In [1]:

```
import random
from pprint import pprint

import numpy as np

from utils import read_data
from constants import FILENAME_EXAMPLE_1, FILENAME_EXAMPLE_4
```

In [2]:

```
#input_data = read_data(FILENAME_EXAMPLE_1)
input_data = read_data(FILENAME_EXAMPLE_4)

days     = input_data['length_of_schedule']
ne       = input_data['number_of_employees']
ns       = input_data['number_of_shifts']
demand   = input_data['temporal_requirements_matrix']
sn       = input_data['shift_name']
ss       = input_data['start_shift']
ls       = input_data['length_shift']
min_ls   = input_data['min_length_of_blocks']
max_ls   = input_data['max_length_of_blocks']
min_do   = input_data['min_days_off']
max_do   = input_data['max_days_off']
min_lw   = input_data['min_length_work_blocks']
max_lw   = input_data['max_length_work_blocks']
nf2      = input_data['nr_sequences_of_length_2']
nf3      = input_data['nr_sequences_of_length_3']
f2       = input_data['not_allowed_shift_sequences_2']
f3       = input_data['not_allowed_shift_sequences_3']
```

In [3]:

```
shifts = ns + 1
day, afternoon, night, dayoff = 1, 2, 3, 4
code = sn + ['-']
```

In [4]:

```
def calculate_result(input_data):
    return [[random.choice(code) for d in range(days)] for e in range(ne)]
```

In [5]:

```
def demand_constraint(result):
    for e in range(ne):
        for d in range(days):
            sum_day, sum_afternoon, sum_night = 0, 0, 0
            if result[e][d] == 'D': sum_day += 1
            elif result[e][d] == 'A': sum_afternoon += 1
            elif result[e][d] == 'N': sum_night += 1
        if sum_day >= demand[0][d] and sum_afternoon >= demand[1][d] and sum_nig
            pass
        else:
            return False
    return True
```

In [6]:

```
def update_demand_constraint(result):
    updated_result = result
    for d in range(days):
        sum_day, sum_afternoon, sum_night = 0, 0, 0
        for e in range(ne):
            if updated_result[e][d] == 'D': sum_day += 1
            elif updated_result[e][d] == 'A': sum_afternoon += 1
            elif updated_result[e][d] == 'N': sum_night += 1

        if sum_day >= demand[0][d]: pass
        else: updated_result[random.randint(0, ne-1)][d] = 'D'
        if sum_afternoon >= demand[1][d]: pass
        else: updated_result[random.randint(0, ne-1)][d] = 'A'
        if sum_night >= demand[2][d]: pass
        else: updated_result[random.randint(0, ne-1)][d] = 'N'
    return updated_result
```

In [7]:

```
#day off constraint
def day_off_constraint(result):
    for e in range(ne):
        count_dayoff = 0
        for d in range(days):
            if result[e][d] == '-': count_dayoff += 1
        if min_do <= count_dayoff <= max_do: pass
        else: return False
    return True
```

In [8]:

```
off constraint
pdate_day_off_constraint(result):
pdated_result = result
for e in range(ne):
    count_dayoff = 0
    for d in range(days):
        if updated_result[e][d] == '-': count_dayoff += 1
    if count_dayoff >= min_do: pass
    else: updated_result[e][random.randint(0, days-1)] = '-'
    if count_dayoff <= max_do: pass
    else: updated_result[e][random.randint(0, days-1)] = random.choice(list(set(co
return updated_result
```

In [9]:

```
#working days in a row constraint
def length_work_blocks_constraint(result):
    for e in range(ne):
        count_consecutive = 0
        min_flag, max_flag = False, False
        for d in range(days - 1):
            if result[e][d] != '-' and result[e][d+1] != '-':
                count_consecutive += 1
                if count_consecutive >= min_lw - 1:
                    min_flag = True
                    if count_consecutive <= max_lw - 1:
                        max_flag = True
            else:  count_consecutive = 0
        if min_flag and max_flag: pass
        else: return False
    return True
```

In [10]:

```
#working days in a row constraint
def update_length_work_blocks_constraint(result):
    updated_result = result
    for e in range(ne):
        count_consecutive = 0
        min_flag, max_flag = False, False
        for d in range(days - 1):
            if updated_result[e][d] != '-' and updated_result[e][d+1] != '-':
                count_consecutive += 1
                if count_consecutive >= min_lw - 1:
                    min_flag = True
                    if count_consecutive <= max_lw - 1:
                        max_flag = True
            else:  count_consecutive = 0

        if min_flag: pass
        else:
            count_concecutive = 0
            for d in range(days):
                if updated_result[e][d] == '-' and count_concecutive <= max_lw -
                    updated_result[e][d] = random.choice(list(set(code) - set('-
                count_concecutive += 1
        if max_flag: pass
        else: updated_result[e][random.randint(0, days-1)] = '-'

    return updated_result
```

In [11]:

```
#forbidden shifts constraint
def forbidden_constraint2(result):
    if f2 == []: return True
    if f2 != []:
        for e in range(ne):
            for d in range(days - 1):
                for f in f2:
                    if result[e][d] == f[0] and result[e][d+1] == f[1]: return F
        return True
```

In [89]:

```
#forbidden shifts constraint
def update_forbidden_constraint2(result):
    updated_result = result
    if f2 == []: return updated_result
    if f2 != []:
        for e in range(ne):
            for d in range(days - 1):
                for f in f2:
                    if updated_result[e][d] == f[0] and updated_result[e][d+1] =
                        for c in code:
                            if [f[0], c] not in f2:
                                updated_result[e][d+1] = c
                                break
        return updated_result
```

In [87]:

```
#forbidden shifts constraint
def update_forbidden_constraint2_2(result):
    updated_result = result
    if f2 == []: return updated_result
    if f2 != []:
        for e in range(ne):
            for d in range(days - 1):
                for f in f2:
                    if updated_result[e][d] == f[0] and updated_result[e][d+1] =
                        for c in code:
                            if [c, f[1]] not in f2:
                                updated_result[e][d] = c
                                break
        return updated_result
```

In [14]:

```
f2
```

```
[['N', 'D'], ['N', 'A'], ['A', 'D']]
```

In [15]:

```
def forbidden_constraint3(result):
    if f3 == []: return True
    if f3 != []:
        for e in range(ne):
            for d in range(days - 2):
                for f in f3:
                    if result[e][d] == f[0] and result[e][d+1] == f[1] \
                                            and result[e][d+2] == f[2]: return F
        return True
```

In [86]:

```
def update_forbidden_constraint3(result):
    updated_result = result
    if f3 == []: return updated_result
    if f3 != []:
        for e in range(ne):
            for d in range(days - 2):
                for f in f3:
                    if result[e][d] == f[0] and result[e][d+1] == f[1] \
                                            and result[e][d+2] == f[2]:
                        for c in code:
                            if [c, f[1], f[2]] not in f3:
                                updated_result[e][d] = c
                                break

        return updated_result
```

In [85]:

```
def update_forbidden_constraint3_2(result):
    updated_result = result
    if f3 == []: return updated_result
    if f3 != []:
        for e in range(ne):
            for d in range(days - 2):
                for f in f3:
                    if result[e][d] == f[0] and result[e][d+1] == f[1] \
                                            and result[e][d+2] == f[2]:
                        for c in code:
                            if [f[0], c, f[2]] not in f3:
                                updated_result[e][d] = c
                                break

        return updated_result
```

In [20]:

```
def calculate_score(solution):
    score = 0
    c1, c2, c3, c4, c5 = demand_constraint, \
                         day_off_constraint, \
                         length_work_blocks_constraint, \
                         forbidden_constraint2, \
                         forbidden_constraint3
    if c1(solution):
        score += 50
    if c2(solution):
        score += 15
    if c3(solution):
        score += 15
    if c4(solution):
        score += 10
    if c5(solution):
        score += 10

    return score
```

In [125]:

```
def solve_problem(input_data):
    score = 100
    result = calculate_result(input_data)
    c1, c2, c3, c4, c5 = demand_constraint, \
                         day_off_constraint, \
                         length_work_blocks_constraint, \
                         forbidden_constraint2, \
                         forbidden_constraint3
    while calculate_score(result) < score:
        if calculate_score(result) < score:
            result = calculate_result(input_data)

        if not c5(result):
            result = update_forbidden_constraint3(result)
            if calculate_score(result) >= score: break
        if not c5(result):
            result = update_forbidden_constraint3_2(result)
            if calculate_score(result) >= score: break
        if not c4(result):
            result = update_forbidden_constraint2(result)
            if calculate_score(result) >= score: break
        if not c4(result):
            result = update_forbidden_constraint2_2(result)
            if calculate_score(result) >= score: break
        if not c3(result):
            result = update_length_work_blocks_constraint(result)
            if calculate_score(result) >= score: break
        if not c2(result):
            result = update_day_off_constraint(result)
            if calculate_score(result) >= score: break
        if not c1(result):
            result = update_demand_constraint(result)
            if calculate_score(result) >= score: break

    return result
```

In [22]:

```
def test_contraints(solution):
    return {
            'demand_constraint': demand_constraint(solution),
            'day_off_constraint': day_off_constraint(solution),
            'length_work_blocks_constraint': length_work_blocks_constraint(solut
            'forbidden_constraint2': forbidden_constraint2(solution),
            'forbidden_constraint3': forbidden_constraint3(solution)
            }
```

In [126]:

```
solution = solve_problem(input_data)
solution
```

```
[['-', 'D', 'N', 'N', 'N', 'N', '-'],
 ['-', 'N', 'N', 'N', 'N', 'N', 'N'],
 ['-', 'D', '-', '-', 'A', 'N', 'N'],
 ['A', 'A', 'A', 'A', 'A', '-', 'N'],
 ['A', 'A', 'N', 'N', '-', '-', '-'],
 ['D', 'D', '-', 'N', 'N', 'N', 'N'],
 ['-', '-', 'D', '-', 'A', 'N', 'N'],
 ['A', 'A', 'N', 'N', '-', '-', '-'],
 ['-', '-', '-', 'A', 'N', 'N', 'N'],
 ['-', 'D', '-', 'N', 'N', 'N', 'N'],
 ['-', 'N', 'N', 'N', 'N', 'N', 'N'],
 ['D', '-', 'A', 'N', 'N', 'N', 'N'],
 ['-', 'A', 'A', 'N', 'N', 'N', '-']]
```

In [127]:

```
calculate_score(solution)
```

```
100
```

In [128]:

```
pprint(test_contraints(solution))
```

```
{'day_off_constraint': True,
 'demand_constraint': True,
 'forbidden_constraint2': True,
 'forbidden_constraint3': True,
 'length_work_blocks_constraint': True}
```

In [62]:

```
res_dayoff_true = update_day_off_constraint(solution)
```

In [90]:

```
res_work_blocks = update_length_work_blocks_constraint(res_dayoff_true)
```

In [98]:

```
res_f2 = update_forbidden_constraint2(res_work_blocks)
```

In [99]:

```
pprint(test_contraints(res_f2))
```

```
{'day_off_constraint': True,
 'demand_constraint': True,
 'forbidden_constraint2': True,
 'forbidden_constraint3': True,
 'length_work_blocks_constraint': True}
```

In [96]:

```
res_f3 = update_forbidden_constraint3_2(res_f2)
```

In [97]:

```
pprint(test_contraints(res_f3))
```

```
{'day_off_constraint': True,
 'demand_constraint': True,
 'forbidden_constraint2': True,
 'forbidden_constraint3': True,
 'length_work_blocks_constraint': True}
```

In [95]:

```
res_f3
```

```
[['-', '-', 'D', '-', 'D', 'D', 'D'],
 ['D', 'D', 'D', 'D', '-', 'A', 'A'],
 ['A', 'A', 'N', '-', '-', 'D', 'D'],
 ['A', 'N', 'N', 'N', 'N', 'N', '-'],
 ['A', '-', '-', 'D', 'D', 'A', 'N'],
 ['-', '-', 'D', 'D', 'D', 'D', 'A'],
 ['D', 'D', 'D', '-', 'D', 'A', '-'],
 ['D', 'D', 'D', 'D', 'D', '-', 'A'],
 ['N', 'N', '-', '-', 'D', 'D', 'A'],
 ['D', 'D', '-', '-', 'D', 'D', 'A'],
 ['-', '-', 'D', 'D', 'D', 'D', 'D'],
 ['D', 'D', 'D', 'D', '-', 'D', 'N'],
 ['-', '-', 'D', 'D', 'A', 'A', 'A']]
```