CS5132 PA1 Report

Prannaya Gupta (M22504)

August 23, 2022

1 Explaining the Algorithm

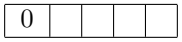
The solution is to use a **deque**. We illustrate it with an altered version of the provided prompt (in order and taken on 18th July 2022, with $k = 3$):

1. Melissa (Expiry date: 16-Apr-2022), vacation trip booked on 17-Sep-2022
2. Shamir (Expiry date: 10-Oct-2022), vacation trip booked on 30-Jul-2022
3. Clarice (Expiry date: 20-Nov-2022), business trip booked on 31-Aug-2022
4. Jacky (Expiry date: 19-Nov-2023), schooling trip booked on 17-Sep-2022
5. Priya (Expiry date: 11-Sep-2022), no vacation booked

First, since we know $N = 5$, thus $n = N - k + 1 = 5 - 3 + 1 = 3$.

We start with the following deque (of size $k = 3$):

Following this, we start by adding the first element, who is Melissa, to the tail, thus we have the following deque now (Note: we use the indices to label):

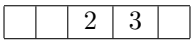


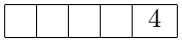
Now, we move to the next element, and we remove all the elements from the tail that are of lower priority than the next element, then we insert the new element.

Melissa and Shamir are both either expired or 180 days to being expired, are both going for vacations as well, hence the trip dates are crucial. Since Shamir's trip date is before Melissa's trip date, Melissa (0) is removed and then Shamir (1) is added.

From here, we move to the next element until we get to the end of the window (which is the 3rd element). First, we remove Shamir (1) and add Clarice (2). At this stage, we move to removing this value and adding it into the final list, which is Clarice (2). We also check if the first value in the deque is a value ahead of a window behind, which would be 1. However, since Shamir isn't in the deque (thanks to the previous step), we ignore it. If he were, we would have removed him from the deque.

This same thing continues in the following sequence:





Thus, we end up with Clarice (2) in the list and both Jacky and Priya not in the list. This is because both are of lower priority that Clarice.

Essentially we have the following pseudocode:

Algorithm 1 The Sliding Window Maximum

```
1:  procedure SlidingMax(a, k)                    ▷ Find maximums as an array
2:      N = len(a)
3:      n = N − k + 1
4:      d = deque()
5:      c = []
6:      for i = 1 to N-1 do
7:          while len(d) ≠ 0 and a[i] > a[d.peekLast] do
8:              d.removeLast()
9:          end while
10:         deq.addLast(i)
11:         if i ≥ n − 1 then
12:             if i not in c then
13:                 c.add(a[d.first])
14:             end if
15:             if d.first = i − n + 1 then
16:                 d.removeFirst()
17:             end if
18:         end if
19:     end for
20:     return c
21: end procedure
```

The time complexity of this algorithm is **O(N)** (since the while loop is effectively a removal of almost all N elements in the list, so $T(N) = 2N$. The space complexity is **O(N)** due to the inclusion of an auxillary array.

2 Testing the Algorithm

To test the algorithm, the following test cases are proposed:

2.1 Priority Testing

Evaluating based on the deadline for this assignment, which is 14th August 2022, we take the following people in:

1. Melissa (Expiry date: 22-Nov-2022), business trip booked on 12-Dec-2022
2. Shamir (Expiry date: 02-Mar-2023), business trip booked on 12-Dec-2022
3. Clarice (Expiry date: 02-Mar-2023), schooling trip booked on 12-Dec-2022
4. Jacky (Expiry date: 02-Mar-2023), schooling trip booked on 11-Apr-2022

This case was determined by the following conditions:

- Expiry Date for Melissa is 100 Days from Today (*effectively* expired)
- Expiry Dates for Shamir, Clarice and Jacky are 200 Days from Today (not *effectively* expired)
- Trip Dates for Melissa, Shamir and Clarice are 120 Days from Today
- Trip Date for Jacky are 240 Days from Today (*much* later)

We spin this same configuration around and then spin k between 1 and 4. This way, we can test whether the priorities themselves work, and whether the deque logic actually works.

2.1.1 Sample Tests

Test 1

```
4,14-Aug-2022
0,T001324,Melissa,22-Nov-2022,12-Dec-2022,B
1,E011964,Shamir,02-Mar-2023,12-Dec-2022,B
2,E013285,Clarice,02-Mar-2023,12-Dec-2022,S
3,T793412,Jacky,02-Mar-2023,11-Apr-2023,S
```

```
k = 2: [0.  T001324 (Melissa), 2.  E013285 (Clarice)]
k = 3: [0.  T001324 (Melissa), 2.  E013285 (Clarice)]
k = 4:  [0.  T001324 (Melissa), 1.  E011964 (Shamir), 2.
E013285 (Clarice), 3.  T793412 (Jacky)]
```

Test 2

```
4,14-Aug-2022
0,T793412,Jacky,02-Mar-2023,11-Apr-2023,S
1,E013285,Clarice,02-Mar-2023,12-Dec-2022,S
2,E011964,Shamir,02-Mar-2023,12-Dec-2022,B
3,T001324,Melissa,22-Nov-2022,12-Dec-2022,B
```

```
k = 2: [1.  E013285 (Clarice), 3.  T001324 (Melissa)]
k = 3: [1.  E013285 (Clarice), 3.  T001324 (Melissa)]
k = 4:  [0.  T793412 (Jacky), 1.  E013285 (Clarice), 2.
E011964 (Shamir), 3.  T001324 (Melissa)]
```

2.2 Corner Case Testing

To isolate corner cases, we uh wrack our brain multiple times. *No biggie.*

2.2.1 Case 1: All are the same

Ideally if all the values are the same, it only picks the first few samples due to the fact that we should prioritize the earlier applicants.

```
10,14-Aug-2022
0,T001324,Melissa,22-Nov-2022,12-Dec-2022,B
1,T001324,Shamir,22-Nov-2022,12-Dec-2022,B
2,T001324,Clarice,22-Nov-2022,12-Dec-2022,B
3,T001324,Jacky,22-Nov-2022,12-Dec-2022,B
4,T001324,Priya,22-Nov-2022,12-Dec-2022,B
5,T001324,Amy,22-Nov-2022,12-Dec-2022,B
6,T001324,Jake,22-Nov-2022,12-Dec-2022,B
7,T001324,Leonard,22-Nov-2022,12-Dec-2022,B
8,T001324,Jeff,22-Nov-2022,12-Dec-2022,B
9,T001324,Jim,22-Nov-2022,12-Dec-2022,B
```

```
k = 2: [0.  T001324 (Melissa), 1.  T001324 (Shamir)]
k = 3:  [0.  T001324 (Melissa), 1.  T001324 (Shamir), 2.
T001324 (Clarice)]
... and so on and so forth.
```

2.2.2 Case 2: (I lose marks for not being able to think of a 5th Case)

2.3 Stress Testing

To test out the values, we randomly determine the expiry date between -180 and 360 days away from today, the trip date as between 0 and 360 days away from today, and randomly assign the trip priority between `"B"`, `"S"`, `"V"` and `"N"`. We randomly generate 10,000 such samples to test how much stress the algorithm can truly take. We test $k = 2$ (which coalesces to one value if it is between 1 and 9998, meant to test if it can take a large window) and $k = 9999$ (which would show if it can conduct pair-to-pair checks). It passes successfully, and due to the volume of test cases, I have instead placed it on the web.

The code (in Python since that's easier to code in) is as follows:

```
# Stress Testing

import datetime as dt
from random import randint, shuffle, choice
today = dt.datetime.now()

def randomSample():
    expiry = (today + dt.timedelta(randint(-180,360))).
        strftime("%d-%b-%Y")
    tripType = choice(["B", "S", "V", "N"])
    trip = (today + dt.timedelta(randint(0,360))).
        strftime("%d-%b-%Y") if tripType != "N" else "NIL"
    return (i, "", "", expiry, trip, tripType)

file = open("tests/applicants3.csv", "w+")
print("10000,14-Aug-2022", file=file)
for i in range(10000):
    print(*randomSample(), sep=",", file=file)

file.close()
```