[Task](https://leetcode.com/problems/task-scheduler/) Scheduler

**package** unsolvedpackage;

**import** java.util.\*;

**public** **class** TaskScheduler {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

**char**[] tasks = {'A','A','A','B','B','B'};

System.***out***.println(*leastInterval*(tasks, 2));

}

**public** **static** **int** leastInterval(**char**[] tasks, **int** n) {

//edge case

**if**(tasks == **null** || tasks.length == 0)

**return** 0;

**int**[] count = **new** **int**[26];

//keep count of each task

**for**(**char** c : tasks) {

count[c - 'A']++;

}

//max heap for the instances of task

Queue<Integer> queue = **new** PriorityQueue<>(26 , Collections.*reverseOrder*());

**for**(**int** i : count) {

**if**(i > 0) {

queue.add(i);

}

}

**int** time = 0;

**while**(!queue.isEmpty()) {

**int** i = 0 ;

//temporary list to add remaining instances of task after scheduling

List<Integer> temp = **new** ArrayList<>();

//iterate till cooling time is not passed

**while**(i <= n) {

**if**(!queue.isEmpty()) {

//check if any other instance of task is remaining add to list

**if**(queue.peek() > 1) {

temp.add(queue.poll() - 1);

}

**else**

queue.poll();

}

//during each iteration increase time

time++;

//if queue is empty and no task is in list break because no more instance of task is remaining to schedule

**if**(queue.isEmpty() && temp.size() == 0)

**break**;

i++;

}

//after cooling period passed add all remaining instances to queue.

**for**(**int** j : temp)

queue.add(j);

}

**return** time;

}

}

Time complexity : O(n). Number of iterations will be equal to resultant time.

Space Complexity : O(1), queue and temp size will not exceed O(26)