Problem 2

Parallel processing

In this problem we will simulate a program that processes a list of jobs in parallel.

Problem description

Task.

You have a program which is parallelized and uses n independent threads to process the given list of m jobs. Threads take jobs in the order they are given in the input. If there is a free thread, it immediately takes the next job from the list. If a thread has started processing a job, it doesn't interrupt or stop until it finishes processing the job. If several threads try to take jobs from the list simultaneously, the thread with smaller index takes the job. For each job you know exactly how long will it take any thread to process this job, and this time is the same for all the threads. You need to determine for each job which thread will process it and when will it start processing.

Input format.

The first line of the input contains integers n and m.

The second line contains m integers t_i — the times in seconds it takes any thread to process $i-th$ job.

The times are given in the same order as they are in the list from which threads take jobs.

Threads are indexed starting from 0

Constrains.

$1 \leq n \leq 10^5 ; 1 \leq m \leq 10^5 ; 0 \leq t_i \leq 10^9.$

Output format.

Output exactly m lines. $i-th$ line (0-based index is used) should contain two space-separated integers - the 0-based index of the thread which will process the i-th job and the time in seconds when it will start processing that job.

Sample output:

```
    input:
      2 5
      1 2 3 4 5
  output
    0 0
    1 0
    0 1
    1 2
    0 4
```

My approach

In this problem, I see that we can utilize the build_heap function which is implemented in **Problem 1**, so I developed into a Priority Queue using the exact same mechanism.

After we have the priority class, we also need to create a Worker class which represents the Thread, this class have 2 attributes: id and nextfreetime. ID is the the id of the worker, nextfreetime is the time that this worker will be free.

What's special about this class is that I have built build-in functions to compare different objects in the same class. Since we want to put these objects into the Priority Queue, we need to compare it efficiently. The **lt()** and **gt()** are the magic functions in python which let us to do so. All we need to do is modified these functions.

Finally we need a function to process the parallel works. This function will iterate the jobs list, for each job we pop out the worker with the lowest priority (min heap is used in this case) which indicates that it free enough to take the next job. All the sortings will be done by Priority Queue. The complexity of this function is O(n) where n is the jobs

In [316]:
```
class PriorityQ(object):
    """ this class is just a part of Priority queue. There are some of the functions will not be optim
ized. It lack of Change priority function and sometimes the sift_up won't work as expect
            There are two mode, max heap and min heap in this class
    """
    def __init__(self,mode='min'):
        self.lst = lst=[]
        self.heap = self.build_heap()
        self.mode = mode
    # find the indexes of parent, left child and right child
    def parent(self,i):
        return (i-1)//2
    def left_child(self,i):
        return 2*i+1
    def right_child(self,i):
        return 2*i+2

    def insert(self,val):
        self.heap.append(val)
        self.sift_up(len(self.heap)-1,self.heap)
    def extract(self):
        self.heap[0],self.heap[-1] = self.heap[-1],self.heap[0] # swap the root with the lowest childr
en
        result = self.heap.pop() # pop out the value with highest priority
        self.sift_down(0,self.heap) # sift down just in case the root is not correct
        return result
    def sift_up(self,i,H):
        if self.mode == 'min':
            while i>=1 and H[self.parent(i)] > H[i]:
                H[self.parent(i)],H[i] = H[i],H[self.parent(i)]
                i = self.parent(i)
        elif self.mode =='max':
            while i>=1 and H[self.parent(i)] < H[i]:
                H[self.parent(i)],H[i] = H[i],H[self.parent(i)]
                i = self.parent(i)
    def sift_down(self,i,H):
        max_idx = i
        try:
            l = self.left_child(i)
            if self.mode == 'min':
                if l <= len(H) and H[l] < H[max_idx]:
                    max_idx = l
            elif self.mode == 'max':
                if l <= len(H) and H[l] > H[max_idx]:
                    max_idx = l
        except:
            pass
        try:
            r = self.right_child(i)
            if self.mode == 'min':
                if r <= len(H) and H[r] < H[max_idx]:
                    max_idx = r
            elif self.mode == 'max':
                if r <= len(H) and H[r] > H[max_idx]:
                    max_idx = r
        except:
            pass
        if i != max_idx:
            H[i],H[max_idx] = H[max_idx],H[i]
            self.sift_down(max_idx,H)

    def build_heap(self):
        n = len(self.lst)
        H = self.lst
        for i in reversed(range(n//2)):
            self.sift_down(i,H)
        return H
    def print_heap(self):
        print(self.heap)
```

In [ ]:
```
class Worker(object):
    def __init__(self,id):
        self.id = id
        self.nextfreetime = 0
    def __lt__(self,other): # build < operation
        if self.nextfreetime == other.nextfreetime:
            return self.id < other.id
        else:
            return (self.nextfreetime < other.nextfreetime)
    def __gt__(self, other): # build > operation
        if self.nextfreetime == other.nextfreetime:
            return (self.id > other.id)
        else:
            return (self.nextfreetime > other.nextfreetime)
    def __eq__(self, other): # build = operation
        return self.nextfreetime == other.nextfreetime
```

In [306]:
```
n,t = open("C:/Users/default.DESKTOP-IV77C8K/Desktop/github/Data-structrure/week2_priority_queues_and_
disjoint_sets/2_job_queue/tests/02", 'r')
workers=int(n.split()[0])
jobs=list(map(int,t.split()))
```

In [309]:
```
len(jobs)
```

Out[309]: 100

In [317]:
```
def parallel_processing(workers,jobs):
    worker_list = [None]*len(jobs)
    start_time_list = [None]*len(jobs)
    PQ = PriorityQ()
    for i in range(workers):
        w = Worker(i)
        PQ.insert(w)
    for j in range(len(jobs)):
        worker=PQ.extract()
        worker_list[j]=worker.id
        start_time_list[j]=worker.nextfreetime
        worker.nextfreetime+=jobs[j]
        PQ.insert(worker)
    for k in range(len(jobs)):
        print("worker, nextimejob:  ",worker_list[k],start_time_list[k] )
```

In [318]:
```
w = 4
j = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 1]
```

In [320]:
```
parallel_processing(w,j)
```

```
worker, nextimejob:  0 0
worker, nextimejob:  1 0
worker, nextimejob:  2 0
worker, nextimejob:  3 0
worker, nextimejob:  0 1
worker, nextimejob:  1 1
worker, nextimejob:  2 1
worker, nextimejob:  3 1
worker, nextimejob:  0 2
worker, nextimejob:  1 2
worker, nextimejob:  2 2
worker, nextimejob:  3 2
worker, nextimejob:  0 3
worker, nextimejob:  1 3
worker, nextimejob:  2 3
worker, nextimejob:  3 3
worker, nextimejob:  0 4
worker, nextimejob:  1 4
worker, nextimejob:  2 4
worker, nextimejob:  3 4
```

In [321]:
```
w = 2
j = [1,2,3,4,5]
```

In [322]:
```
parallel_processing(w,j)
```

```
worker, nextimejob:  0 0
worker, nextimejob:  1 0
worker, nextimejob:  0 1
worker, nextimejob:  1 2
worker, nextimejob:  0 4
```

In [315]:
```
parallel_processing(workers,jobs)
```

```
worker, nextimejob:  0 0
worker, nextimejob:  1 0
worker, nextimejob:  2 0
worker, nextimejob:  3 0
worker, nextimejob:  4 0
worker, nextimejob:  5 0
worker, nextimejob:  6 0
worker, nextimejob:  7 0
worker, nextimejob:  8 0
worker, nextimejob:  9 0
worker, nextimejob:  7 28787989
worker, nextimejob:  0 124860658
worker, nextimejob:  5 235543106
worker, nextimejob:  7 246504708
worker, nextimejob:  4 311346104
worker, nextimejob:  3 349021732
worker, nextimejob:  1 388437511
worker, nextimejob:  9 409836312
worker, nextimejob:  3 595181715
worker, nextimejob:  9 619331540
worker, nextimejob:  6 665655446
worker, nextimejob:  8 706718118
worker, nextimejob:  1 707425685
worker, nextimejob:  2 753494620
worker, nextimejob:  5 845266823
worker, nextimejob:  3 882135358
worker, nextimejob:  0 1030418770
worker, nextimejob:  7 1127475443
worker, nextimejob:  0 1226053489
worker, nextimejob:  1 1239139577
worker, nextimejob:  4 1283739291
worker, nextimejob:  8 1294605118
worker, nextimejob:  6 1439488110
worker, nextimejob:  5 1448354598
worker, nextimejob:  3 1449889884
worker, nextimejob:  2 1488265968
worker, nextimejob:  8 1492349613
worker, nextimejob:  1 1543303433
worker, nextimejob:  8 1552026381
worker, nextimejob:  1 1559394794
worker, nextimejob:  9 1564932477
worker, nextimejob:  6 1700692640
worker, nextimejob:  8 1762133312
worker, nextimejob:  9 1785403332
worker, nextimejob:  9 1812712043
worker, nextimejob:  4 1837312198
worker, nextimejob:  0 1923308283
worker, nextimejob:  6 1925648764
worker, nextimejob:  2 1951684676
worker, nextimejob:  8 2051689774
worker, nextimejob:  5 2089663653
worker, nextimejob:  7 2096109190
worker, nextimejob:  8 2100522306
worker, nextimejob:  3 2222963076
worker, nextimejob:  1 2322394874
worker, nextimejob:  4 2343014119
worker, nextimejob:  6 2347796988
worker, nextimejob:  4 2405352997
worker, nextimejob:  1 2461363674
worker, nextimejob:  8 2507700390
worker, nextimejob:  0 2563144828
worker, nextimejob:  9 2649950912
worker, nextimejob:  2 2656370100
worker, nextimejob:  6 2792823301
worker, nextimejob:  1 2831769133
worker, nextimejob:  9 2930053292
worker, nextimejob:  7 2936370273
worker, nextimejob:  3 2946336306
worker, nextimejob:  5 3033959396
worker, nextimejob:  0 3106934106
worker, nextimejob:  8 3114837717
worker, nextimejob:  4 3140480513
worker, nextimejob:  2 3205065638
worker, nextimejob:  2 3236887517
worker, nextimejob:  2 3282959311
worker, nextimejob:  0 3306478249
worker, nextimejob:  3 3363648528
worker, nextimejob:  8 3408035489
worker, nextimejob:  4 3439390637
worker, nextimejob:  6 3444003346
worker, nextimejob:  1 3470199591
worker, nextimejob:  5 3480453036
worker, nextimejob:  0 3490305631
worker, nextimejob:  1 3506560968
worker, nextimejob:  7 3585123350
worker, nextimejob:  0 3618937640
worker, nextimejob:  0 3638771343
worker, nextimejob:  2 3792650094
worker, nextimejob:  4 3815595208
worker, nextimejob:  5 3884584250
worker, nextimejob:  9 3911731663
worker, nextimejob:  8 3932551852
worker, nextimejob:  5 3943003928
worker, nextimejob:  5 3992901123
worker, nextimejob:  1 4042282164
worker, nextimejob:  7 4055834901
worker, nextimejob:  9 4086865161
worker, nextimejob:  0 4115619221
worker, nextimejob:  5 4181243134
worker, nextimejob:  6 4192821467
```