بنام خدا

۱. در این مسئله با توجه به داده های داده شده و به کمک روش m بزرگ یک مسئله assignment تعریف کردم که در ان همیشه سعی میشود هزینه کاهش یابد.

```
def read_file(path):
    print(path)

    file = open(path, 'r')
    for line in file:
        flight_numbers, worker_numbers = line.split()
        break

    df = pd.read_csv(path, sep=' ', header=None, skiprows=1)
    df.head()
    costs=[]
    for item in df.to_numpy():
      item_costs=[]
      for task in item:
        if task ==1:
          item_costs.append(0)
        else:
          item_costs.append(100)
      costs.append(item_costs)
    num_workers = int(worker_numbers)
    num_tasks = int(flight_numbers)
    return costs,num_workers,num_tasks,path
```

در مرحله بعد به کمک ortools و تیکه کد زیر مسئله را حل نمودم:

```python
def optimize(costs,num_workers,num_tasks,path):
    solver = pywraplp.Solver.CreateSolver('SCIP')

    if not solver:
        print('solver init error')
    # x[j, i] is an array of 0-1 variables, which will be 1
    # if worker j is assigned to flight i.
    x = {}
    for i in range(num_workers):
        for j in range(num_tasks):
            x[i, j] = solver.IntVar(0, 1, '')
    # Each worker is assigned to at most 1 task.
    for i in range(num_workers):
        solver.Add(solver.Sum([x[i, j] for j in range(num_tasks)]) == 1)

    # Each task is assigned to exactly one worker.
    for j in range(num_tasks):
        solver.Add(solver.Sum([x[i, j] for i in range(num_workers)]) >= 1)
    objective_terms = []
    for i in range(num_workers):
        for j in range(num_tasks):
            objective_terms.append(costs[j][i] * x[i, j])
    solver.Minimize(solver.Sum(objective_terms))
    status = solver.Solve()
    result=[]
    if status == pywraplp.Solver.OPTIMAL or status == pywraplp.Solver.FEASIBLE:
        for i in range(num_workers):
            for j in range(num_tasks):
                # Test if x[i,j] is 1 (with tolerance for floating point arithmetic).
                if x[i, j].solution_value() > 0.5:
                    result.append((j,i,-1 if costs[j][i]==100 else 1))
                    pass
        res_df=pd.DataFrame(result,columns=['Flight','Worker','Assigned'])
        # display(res_df)
        assign_res=[]
        for item in res_df.groupby('Flight').sum()['Assigned'].to_numpy():
          if item<0:
            assign_res.append(-1)
          else:
            assign_res.append(item)
        print(assign_res)
```

در اینجا محدودیت ها عبارتند از:

- هر کارگر فقط در یک هواپیما کار میکند
- هر هواپیما حداقل یک کارمند دارد

متغییر های مدلسازی عبارتند از:

- X برای کار گر و پرواز (درصورتی که گارگری روی پروازی کار کند ۱ و در غیر اینصوزت ۰)

نمونه خروجی کار را در ادامه میبینیم:

```
01
[2, 1, -1]
02
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1]
03
[-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1]
04
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1]
```

۲. در سوال دوم هدف قرار دادن نمودار ها در کمترین فضا با محدود کردن تعداد برخورد بود

برای مدل سازی این مسئله از الگوریتم Bin Packing استفاده شد و هر فضا را به یک کوله پشتی و هر نمودار را به یک ایتم مدل کردیم با این فرض که ارزش هر نمودار ۱ است

```python
def create_data_model(n,all_ver):
    """Create the data for the example."""
    data = {}
    weights = np.ones(n)
    data['weights'] = weights
    data['items'] = list(range(len(weights)))
    data['bins'] =data['items']
    data['has_conflicts']=vers_conflict(all_ver)
    data['bin_capacity'] = n
    # print(data['has_conflicts'])
    return data
```

همینطور برای مدل کردن برخوردها یک لیست با ایندکس متناظر نمودار ها در نظر گرفتیم

که این لیست به کمک توابع زیر و مفاهیم صعودی و نزولی و تساوی بودن دو گراف پیاده سازی شد.

```python
def vers_conflict(vers):
  res={}
  for i in range(len(vers)):
    new_range=list(range(len(vers))).copy()
    new_range.remove(i)
    for j in new_range:
      item1=vers[i]
      item2=vers[j]
      hc=has_conflict(item1,item2)[1]
      if(hc==False and (j,i) not in res.keys()):
        res[(i,j)]=0
      elif((j,i) not in res.keys()):
        res[(i,j)]=1

  return res
```

```python
def has_conflict(a,b):
  state=[]
  k=len(a)
  for index in range(k):
    if(a[index]==b[index]):
      state.append(0)
    elif(a[index]>b[index]):
      state.append(1)
    elif(a[index]<b[index]):
      state.append(2)
  bool_state=[]
  for i in range(len(state)):
    bool_state.append((i,(state[0]==state[i])))
  res=None
  for item in bool_state:
    if item[1]==False:
      res=item[0]

  return res,res!=None
```

در مرحله بعد به سراغ تعریف مدل ریاضی و پیاده سازی آن رفتم

محدودیت ها:

- هر نمودار فقط در یک فضا وجود دارد
- هر فضا حداقل یک نمودار دارد
- دو نمودار با برخورد در یک فضا نباشند

متغییر ها:

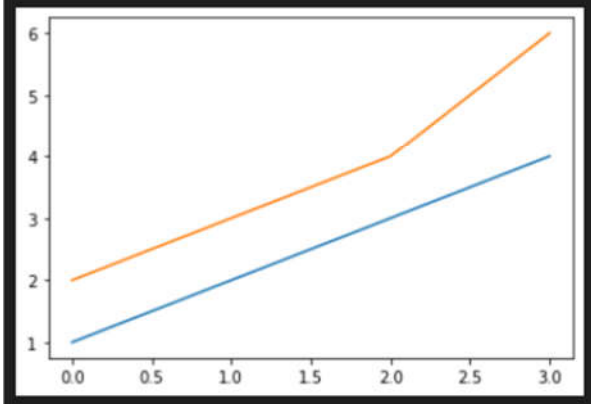
- X برای وجود و یا عدم وجود نمودار در یک فضا
- Y برای خالی بودن یا نبودن یک فضا

در ادامه بخش مهم کار که مربوط به چک کردن برخورد است اوردم:

```
# conflicts constraint
for bin in data['bins']:
  bin_items_index=[]
  for value in x.keys():
    if value[1]==bin:
      bin_items_index.append(value[0])
      pass
    pass
  for i1 in bin_items_index:
    new_items=bin_items_index.copy()
    new_items.remove(i1)
    for i2 in new_items:
      if ((i1,i2)  in data['has_conflicts'].keys()):
        if(data['has_conflicts'][(i1,i2)]==1):
          # print('conflict')
          solver.Add(sum(x[index,bin] for index in [i1,i2])<=1)
      pass
    pass
  pass
pass
```

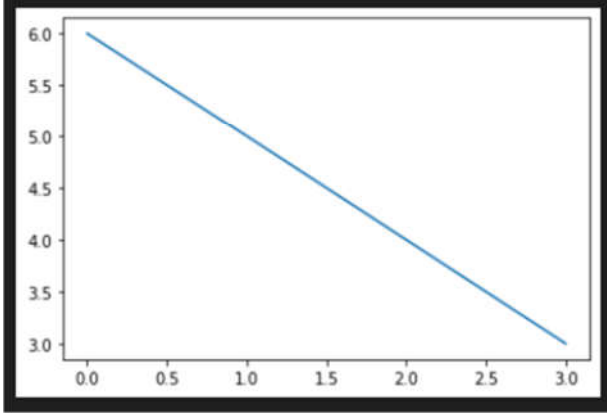
نمونه ای از نتیجه کار در ادامه اورده شده است:

```
01
Bin number 0
  Items packed: [array([1, 2, 3, 4]), array([2, 3, 4, 6])]
```




```
  Total weight: 2.0

Bin number 1
  Items packed: [array([6, 5, 4, 3])]
```



دراین تمرین:

- از لایبرری numpy برای عملیات روی ارایه ها
- از لایبرری pandas برای خواندن داده ها
- از لایبرری matplotlib برای رسم نمودار ها
- و از لایبرری ortools برای حل مسائل بهینه سازی
- از google colabratory برای اجرای کد

استفاده نمودم

فایل کدها و دیتاها به پیوست و بصورت فشرده ارسال میگردد.

همچنین کد ها در ادرس های زیر موجود است.

- https://colab.research.google.com/drive/1uyCQl6HOyn1HqQTbOfH0uDgUNJ9ujX73
- https://colab.research.google.com/drive/1jkCOpA32sIsDm3UWc-Wss8EIeEQMwI63

باتشکر

امیرحسن امیرماهانی