<u>AUTOMATED MARKET OPENING SCHEDULER DURING COVID</u>

- ## Problem Statement:

  The problem given is defined by two characteristic that is:

  1. all types of shops opening in one time slot in the same market should sell related items (items generally bought together)
  2. all types of shops opening in parallel markets should be as far away as possible to avoid people's movement to all of the markets (selling items that are generally not bought together)

  The goodness of a schedule can be calculated using similarities of all pairs within a single time slot in the same market (s(t1, t2)) and distances of all pairs within a single time slot in the parallel market (d(t1,t2))

  Objective Function = s (t1, t2)+C*[d(t1, t2)]

  s (t1, t2) = 1 - d(t1, t2)

- ## Goal:

  Search algorithm to find the best possible schedule with maximum objective function value.

- ## Proposed Solution:

  - The solution will be optimal when we choose the algorithm that finds optimal value of goodness of schedule function that is our objective function. This can be achieved by using local search algorithms as local search algorithms are useful for solving optimization problems, in which the aim is to find the best state according to an objective OPTIMIZATION PROBLEM function.
  - We have used the ***Hill Climbing Algorithm*** to maximize the given function using the given values of input.
  - Maximize the goodness of schedule function with respect to time slots.

- **Description:**
    - **Functions:**
        - **objectiveCal**: This function computes the goodness of schedule function value of each shop from the given shops and then returns the maximum value with its position in one time slot.
        - **optimalSlot**: Function returns the optimal solution for different time slots.
    - **Attributes:**
        - **k:** total types of shops opening in one time slot in one market
        - **m:** number of parallel markets
        - **T:** number of time slots
        - **C:** trade-off constant

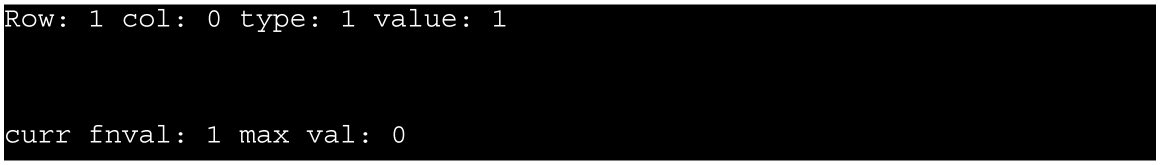
- **Example:**

Given: k=2, T=1, m=2, C=1.

Shops= {1,2,3,4}

Steps:

- arr[0][0]=selects shop 4(randomly).
- From remaining shops it will select one shop and calculate the value of objective function at each position.
- Returns maximum value of objective function along with position for the selected shop.
- From the given example we select shop 1, 2 and 3 and return its position where it maximizes objective function.



```
Row: 1 col: 0 type: 1 value: 1


curr fnval: 1 max val: 0
```

```
row: 1 col: 0 type: 2 value: 0.7

curr fnval: 0.7 max val: 1
```

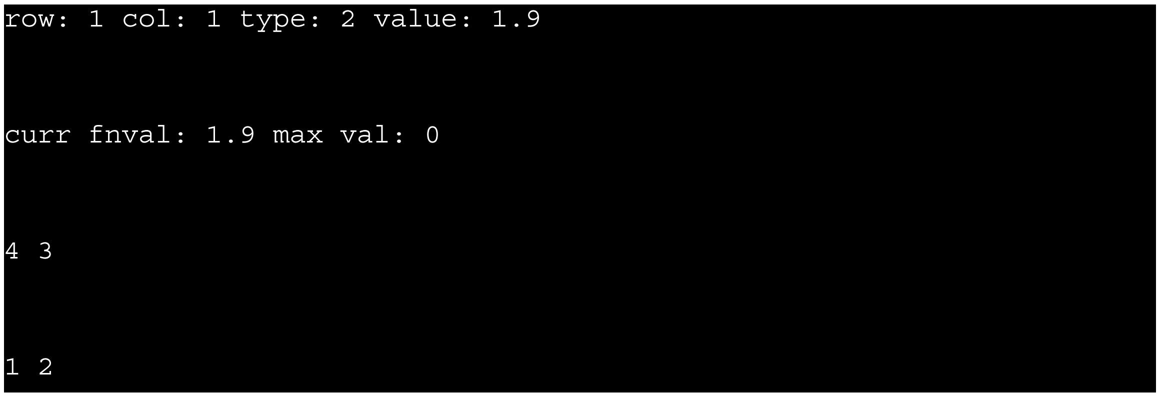
```
row: 0 col: 1 type: 3 value: 0.7

curr fnval: 0.7 max val: 1
```

- We see that row 1 and column 0 have a maximum value of objective function that is equal to 1, it selects it.

```
4 0

1 0
```

- Remaining shops ={ 2,3}
- Using the same concept we iterate for other remaining shops for one time slot.

```
row: 1 col: 1 type: 2 value: 1.3

curr fnval: 1.3 max val: 0

row: 0 col: 1 type: 3 value: 1.5

curr fnval: 1.5 max val: 1.3

4 3

1 0
```



```
row: 1 col: 1 type: 2 value: 1.9

curr fnval: 1.9 max val: 0

4 3

1 2
```

- **Conclusion:** We get the desired schedule of shops in each time slot. In our algorithm there is exponential decrease in time because we remove the shops that have been used and positioned correctly.