In [ ]:

```python
import pandas as pd
import numpy as np

#Max Social Link of Tohru Adachi
MAX_ADACHI = 6
ADACHI_DAY_LEVELS = np.array([1,2,5,6])-1

def setup():
  # Data loading and setting up
  df = pd.read_csv('d.csv')
  availabilities = df[df.columns[2]].astype(str)
  months = df[df.columns[0]].ffill().astype(str)
  date = df[df.columns[1]].astype(np.uint8)
  df = df.drop(df.columns[range(3)], axis=1)
  data = df.to_numpy().astype(np.uint8)

  # SLink mat setup
  #Day Characters
  sl_max = np.ones((19,)) * 10
  #Tohru Adachi
  adachi = df.columns.get_loc('Tohru Adachi')
  sl_max[adachi] = MAX_ADACHI
  #Marie
  sl_max[9] = 4
  #Eri Minami
  eri = df.columns.get_loc('Eri Minami')
  sl_max[eri] = 14
  #Daisuke Nagase
  daisuke = df.columns.get_loc('Daisuke Nagase')
  sl_max[daisuke] = 12
  #Yumi Ozawa
  yumi = df.columns.get_loc('Yumi Ozawa')
  sl_max[yumi] = 11
  #Ai Ebihara (only 9 lvl because lvl 1 is automatic)
  ai = df.columns.get_loc('Ai Ebihara')
  sl_max[ai] = 9
  #Naoki Konishi
  naoki = df.columns.get_loc('Naoki Konishi')
  sl_max[naoki] = 11

  #Night Characters
  #Nanako
  nanako = df.columns.get_loc('Nanako Dojima')
  sl_max[nanako] = 16
  #Ryotaro
  ryotaro = df.columns.get_loc('Ryotaro Dojima')
  sl_max[ryotaro] = 13
  #Sayoko
  sayoko = df.columns.get_loc('Sayoko Uehara')
  sl_max[sayoko] = 14

  mat = np.diag(sl_max).astype(np.uint8)

  listrange = lambda x,y: list(range(x, y))
  day_chars = [*listrange(0, 2), *listrange(3,12), *listrange(14, 17), 18]
  night_chars = sorted(list(set(listrange(0,19))-set(day_chars)))
  return {'availabilities': availabilities, 'mat': mat, 'months': months, 'date': date,
'data': data, 'day_chars': day_chars, 'night_chars': night_chars, 'df': df}
```

Weights ¶

In [ ]:

```
def setup_all():
  weights = np.ones((19,))
  return weights
```

Simulation

In [ ]:

```
def process_adachi(level, idx, night_chars, day_chars):
    #print(night_chars, day_chars)
    if level in ADACHI_DAY_LEVELS:
        if idx in night_chars:
            night_chars.remove(idx)
            day_chars.append(idx)
    else:
        if idx in day_chars:
            night_chars.append(idx)
            day_chars.remove(idx)
    return night_chars, day_chars

def iterate(r: np.ndarray, data: np.ndarray, mat: np.ndarray, night_chars, day_chars):
    s = data.sum(axis=0)
    r = np.divide(r, s, out=np.zeros_like(r).astype(np.float32), where=s!=0)
    res = mat@r
    maxdata = np.argmax(res)
    if res[maxdata] <= 0: return mat, None, night_chars, day_chars
    curr = mat[maxdata, maxdata]
    if curr <= 0: return mat, None, night_chars, day_chars
    mat[maxdata, maxdata] = curr - 1
    adachi = 11
    if maxdata == adachi:
        night_chars, day_chars = process_adachi(MAX_ADACHI-curr+1, adachi, night_chars, 
day_chars)
    return mat, maxdata, night_chars, day_chars

def simulate(data, mat, weights, availabilities, night_chars, day_chars, **kwargs):
    selecteds = []
    for i, r in enumerate(data.copy()):
        r = r * weights
        if i == 105:
            mat[9, 9] = 6 + mat[9, 9]
        if availabilities[i] == 'b':
            a = r.copy()
            a[night_chars] = 0
            d = data[i:].copy()
            d[:, night_chars] = 0
            mat, selected1, night_chars, day_chars = iterate(a, d, mat, night_chars, da
y_chars)
            a = r.copy()
            a[day_chars] = 0
            a[selected1] = 0
            d = data[i:].copy()
            d[:, day_chars] = 0
            mat, selected2, night_chars, day_chars = iterate(a, d, mat, night_chars, da
y_chars)
            selected = (selected1, selected2)
        else:
            mat, selected, night_chars, day_chars = iterate(r, data[i:], mat, night_cha
rs, day_chars)
            selected = (selected, None)
        selecteds.append(selected)
    return selecteds, np.diag(mat)
```

Find weights

In [ ]:

```
weights = setup_all()
for i in range(1000):
  data = setup()
  data['weights'] = weights
  _, res = simulate(**data)
  if (res == 0).all():
    break
  weights[np.where(res > 0)] += .05
print(f'iteration needed: {i}')
print(f'final sl residue: {res}')
print(f'final sl residue names: {data["df"].columns[res > 0]}')
print(f'weights: {weights}')
```

```
iteration needed: 0
final sl residue: [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
final sl residue names: Index([], dtype='object')
weights: [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
```

Final simulation and Save result

In [ ]:

```
data = setup()
data['weights'] = weights
df = data['df']
date = data['date']
months = data['months']
selecteds, res = simulate(**data)
print(res)
print(data['df'].columns[res > 0])
names = df.columns.to_numpy().copy()
none = names.shape[0]
names = np.append(names, 'None')
p = []
for i, (d, m) in enumerate(zip(date, months)):
    s1, s2 = selecteds[i]
    if s1 is None: s1 = none
    if s2 is None:
        p.append(names[s1])
    else:
        p.append(f'Day: {names[s1]}, Night: {names[s2]}')

import csv
with open('Final Result.csv', 'w') as f:
    writer = csv.writer(f, lineterminator='\n')
    writer.writerows(zip(months,date,p))
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
Index([], dtype='object')
```