	000098f8aa8b68a125148caff0a02827	0.831150	1	0
1	0000eae7d06e8c6033731d1c1ba0c382	0.571418	1	0
2	0000f7fa001a8d49b8e83b91a1215e17	0.905089	1	1
3	0001c39409954fe4b4148a23154fa905	0.089476	0	1
4	00029eeaef3671876a5d119acff7e792	0.079060	0	0

In [39]:
```
print(f"test_accuracy: {np.round(accuracy_score(test_score_frame.target, test_score_frame.pred), 4)}")
print(f"test_f1: {np.round(f1_score(test_score_frame.target, test_score_frame.pred), 4)}")
print(f"test_roc_auc: {np.round(roc_auc_score(test_score_frame.target, test_score_frame.pred_proba), 4)}")
```

```
test_accuracy: 0.8238
test_f1: 0.8216
test_roc_auc: 0.8928
```

In [ ]:
```
# test_score_frame.loc[:, ['user_id', 'pred']].rename(columns={'pred': 'target'}) \
#  .to_csv(os.path.join('data', 'data_from_allcups', 'pred_test_users_v1.csv'), index=False, sep=';')
```

In [ ]:

Стоит отметить, что в целом финальная задача была направлена не только на построение модели машинного обучения, но также на анализ данных. Мы специально часть пользователей из тестового множества добавили в обучающее множество. Таких людей сложно предсказать моделью, так как они очень шумные. Если обнаружить этот инсайт, то метрики значимо увеличатся. На сколько?

In [40]:
```
users_intersection = list(set(train.user_id.unique()) & set(test.user_id.unique()))
```

In [41]:
```
len(users_intersection)
```

Out[41]:
```
6380
```

In [42]:
```
test_score_frame = test_score_frame.merge(train_labels.rename(columns={'target': 'train_target'}), on='user_id', how='left')
```

```
test_score_frame.loc[test_score_frame.user_id.isin(users_intersectio
n), 'pred'] = test_score_frame.loc[test_score_frame.user_id.isin(use
rs_intersection), 'train_target']
test_score_frame = test_score_frame.drop('train_target', axis=1)
test_score_frame = test_score_frame.astype({'pred': 'int'})
```

In [43]:
```
test_score_frame.head()
```

Out[43]:

	user_id	pred_proba	pred	target
0	000098f8aa8b68a125148caff0a02827	0.831150	0	0
1	0000eae7d06e8c6033731d1c1ba0c382	0.571418	1	0
2	0000f7fa001a8d49b8e83b91a1215e17	0.905089	1	1
3	0001c39409954fe4b4148a23154fa905	0.089476	0	1
4	00029eeaef3671876a5d119acff7e792	0.079060	0	0

In [44]:
```
print(f"test_accuracy: {np.round(accuracy_score(test_score_frame.tar
get, test_score_frame.pred), 4)}")
print(f"test_f1: {np.round(f1_score(test_score_frame.target, test_sc
ore_frame.pred), 4)}")
print(f"test_roc_auc: {np.round(roc_auc_score(test_score_frame.targe
t, test_score_frame.pred_proba), 4)}")
```

```
test_accuracy: 0.8795
test_f1: 0.8768
test_roc_auc: 0.8928
```

In [ ]:
```
# test_score_frame.loc[:, ['user_id', 'pred']].rename(columns={'pred
': 'target'}) \
#  .to_csv(os.path.join('data', 'data_from_allcups', 'pred_test_user
s_v2.csv'), index=False, sep=';')
```

Задача A. AI Angry Birds

Имя входного файла: стандартный ввод
Имя выходного файла: стандартный вывод
Ограничение по времени: 0.5 секунд
Ограничение по памяти: 256 мегабайт

ИИ Маруся учится играть в одну из версий игры Angry Birds. Напоминаем, что суть игры заключается в том, чтобы, стреляя из рогатки птичками, уничтожить сооружения, в которых находятся зелёные свинки. В этой версии игры есть 4 домика, в которых находятся зелёные свинки. Крепкость i-го домика равна a_i.

Маруся может стрелять только одним видом птичек. Птичка этого вида имеет силу g, то есть попадание в i-й домик (а Маруся стреляет без промахов) приводит к понижению его крепкости на g. Если крепкость домика после попадания перестала быть положительной, то домик разрушен и находившиеся в нём свинки скрылись в неизвестном направлении. В противном

случае домик ещё стоит и до следующего выстрела Маруси свинки из неразрушенных домиков, за исключением свинок из i-го домика, успеют частично починить i-й домик, а именно свинки из j-го домика, если тот ещё не разрушен, повысят крепкость i-го домика на b_j. Например, если на данный момент разрушен только первый домик, а во второй домик Маруся только что стреляла, то до следующего выстрела крепкость второго домика повысится на b_3+b_4. Заметьте, что по итогу описанного процесса крепкость домика может даже стать больше изначальной.
Приведите тактику для разрушения всех домиков или сообщите, что разрушить все домики невозможно. Заметьте, что не требуется минимизировать суммарное количество выстрелов, но запрещается стрелять в уже разрушенный домик.

Формат входных данных

В i-й из первых четырёх строк ввода содержится два целых числа a_i и b_i, описывающих i-й домик ($0 \le a_i, b_i \le 10^{18}$). Если a_i положительно, то i-й домик имеет крепкость a_i. Если a_i равно 0, то домик уже разрушен.
В последней, пятой строке содержится число g – сила птички ($1 \le g \le 10^{18}$).

Обратите внимание, что входные данные и ответ могут быть достаточно большими, поэтому следует использовать 64-битный тип данных, например `long long` в C/C++, `long` в Java, `int64` в Pascal.

Формат выходных данных

Очевидно, что не имеет смысла переключаться с одной цели на другую до тех пор, пока изначальная цель не уничтожена. Поэтому в качестве ответа, если он существует, выведите 4 строки. i-я строка должна содержать два целых числа k_i и c_i — номер домика, который будет разрушен i-м, иколичество выстрелов из рогатки по k_i-му домику.
Если домик был разрушен изначально, то его нужно вывести тоже, причём не важно в какой из строк.
Если правильных ответов несколько, то выведите любой их них. Стрелять в уже разрушенный домик нельзя и заметьте, что не требуется минимизировать суммарное количество выстрелов.
Если ответа не существует, выведите одно число −1.

Система оценки

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены.

Подзадача	Баллы	Ограничения	Необходимые подзадачи	Информация о проверке
0	–	примеры из условия		полная
1	10	$a_2 = a_3 = a_4 = 0$ $a_i, b_i, g \leqslant 10^9$ для всех i		первая ошибка
2	15	$a_3 = a_4 = 0$ $a_i, b_i, g \leqslant 10^9$ для всех i	1	первая ошибка
3	15	$a_4 = 0$ $a_i, b_i, g \leqslant 10^9$ для всех i	0, 1, 2	первая ошибка
4	15	$b_i = 0$ для всех i		первая ошибка
5	20	$a_i, b_i, g \leqslant 10^6$ для всех i		первая ошибка
6	10	$a_i, b_i, g \leqslant 10^9$ для всех i	0 – 3, 5	первая ошибка
7	15	без дополнительных ограничений	0 – 6	первая ошибка

Пример

стандартный ввод	стандартный вывод
6 3 3 1 7 2 0 70 5	1 2 3 2 4 0 2 1

Пояснение к примеру

После первого выстрела в первый домик его крепкость стала равна 6−5 = 1. После этого свинки из второго и третьего домиков починили первый домик и его крепкость стала равна 1 + 1 + 2 = 4. После этого первый домик был разрушен вторым выстрелом.

После первого выстрела в третий домик его крепкость стала равна 7 − 5 = 2. После этого свинки из второго домика починили третий домик и его крепкость стала равна 2 + 1 = 3. После этого третий домик был разрушен следующим выстрелом, а следующим выстрелом был разрушен и второй домик.

Существуют и другие правильные ответы, поэтому вывод вашей программы может отличаться от указанного в условии задачи.

Решение

Пусть B равно сумме параметров b тех домиков, которые ещё не разрушены (параметр a, то есть крепкость домика положительна). Тогда если i-й домик не разрушен, мы можем разрушить его следующим в одном из двух случаев: либо $a_i \leqslant g$, то есть мы можем разрушить i-й домик одним выстрелом, либо $g > B - b_i$, то есть после первого выстрела и первой починки i-го домика его крепкость уменьшится и каждый последующий выстрел, даже с учётом следующей за ним починки, тоже будет уменьшать крепкость i-го домика.

Для полного решения можно 4 раза пройтись по всем домикам. На каждом проходе проверить, есть ли домик, который можно разрушить. Если есть, брать из таких домиков любой и разрушать. Предположим, что был выбран i-й домик. Если $a_i \leqslant g$, то количество выстрелов равно 1, в противном случае количество выстрелов равно $1 + \left\lceil \frac{a_i - g}{g - (B - b_i)} \right\rceil$. После чего i-й домик оказался разрушен и B уменьшилось на b_i.

Пример полного решения на Python 3:

```python
a, b = [-1] * 4, [-1] * 4
k, c = [-1] * 4, [-1] * 4

B = 0
destroyed = 0
for i in range(4):
    a[i], b[i] = map(int, input().split())
    if a[i] > 0:
        B += b[i]
    else:
        k[destroyed] = i
        c[destroyed] = 0
        destroyed += 1

g = int(input())

while destroyed < 4:
    i = -1
    for j in range(4):
        if a[j] > 0 and (a[j] <= g or g > B - b[j]):
            i = j

    if i == -1:
        print(-1)
        break

    B -= b[i]
    k[destroyed] = i
    c[destroyed] = 1 if a[i] <= g else 1 + (a[i] - B - 1) // (g - B)
    a[i] = 0
    destroyed += 1

else:
    for i in range(4):
        print(k[i] + 1, c[i])
```

Пример медленного расчёта количества выстрелов (подходит для решения подзадачи 5):

```python
    B -= b[i]
    k[destroyed] = i
    c[destroyed] = 1
    a[i] -= g
    while a[i] > 0:
        a[i] -= g - B
        c[destroyed] += 1
    destroyed += 1
```

Пример расчёта количества выстрелов без учёта параметров b (подходит для решения подзадач 1 и 4):

```python
    c[destroyed] = (a[i] + g - 1) // g
```

В подзадаче 2 можно было сначала разрушить те домики, которые можно разрушить за один выстрел, а затем пытаться разрушить тот домик, который не разрушен и имеет наибольшее значение параметра b. Подобную логику можно было использовать и в полном решении.
В подзадаче 3 можно было обработать отдельно каждый порядок разрушения домиков, коих 3! = 6.

В подзадаче 6 можно было не использовать 64-битный целочисленный тип данных, а обойтись 32-битным.
Также существует полное решение, которое перебирает все возможные порядки разрушения домиков, коих 4! = 24, и каждый проверяет, может ли этот порядок быть ответом. Особенно удобно реализовывать такой подход к решению задачи на C++ с использованием функции next_permutation

Задача B. Перестановка с условиями

Имя входного файла: стандартный ввод
Имя выходного файла: стандартный вывод
Ограничение по времени: 1 секунда
Ограничение по памяти: 256 мегабайт

Артем хочет найти перестановку длины n, в которой a элементов больше обоих своих соседей и b элементов меньше обоих своих соседей.
Перестановка — это последовательность длины n целых чисел от 1 до n, в которой все числа встречаются ровно по одному разу. Например, (1), (4, 3, 5, 1, 2), (3, 2, 1) — перестановки, а (1, 1), (4, 3, 1), (2, 3, 4) — нет.

Формат входных данных

В первой строке содержится три целых числа n, a и b ($3 \le n \le 10^5$; $2 \le a, b \le 10^6$) — длина перестановки, число элементов больше обоих своих соседей и число элементов меньше обоих своих соседей.

Формат выходных данных

В качестве ответа выведите «YES» (без кавычек), если существует перестановка, и «NO» в противном случае. Если перестановка существует, то выведите её на второй строке.
Слова «YES» и «NO» можно выводить в любом регистре, например, «YES», «Yes», «yEs», и так далее.

Система оценки

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены.

Подзадача	Баллы	Ограничения	Необходимые подзадачи	Информация о проверке
0	–	примеры из условия		полная
1	10	$n \leqslant 5$		первая ошибка
2	15	$n \leqslant 10$	1	первая ошибка
3	20	$a < b$		первая ошибка
4	20	$a > b$		первая ошибка
5	15	$a = b$		первая ошибка
6	20	без дополнительных ограничений	1–5	первая ошибка

Примеры

стандартный ввод	стандартный вывод
5 2 2	No
7 1 0	Yes 1 7 6 5 4 3 2
7 3 2	Yes 1 3 2 5 4 7 6

Решение

Первая подзадача

Для решения на 5 баллов можно было ручками перебрать все перестановки.

Вторая подзадача

Для решения на 10 баллов можно было перебрать все перестановки и проверить, какие подходят.

Существование перестановки

Перестановки, в которой a элементов больше своих соседей и *b* элементов меньше своих соседей, существуют, когда абсолютная разность *a* и *b* не больше 1. Тоесть если $|a - b| > 1$, то не существует перестановки, удовлетворяющей условиям. Так же если *a+b+2 > n*, то

перестановки не существует, так как крайние элементы перестановки не могут быть больше или меньше двух соседей.

Случай a < b

Так как $|a - b| \leqslant 1$ и $a < b$, то $b = a + 1$.

Решим двойственную задачу: надо расставить $n - 1$ знак $<$ или $>$, так чтобы было a подстрок $<>$ и b подстрок $><$.

Тогда для решения данной подзадачи подходит паттерн $><><><\ldots$, но когда количество подстрок $<>$ ровно a и подстрок $><$ ровно b, на оставшиеся позиции надо поставить $<$.

Перейдем к перестановкам: паттерну $><><><\ldots$ удовлетворяет перестановка $2, 1, 4, 3, 6, 5, \ldots$, то есть перестановка $1, 2, 3, \ldots$, у которой на позициях $2 \cdot i$ и $2 \cdot i + 1$ поменяли элементы. Когда количество элементов больших своих соседей стало ровно a и элементов меньших своих соседей стало ровно b, то надо удовлетворять паттерну $<<<\ldots$. Данному паттерну удовлетворяет исходный суффикс перестановки.

Реализация на C++:

```
vector<int> p(n);
iota(p.begin(), p.end(), 1);
for(int i = 0; i + 1 < n && a >= 0 && b > 0; i += 2){
    swap(p[i], p[i + 1]);
    a--;
    b--;
}
```

Случай a > b

Аналогично $a < b$ решается данная подзадача, только уже подходит паттерн $<><><\ldots$. Данному паттерну удовлетворяет перестановка вида $1, 3, 2, 5, 4, \ldots$. Когда количество элементов больших своих соседей стало ровно a и элементов меньших своих соседей стало ровно b, то надо удовлетворять паттерну $>>>\ldots$. Данному паттерну удовлетворяет исходный суффикс перестановки, если его развернуть.

Реализация на C++:

```
vector<int> p(n);
iota(p.begin(), p.end(), 1);
int i = 1;
for(; i + 1 < n && a > 0 && b > 0; i += 2){
    swap(p[i], p[i + 1]);
    a--;
    b--;
}
i = min(n, i);
reverse(p.begin() + i, p.begin() + n);
```

Случай a = b

Для решения данной подзадачи можно использовать паттерн для $a < b$ и $a > b$, но когда количество элементов больших своих соседей стало ровно a и элементов меньших своих соседей стало ровно b, нужно аккуратно поставить элементы так, чтобы не появился новый элемент больший или меньший своих соседей.

Реализация на C++:

```
vector<int> p(n);
iota(p.begin(), p.end(), 1);
for(int i = 1; i + 1 < n && a > 0 && b > 0; i += 2){
    swap(p[i], p[i + 1]);
    a--;
    b--;
}
```

Задача C. Капли дождя...

Имя входного файла: стандартный ввод
Имя выходного файла: стандартный вывод
Ограничение по времени: 1 секунда
Ограничение по памяти: 256 мегабайт

В детстве Артем любил наблюдать за капельками на запотевшем окне. Иногда эти капельки устраивали очень увлекательные гонки!
Сейчас Артем уже не маленький мальчик, чтобы просто любоваться этими гонками. Сейчас он уже способен сказать, какая капелька придет к «финишу» первой, зная некоторую информацию об этих капельках.
А способны ли вы ответить на этот вопрос?
Формально процесс выглядит следующим образом: вам дана бесконечная плоскость (запотевшее окно), на этой плоскости располагается n точек (капельки воды).
Для каждой капельки известна следующая информация — *x, y, size* координаты капельки на окне и размер капельки.
Каждая капелька, в силу своей природы, старается течь вниз по стеклу. В нашей интерпретации мы будем говорить, что каждая капелька стремится уменьшить свою y координату до величины, равной 0.
За каждую единицу пройденного расстояния размер капельки может уменьшиться на *d* (это происходит в следствие того, что капелька оставляет за собой след на стекле).
Размер капельки уменьшается, если она движется не по чужому следу, а прокладывает свой.
Формально изменение, происходящее с положением капельки за секунду, можно выразить следующей схемой:

1. Если капелька достигла координаты *y* = 0, то она не двигается.
2. Капелька пытается изменить свою y координату на 1, не теряя в размере.
   Это возможно, если существовала некоторая другая капелька, которая уже прошла этим путем.

3. Капелька пытается изменить свою y координату на 1, оставляя за собой след, и уменьшается в размере на *d*.
   Это возможно, если размер капельки не меньше, чем *d*.
4. Если капелька не смогла переместиться согласно условиям пунктов выше, то она не двигается.

В том случае, если при перемещении капельки она оказывается в позиции, где есть другая капелька, тогда эти капельки сливаются, и их размеры складываются.
Для лучшего осознания процесса обратитесь к пояснениям в примере.
Артем попросил вас посчитать, через какое минимальное время найдется капелька, которая достигнет «финиша» — координаты y = 0. Если это никогда не произойдет — выведите −1.

Формат входных данных

В первой строке вводится два числа $1 \le n, d \le 10^5$ — изначальное количество капелек на плоскости и величина, на которую каждая капелька уменьшается при изменении своей *y* координаты.
В следующих *n* строках вводится информация про n капелек воды.
В i строке вводится три числа $1 \le x, y, size \le 10^9$ — позиция i капли в начальный момент времени и изначальный размер капли.
Гарантируется, что все пары x_i, y_i различны.

Формат выходных данных

Выведите минимальное время, через которое какая-нибудь из капелек достигнет координаты 0 или -1, если это никогда не произойдет.

Система оценки

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены.

Подзадача	Баллы	Ограничения	Необходимые подзадачи	Информация о проверке
0	–	примеры из условия		полная
1	15	$n \leqslant 100, x_i = 1$		первая ошибка
2	20	$n \leqslant 1000$	1	первая ошибка
3	30	$MAX(y_i) \leqslant 10^6$. Все $size_i$ — уникальны		первая ошибка
4	35	без дополнительных ограничений	1–3	первая ошибка

Примеры

стандартный ввод	стандартный вывод
3 1 1 2 1 1 3 2 1 4 1	3
3 1 1 2 1 1 3 1 1 4 1	-1
2 2 1 2 3 1 4 5	4

Замечание

Пояснение к примеру 3.

Нулевой момент времени:

- Первая капля имеет координаты (1, 2), а её размер равен 3
- Вторая капля имеет координаты (1, 4), а её размер равен 5

Первый момент времени:

- Первая капля уменьшает свою y координату на 1 и уменьшается в размере на 2.
- Первая капля имеет координаты (1, 1), а её размер равен 1.
- Вторая капля уменьшает свою y координату на 1 и уменьшается в размере на 2.
- Вторая капля имеет координаты (1, 3), а её размер равен 3

Второй момент времени:

- Первая капля не может уменьшить свою y координату на 1, вследствие недостаточности размера.
- Первая капля имеет координаты (1, 1), а её размер равен 1.
- Вторая капля уменьшает свою y координату на 1 и уменьшается в размере на 2.
- Вторая капля имеет координаты (1, 2), а её размер равен 1

Третий момент времени:

- Первая капля не может уменьшить свою y координату на 1, вследствие недостаточности размера.
- Первая капля имеет координаты (1, 1), а её размер равен 1.

- Вторая капля уменьшает свою y координату на 1, но не уменьшается в размере, ведь в момент времени 1 первая капля уже совершала такое изменение и оставляла свой след.
- Вторая капля попадает в координату с первой и их размеры сливаются.
- Объединенная капля имеет координаты (1, 1), а её размер равен 2.

Четвертый момент времени:

- Объединенная капля уменьшает свою y координату на 1 и уменьшается в размере на 2.
- Объединенная капля имеет координаты (1, 0), а её размер равен 0.
- Объединенная капля достигла финиша в момент времени 4.

Решение

Будем решать задачу по отдельности для каждой координаты X.
Утверждается, что некоторая капелька i может дойти до координаты $Y = 0$, тогда и только тогда, когда сумма размеров всех каелек, стоящих до неё больше либо равна чем высота (Y_i) тестируемой капельки.
Таким образом задача сводится к построению префикс сумм для каждой координаты X и проверке достижимости для каждой капельки по отдельности.

Задача D. Прыгающий робот

Имя входного файла: стандартный ввод
Имя выходного файла: стандартный вывод
Ограничение по времени: 1 секунда
Ограничение по памяти: 256 мегабайт

Дано натуральное число n ($1 \le n \le 2 \cdot 10^5$). Также дано два массива размера n: $a_1, \dots a_n$ ($1 \le a_i \le 10^9$) и $b_1, \dots b_n$ ($1 \le b_i \le 10^9$).
У вас есть робот, который может прыгать по массиву a. А именно, находясь на i-м элементе, он может прыгнуть вправо на ближайший элемент j ($i < j$) такой, что $a_j > a_i$. При этом, находясь на i-м элементе массива, робот должен заплатить b_i монет. Если он не может заплатить за очередной элемент в этой последовательности, робот не может совершить прыжок, и процесс останавливается.
Вы хотите q раз сделать следующее: поставить робота на позицию s_i ($1 \le s_i \le n$) и дать ему m_i ($1 \le m_i \le 10^{18}$) монет, после чего посмотреть, сколько прыжков он сможет сделать.
Ответьте на этот вопрос для разных значений s_i и m_i.

Формат входных данных

В первой строке задано натуральное число n ($1 \le n \le 2 \cdot 10^5$).

Во второй строке через пробел записан массив $a_1, \ldots a_n$ ($1 \le a_i \le 10^9$).В третьей строке через пробел записан массив $b_1, \ldots b_n$ ($1 \le b_i \le 10^9$). В четвертой строке записано натуральное число q ($1 \le q \le 2\ 10^5$).

В следующих q строках через пробел записано по два числа: в i-й строке записаны числа s_i ($1 \le s_i \le n$) и m_i ($1 \le m_i \le 10^{18}$).

Формат выходных данных

Выведите q строк: в i-й строке должно быть записано единственное число — ответ на i-й запрос.

Система оценки

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены.

Подзадача	Баллы	Ограничения	Необходимые подзадачи	Информация о проверке
0	–	примеры из условия		полная
1	10	$n, q \leqslant 1000$	—	первая ошибка
2	20	$a_i < a_{i+1}$ для всех $0 \leqslant i \leqslant n-1$	—	первая ошибка
3	30	$n \leqslant 5000$	1	первая ошибка
4	40	без дополнительных ограничений	1–3	первая ошибка

Пример

стандартный ввод	стандартный вывод
3 1 3 2 2 1 3 4 1 2 1 1 1 5 2 3	1 0 2 1

Решение

Будем отвечать на запросы в оффлайне. Для этого для каждого начального индекса найдем список запросов, которые ему соответствуют. Теперь пройдемся по всем индексам от большего к меньшему, поддерживая стек тех позиций, по которым робот может прыгать, начав с текущей позиции. А именно, первая позиция в этом стеке будет текущей, следующая

— первая такая, что для нее значение в массиве a больше, и так далее.

Допустим, робот начнет прыгать с текущей позиции. Тогда посещенные им позиции будут образовывать префикс позиций, хранимых сейчас на стеке. Причем это будет максимальный такой префикс, что сумма в массиве b по этим позициям не превосходит значения m для текущего запроса. Чтобы найти такой префикс, будем хранить в стеке вместе с позициями суффиксные суммы по массиву b. Имея эту информацию, максимальный префикс можно найти, используя бинпоиск.

Остается вопрос, как поддерживать такой стек. Допустим, для i-й позиции стек уже готов, попробуем получить его для $i - 1$-й позиции. Для этого заметим, что достаточно удалить несколько позиций из начала стека, для которых значение в массиве a не больше, чем значение для $i - 1$ позиции. После чего достаточно добавить в стек $i - 1$-й элемент и посчитать суффиксную сумму для текущей верхушки стека по формуле $suf_{cur} = suf_{prev} + b_{i-1}$.

Альтернативным решением может быть построение дерева по массиву a таким образом, что предок i-й вершины — это ближайший справа элемент j такой, что $a_j > a_i$ — то есть элемент, на который робот прыгнет с i-го элемента, если ему хватит денег. Также на каждой вершине напишем значение b_i — сколько робот должен заплатить за ее посещение. Тогда ответ на запрос (s_i, m_i) — это длина максимального пути вверх по дереву от вершины s_i такого, что сумма на этом пути не больше m_i. Это стандартная задача, решаемая с помощью техники бинарных подъемов.

Задача E. Плохое слово и путь в матрице

Имя входного файла: стандартный ввод

Имя выходного файла: стандартный вывод

Ограничение по времени: 1 секунда

Ограничение по памяти: 1024 мегабайта

Вы просто проснулись в теле хомяка, в левой верхней клетке матрицы размера n m, зная лишь, что получить обратно ваше имя и вашу память вы сможете только решив задачу ниже.

Хомяк хочет попасть из стартовой клетки — (1, 1), в клетку правого нижнего угла (n, m).

Хомяк может переходить в любую смежную с ним клетку. Клетки считаются смежными, если они имеют общую сторону.

Введем понятие строка пути — это такая строка, которую мы получим, если выпишем символы во всех клетках, в которых бывал хомяк, двигаясь по некоторому пути. Отметим, что символы нужно выписывать в том порядке, в котором хомяк посещал их, двигаясь по конкретному пути.

Назовем путь хорошим, если его строка пути не содержит плохое слово s как подстроку.

Вас просят найти кратчайший хороший путь из клетки (1, 1) в клетку (n, m), для заданного плохого слова s.

Формат входных данных

В первой строке входных данных находится три натуральных числа n, m, k — размеры матрицы и длина плохого слова s ($1 \le n\ m \le 10^5$, $1 \le k \le 10^2$).

В следующих n строках вводится матрица, состоящая из строчных букв латинского алфавита. В последней строке на вход подается плохое слово s, состоящее из строчных букв латинского алфавита.

Формат выходных данных

В единственной строке выведите ответ на задачу — минимальную длину хорошего пути. Если не существует хорошего пути — выведите -1.

Система оценки

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены.

Подзадача	Баллы	Ограничения	Необходимые подзадачи	Информация о проверке
0	–	примеры из условия		полная
1	30	Все символы строки s одинаковы.		первая ошибка
2	30	$n \cdot m \leqslant 10^3, k \leqslant 10$, s не содержит других символов кроме 'a', 'b'.		первая ошибка
3	40	без дополнительных ограничений	1–2	первая ошибка

Примеры

стандартный ввод	стандартный вывод
5 3 1 bbb aab bbb baa bbb a	10
1 1 1 f f	-1
1 8 4 bbbaabab bbaa	9

Решение

С виду задача напоминает классическую задачу на алгоритм поиска в ширину, так оно и есть! В данном случае нужно лишь добавить состояния, которые помогут нам гарантировать, что в процессе нашего пути мы не получим плохое слово, как подстроку.

Для этого я предлагаю добавить состояние вида — максимальный префикс плохой строки, который мы получим, если дойдем до некоторой клетки в матрице.

Таким образом матрица расстояний для бфса будет иметь вид $bfs[i][j][k]$ — кратчайшее расстояние от стартовой точки, до точки (i, j), если в процессе нашего пути мы не набрали плохое слово полностью и сейчас, стоя в клетке (i, j), мы набрали первые k символов от плохой строки.

Предполагалось, что эту информацию нужно будет пересчитывать с помощью автомата КМП построенного на префикс функции плохой строки, но в силу того, что ограничения на плохую строку были небольшими, это можно было сделать и с помощью предподсчета в лоб.