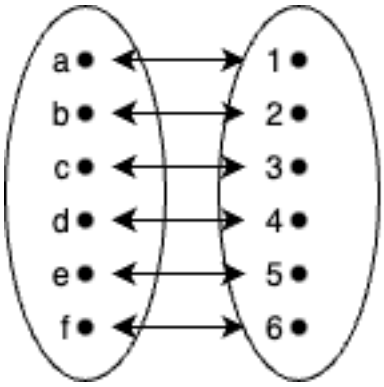
OSDA Homework 2

Andrei Mostachev

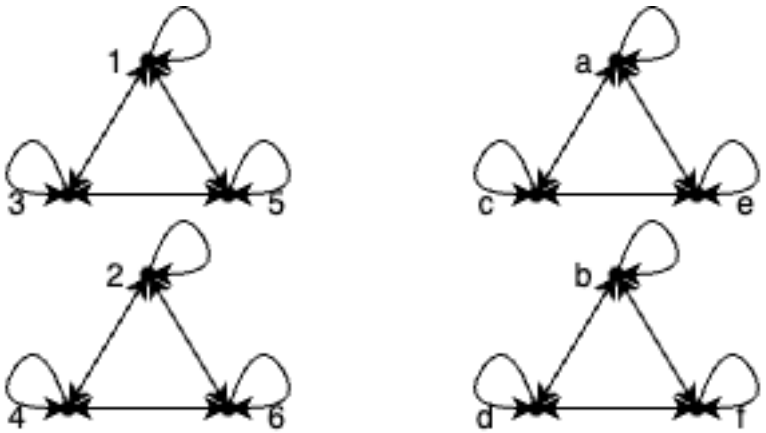
Problem 1

1.

Just looking at the given relations, it's simple to see that they are isomorphic, as the sum of numbers being even and two letters being an even number of letters apart are, essentially the same definition. We can more rigorously prove isomorphism by providing a bijective relation. In our case, this would be a simple mapping between a letter and its alphabetical number:



The distance between two letters is the difference between their alphabetical numbers, and in the case of R_1, the relation is defined by the sum of two numbers. The sum and the difference have the same parity, so the relations are isomorphic. We can also see this in their respective graphs:

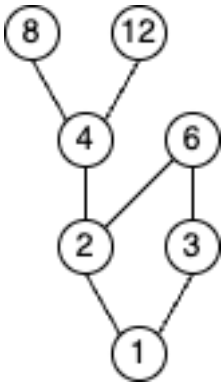


2.

The graphs representing the relations clearly have cycles. A topological sorting requires the graph to be acyclic, thus providing any such sorting for any of the given relations is impossible.

Problem 2

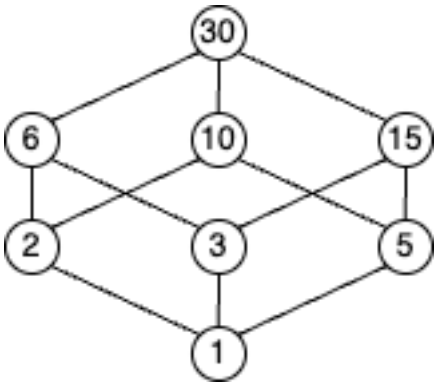
1.
Hasse diagram:



2.
Three antichains: $\{2, 3\}, \{4, 6\}, \{8, 12\}$.
3.
Three ideals: $\{1, 2, 4, 8\}, \{1, 2, 4, 12\}, \{1, 2, 3, 6\}$.
4.
Three filters: $\{4, 8, 12\}, \{1, 2, 3, 4, 6, 8, 12\}, \{2, 4, 6, 8, 12\}$.

Problem 3

1.
Hasse diagram:



2.
We have two obvious "horizontal"antichains in the Hasse diagram. They indicate that the dimension is no less than 3. Let's construct 3 strictly ordered sets which when combined produce the given relation:

$\{1, 2, 3, 5, 6, 10, 15, 30\}$

$\{1, 5, 3, 15, 2, 10, 6, 30\}$

$\{1, 2, 10, 3, 6, 5, 15, 30\}$

Problem 4

The Python function to fit and compute the metrics:

```
def preprocess_and_train(in_df, model, target_column, max_nans_cutoff=0.15):
  models = {
      'Decision tree' : (DecisionTreeClassifier(),
       {'max_depth': [2, 4, 6, 8],
       'min_samples_split': [2, 4, 6, 8],
       'min_samples_leaf': [1, 2, 3, 4]}),
      'Random forest' : (RandomForestClassifier(),
       {'max_depth': [2, 4, 8],
       'min_samples_split': [2, 4, 8],
       'min_samples_leaf': [1, 2, 4],
       'max_features' : ['sqrt', 'log2']}),
      'XGBoost' : (XGBClassifier(objective= 'binary:logistic'),
       {'max_depth' : range (2, 10, 1),
       'n_estimators': range(60, 220, 40),
       'learning_rate': [0.1, 0.01, 0.05]}),
      'Catboost' : (CatBoostClassifier(),
       {'depth' : [4, 6, 8], 'learning_rate' : [0.01, 0.02, 0.04],
       'iterations' : [10, 20, 80]}),
      'K-NN' : (KNeighborsClassifier(),
       {'n_neighbors' : [2, 4, 8, 16]}),
      'Naive Bayes' : (BernoulliNB(),
       {'alpha': [0.01, 0.1, 0.5, 1.0], 'fit_prior': [True, False],
       'binarize': [None, 1.0, 10.0]})
      }

  if model not in models.keys():
    raise Exception('Unknown model!')

  df = in_df.copy()

  if model in ['Random forest', 'K-NN', 'Naive Bayes']:
    dropcols = []
    for c in df.columns:
      if df[c].isna().sum() > len(df) * max_nans_cutoff:
        dropcols.append(c)

    df.drop(columns=dropcols, inplace=True)

    df.dropna(inplace=True)

  def custom_combiner(feature, category):
    return str(feature) + "_" + str(category)

  obj_columns = df.columns[df.dtypes == object]
```

```
custom_fnames_enc = OneHotEncoder(feature_name_combiner=custom_combiner)\
.fit(df[obj_columns])
custom_fnames_enc.get_feature_names_out()
out_df = pd.DataFrame(custom_fnames_enc.transform(df[obj_columns])\
.toarray(), columns=custom_fnames_enc.get_feature_names_out())

df.reset_index(inplace=True)

df = out_df.join(df)
df.drop(columns=obj_columns, inplace=True)

X_train, X_test, y_train, y_test = train_test_split(df[df.columns[:-1]],
  df[target_column], test_size=0.3, random_state=42,
  stratify=df[target_column])

print('Dataset ready. Proceeding to training')

model, params = models[model]

clf = copy.copy(model)

search = GridSearchCV(clf, params, cv=5, scoring='f1')
search.fit(X_train, y_train)

print('Train complete. Calculating metrics')

clf = copy.copy(model)
clf.set_params(**search.best_params_)

clf = clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

return (metrics.f1_score(y_test, y_pred), metrics.accuracy_score(y_test, y_pred),
  search.best_params_)
```

Results:

F1-scores:

	Decision tree	Random forest	XGBoost	Catboost	K-NN	Naive Bayes
Rain in Australia	0.480977	0.535951	0.647652	0.582836	0.327680	0.484288
Drinking habit	0.707955	0.717955	0.735352	0.731852	0.424936	0.668646
Motorcycle use	0.912281	0.854701	0.913793	0.896552	0.787879	0.796460

Accuracy scores:

	Decision tree	Random forest	XGBoost	Catboost	K-NN	Naive Bayes
Rain in Australia	0.829933	0.847609	0.862467	0.850933	0.804153	0.766390
Drinking habit	0.703867	0.714533	0.730800	0.728133	0.507400	0.665133
Motorcycle use	0.870130	0.779221	0.870130	0.844156	0.727273	0.701299