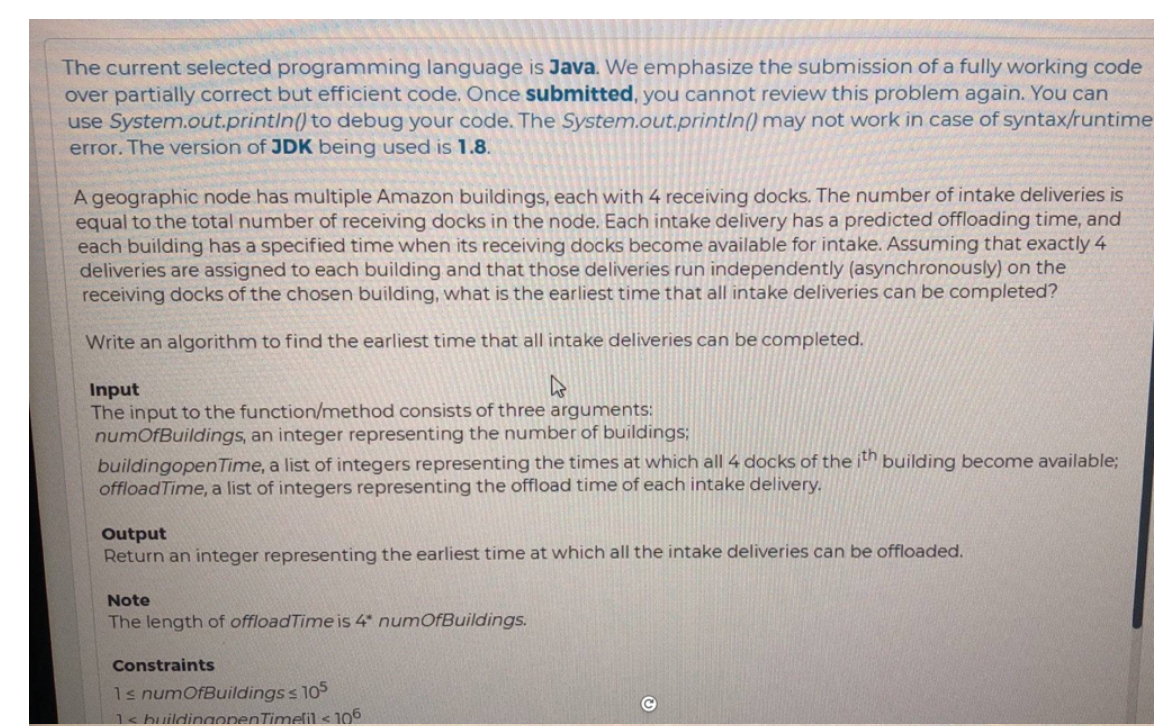
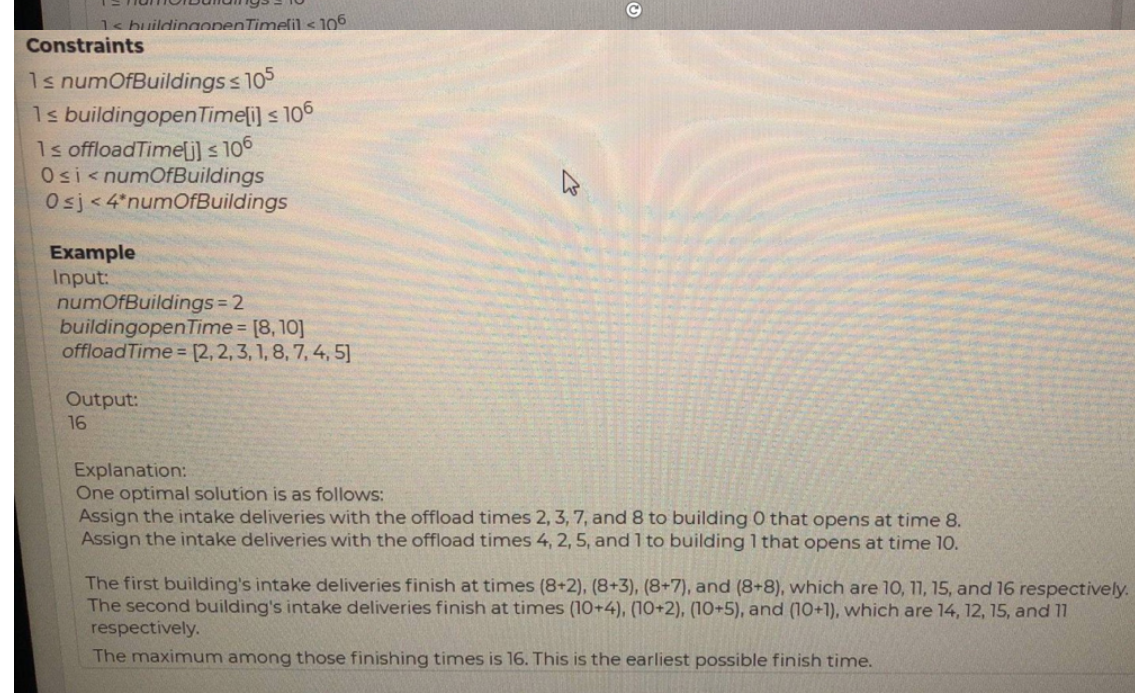
1. Amazon | OA 2020 | Earliest Time to Complete Deliveries

LEETCODE DISCUSS : https://leetcode.com/discuss/interview-question/892370/





Java Greedy

public static int earliestTime(int numOfBuildings, List<Integer> openTime, List<Integer> offloadTime) {

openTime.sort(null);

offloadTime.sort(Collections.reverseOrder());

int earliestTime = 0;

for (int i = 0; i < numOfBuildings; i++) {

earliestTime = Math.max(earliestTime, openTime.get(i) + offloadTime.get(i \* 4));

}

return earliestTime;

}

Greedy Algorithm:

* Sort the offload time (descending order) [8 7 5 4 3 2 2 1]
* Sort the building open time (ascending order) [8 10]
* Start assigning four from offload time to each building, in the given example
  + Building 0 with time 8 gets - 8,7,5,4 (max\_time = 8+8=16)
  + Building 1 with time 10 gets - 3,2,2,1 (max\_time = 10+3=13)
* For further simplicity it can be max (building\_open\_time[i] + offload\_time[4\*i-1]) where i = 3,7....4n-1, i=0 case should also be taken separately
  + For instance in this example : max(8+8 , 3+10)

Optional :

import java.util.\*;

class CompleteDeliveries{

public static void main(String[] args){

int numBuilding = 2;

int[] openTime = {8,10};

int[] offloadTime = {2,2,3,1,8,7,4,5};

//int[] offloadTime = {2,2,3,1,8,7,4,5,13,5,4,2};

System.out.println(findEarliestTime(numBuilding,openTime,offloadTime));

}

public static int findEarliestTime(int numBuilding,int[] openTime,int[] offloadTime){

Arrays.sort(openTime);

// To sort in descending order, first sort in ascending the swap elements

Arrays.sort(offloadTime);

// Swap first and last elements and proceed

for (int i = 0, j = offloadTime.length - 1, tmp; i < j; i++, j--) {

tmp = offloadTime[i];

offloadTime[i] = offloadTime[j];

offloadTime[j] = tmp;

}

int max = Integer.MIN\_VALUE;

for(int i=0;i<numBuilding;i++)

max = Math.max(openTime[i]+offloadTime[i\*4],max);

return max;

}

}