Sample Midterm 2023_YB

October 7, 2024

```
[1]: ## 2023 - Q1. Ordering Kitchen Cabinets (7 Points)
```

```
[31]: instock={'A': [1,3,8,9,14,18,30,31,60,61,80,90,100],'B':␣
        ↪[5,6,7,20,21,40,70,71,100],'C': [1,2,9,10,25,50,90,91,100],'D':␣
        ↪[3,4,10,11,29,35,59,81,100]}
      delivery_time={'A':3,'B':5,'C':7,'D':4}
      today=21
```

```
[173]: earliest_order_date_A = max(instock['A']) + 1
       earliest_order_date_B = max(instock['B']) + 1
       earliest_order_date_C = max(instock['C']) + 1
       earliest_order_date_D = max(instock['D']) + 1

       delivery_list = [0] * len(instock)

       for cabinet, dates in instock.items():
           for date in dates:
               if cabinet == 'A' and date >= today and date < earliest_order_date_A:
                   earliest_order_date_A = date
                   earlist_delivery_date_A = earliest_order_date_A + delivery_time['A']
                   delivery_list[0] = earlist_delivery_date_A
                   print(cabinet, earliest_order_date_A)
               if cabinet == 'B' and date >= today and date < earliest_order_date_B:
                   earliest_order_date_B = date
                   earlist_delivery_date_B = earliest_order_date_B + delivery_time['B']
                   delivery_list[1] = earlist_delivery_date_B
                   print(cabinet, earliest_order_date_B)
               if cabinet == 'C' and date >= today and date < earliest_order_date_C:
                   earliest_order_date_C = date
                   earlist_delivery_date_C = earliest_order_date_C + delivery_time['C']
                   delivery_list[2] = earlist_delivery_date_C
                   print(cabinet, earliest_order_date_C)
               if cabinet == 'D' and date >= today and date < earliest_order_date_D:
                   earliest_order_date_D = date
                   earlist_delivery_date_D = earliest_order_date_D + delivery_time['D']
                   delivery_list[3] = earlist_delivery_date_D
                   print(cabinet, earliest_order_date_D)
       print(delivery_list)
       earliest_delivery_date = max(delivery_list)
       print(earliest_delivery_date)
```

```
A 30
B 21
C 25
D 29
[33, 26, 32, 33]
33
```

[193]:
```
instock={'A': [1,3,8,9,14,18,30,31,60,61,80,90,100],'B':␣
↪[5,6,7,20,21,40,70,71,100],'C': [1,2,9,10,25,50,90,91,100],'D':␣
↪[3,4,10,11,29,35,59,81,100]}
delivery_time={'A':3,'B':5,'C':7,'D':4}
today=21

earliest_deliver_dates = []

cabinet_list = list(instock.keys())

for i in range(len(cabinet_list)):
    cabinet = cabinet_list[i]
    order_dates_list = instock[cabinet]
    for dates in order_dates_list:
        if dates >= today:
            earliest_deliver_dates.append(dates + delivery_time[cabinet])
            break
    print(cabinet)
    print(earliest_deliver_dates)
```

```
A
[33]
B
[33, 26]
C
[33, 26, 32]
D
[33, 26, 32, 33]
```

[6]:
```
# My code #1

def earliest(instock, delivery_time, today):
    earliest_order_date_A = float('inf')
    earliest_order_date_B = float('inf')
    earliest_order_date_C = float('inf')
    earliest_order_date_D = float('inf')
    delivery_list = [0] * len(instock)

    for cabinet, dates in instock.items():
        for date in dates:
            if cabinet == 'A' and date >= today and date <␣
↪earliest_order_date_A:
                earliest_order_date_A = date
                earlist_delivery_date_A = earliest_order_date_A +␣
↪delivery_time['A']
                delivery_list[0] = earlist_delivery_date_A
            if cabinet == 'B' and date >= today and date <␣
↪earliest_order_date_B:
                earliest_order_date_B = date
                earlist_delivery_date_B = earliest_order_date_B +␣
↪delivery_time['B']
                delivery_list[1] = earlist_delivery_date_B
            if cabinet == 'C' and date >= today and date <␣
↪earliest_order_date_C:
                earliest_order_date_C = date
                earlist_delivery_date_C = earliest_order_date_C +␣
↪delivery_time['C']
                delivery_list[2] = earlist_delivery_date_C
            if cabinet == 'D' and date >= today and date <␣
↪earliest_order_date_D:
                earliest_order_date_D = date
                earlist_delivery_date_D = earliest_order_date_D +␣
↪delivery_time['D']
                delivery_list[3] = earlist_delivery_date_D
    earliest_delivery_date = max(delivery_list)
    return earliest_delivery_date
```

[4]:
```
# My code #2

def earliest(instock, delivery_time, today):
    earliest_deliver_dates_list = []
    cabinet_list = list(instock.keys())

    for i in range(len(cabinet_list)):
        cabinet = cabinet_list[i]
        order_dates_list = instock[cabinet]
        for dates in order_dates_list:
            if dates >= today:
                earliest_deliver_dates_list.append(dates +␣
↪delivery_time[cabinet])
                break
    earliest_deliver_dates = max(earliest_deliver_dates_list)

    return earliest_deliver_dates
```

[8]:
```
# My code #3: Best.

def earliest(instock, delivery_time, today):
    cabinets = instock.keys()
    earliest_list = []
    for cabinet in cabinets:
        order_dates = instock[cabinet]
        delivery_dates = delivery_time[cabinet]
        for date in order_dates:
            if date >= today:
                earliest_list.append(date + delivery_dates)
                break
    return max(earliest_list)
```

[252]:
```
# Solution

def earliest(instock, delivery_time, today):
    earliest_deliver_dates = []
    for cabinet in instock.keys():
        orderdate = min([day for day in instock[cabinet] if day >= today])
        earliest_deliver_dates.append(orderdate + delivery_time[cabinet])
    return max(earliest_deliver_dates)
```

[385]:
```
# Sample runs:
instock={'A': [1,3,8,9,14,18,30,31,60,61,80,90,100],'B':␣
 ↪[5,6,7,20,21,40,70,71,100],'C': [1,2,9,10,25,50,90,91,100],'D':␣
 ↪[3,4,10,11,29,35,59,81,100]}
delivery_time={'A':3,'B':5,'C':7,'D':4}
earliest(instock,delivery_time, 21)
```

[385]:
```
33
```

[387]:
```
print(earliest(instock,delivery_time, 1))
```

```
10
```

[389]:
```
earliest(instock,delivery_time, 60)
```

[389]:
```
97
```

[379]:
```
## 2023 - Q2. Scheduling Contractors (8 Points): Don't try to be a clean code.
```

[240]:
```
```

```
available={'Cabinet Installers':[3,4,7,8,9,12,21,23],'Electrician':
↪[6,7,11,12,15,16,24,25],'Plumber':[1,4,7,8,13,17,19,21],'Painters':
↪[5,8,9,12,13,14,15,18,19,20,23]}
sequence=['Cabinet Installers','Plumber','Electrician','Painters']
days_needed=[3,2,1,3]

days_list = []

for i in range(len(sequence)):
    count = 0
    contractor = sequence[i]
    available_days = available[contractor]
    for days in available_days:
        if i == 0:
            days_list.append(days)
            count += 1
            if count == days_needed[i]:
                break
        else:
            if days > days_list[-1]:
                days_list.append(days)
                count += 1
                if count == days_needed[i]:
                    break
    print(contractor)
    print(days_list)
```

```
Cabinet Installers
[3, 4, 7]
Plumber
[3, 4, 7, 8, 13]
Electrician
[3, 4, 7, 8, 13, 15]
Painters
[3, 4, 7, 8, 13, 15, 18, 19, 20]
```

[260]:
```
# My code

def completion_date(available, sequence, days_needed):
    days_list = []

    for i in range(len(sequence)):
        count = 0
        contractor = sequence[i]
        available_days = available[contractor]
        for days in available_days:
            if i == 0:
                days_list.append(days)
                count += 1
                if count == days_needed[i]:
                    break
            else:
                if days > days_list[-1]:
                    days_list.append(days)
                    count += 1
                    if count == days_needed[i]:
                        break
    return days_list[-1]
```

[161]:
```
# Sample runs:
available={'Cabinet Installers':[3,4,7,8,9,12,21,23],'Electrician':
 ↪[6,7,11,12,15,16,24,25],'Plumber':[1,4,7,8,13,17,19,21],'Painters':
 ↪[5,8,9,12,13,14,15,18,19,20,23]}
sequence=['Cabinet Installers','Plumber','Electrician','Painters']
days_needed=[3,2,1,3]
completion_date(available,sequence,days_needed)
```

[161]:
```
20
```

[165]:
```
completion_date(available,['Plumber','Painters'],[4,4])
```

[165]:
```
14
```

[169]:
```
completion_date(available,['Cabinet Installers','Painters'],[4,5])
```

[169]:
```
15
```

[242]:
```
completion_date(available,['Plumber','Cabinet Installers','Plumber'],[2,3,1])
```

[242]:
```
13
```

[377]:
```
## 2023 - Q3. Simulating Contractor Availability (9 Points): Hardest
```

[14]:
```
from numpy.random import default_rng
rng = default_rng()

T = 10

schedule = []

for day in range(1, T+1):
    print(f'day: {day}')
    number_of_jobs = rng.poisson(0.4)
    print(f'number_of_jobs: {number_of_jobs}')
    for jobs in range(number_of_jobs):
        length_of_days = rng.choice([1,2,3,5],p=[0.3,0.4,0.2,0.1])
        schedule += [0] * length_of_days
        print(f'jobs: {jobs}, length: {length_of_days}')
    if len(schedule) < day:
        schedule.append(1)
    print(schedule)
print([day+1 for day in range(T) if schedule[day] == 1])
```

```
day: 1
number_of_jobs: 0
[1]
day: 2
number_of_jobs: 0
[1, 1]
day: 3
number_of_jobs: 1
jobs: 0, length: 1
[1, 1, 0]
day: 4
number_of_jobs: 0
[1, 1, 0, 1]
day: 5
number_of_jobs: 0
[1, 1, 0, 1, 1]
day: 6
number_of_jobs: 0
[1, 1, 0, 1, 1, 1]
day: 7
number_of_jobs: 1
jobs: 0, length: 1
[1, 1, 0, 1, 1, 1, 0]
day: 8
number_of_jobs: 2
jobs: 0, length: 5
jobs: 1, length: 3
[1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
day: 9
number_of_jobs: 1
jobs: 0, length: 2
[1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
day: 10
number_of_jobs: 0
[1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1, 2, 4, 5, 6]
```

[20]:
```
# Alternative Solution: Best
```

```python
def availability(T):
    from numpy.random import default_rng
    rng = default_rng()

    free_days = [1] * T
    for i in range(T):
        num_jobs = rng.poisson(0.4)
        for j in range(num_jobs):
            days_needed = rng.choice([1, 2, 3, 5], p=[.3, .4, .2, .1])
            for k in range(i, T):
                if free_days[k] == 1:
                    free_days[k] = 0
                    days_needed -= 1
                if days_needed == 0:
                    break

    schedule = []
    for i in range(T):
        day = i + 1
        if free_days[i] == 1:
            schedule.append(day)
    return schedule
```

[22]:
```python
availability(10)
```

[22]:
```
[1, 2, 8, 9, 10]
```

[24]:
```python
availability(30)
```

[24]:
```
[1, 2, 3]
```