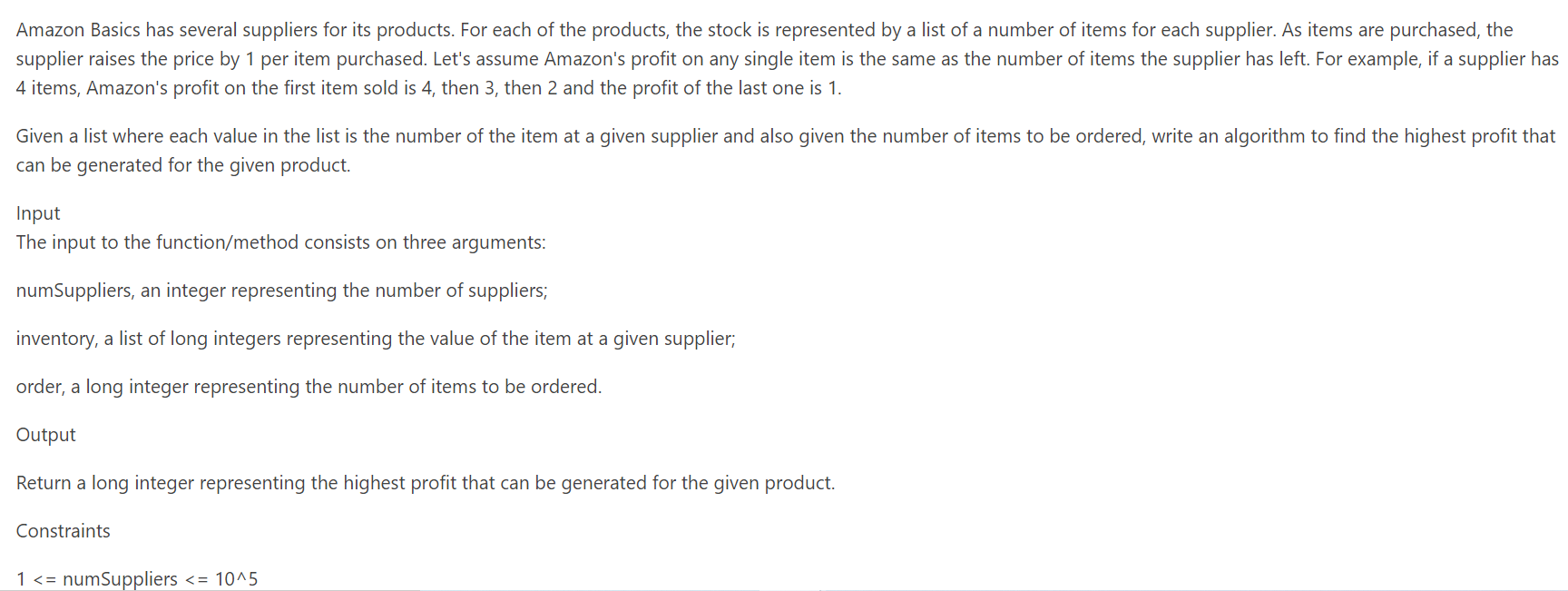
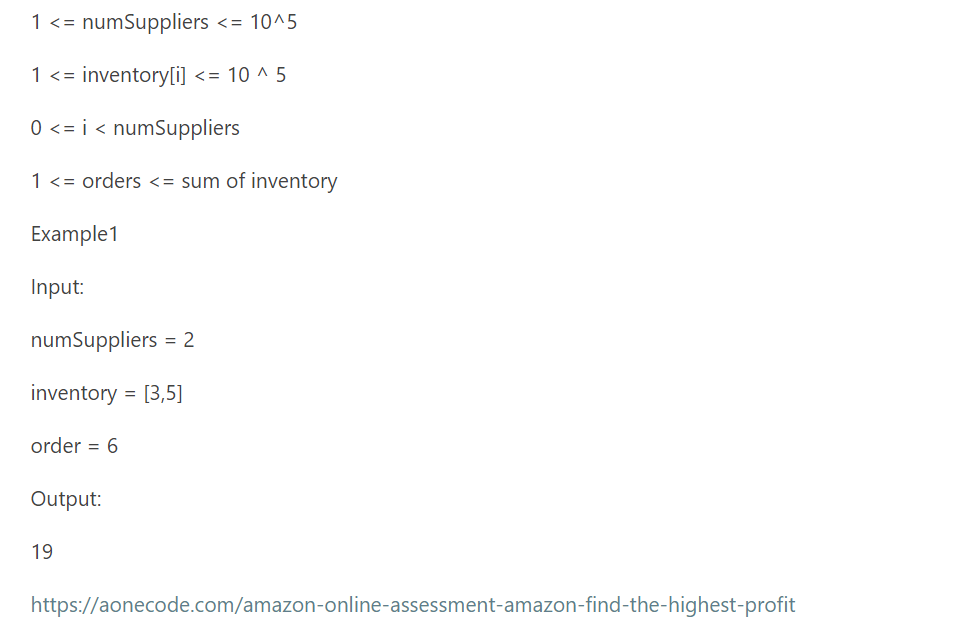
Amazon | OA 2020 | Find The Highest Profit

LEETCODE : <https://leetcode.com/discuss/interview-question/823177/>





Java version using HashMap from @kamui\_amaterasu33 with comments

import java.util.\*;

class max\_profit{

public static int maximum\_profit(int[] inventory, int order){

// create map with inventory value and no of occurences in map i.e if there are 2 items of stock 10,10 map will be 10,2

HashMap<Integer, Integer> hm=new HashMap();

for(int i:inventory){

hm.put(i, hm.getOrDefault(i,0)+1);

}

// find the maximum value in map. avoid sort so that we can achieve solution in O(n)

// we know that after each item is purchased profit is going to reduce by cut\_item-1

int cur\_max = Integer.MIN\_VALUE;

for(int i:hm.keySet()){

cur\_max=Math.max(cur\_max,i);

}

System.out.println(cur\_max);

int answer=0;

while(order>0){

// get the no of items in stock for cur\_max inventory

int no\_items = Math.min(order, hm.get(cur\_max));

answer += no\_items\*cur\_max; // calculate current profit

order = order-no\_items; // reduce no of items to order still

int left\_items = hm.get(cur\_max) - no\_items; // get the no of items in cur\_max key and deduce the no of items ordered

hm.put(cur\_max-1, hm.getOrDefault(cur\_max-1, 0)+no\_items); // we know for when we order next item it's price is going to drop by 1

// if there is entry in map then update the map value with no of items. if not create new entry

// if no items are left in stock for max\_profit inventory item. remove the entry from map and reduce the profit by 1

if(left\_items == 0){

hm.remove(cur\_max);

cur\_max--;

}

}

return answer;

}

public static void main(String[] args) {

int[] inventory = {10,2,8,4,6};

int order = 20;

int profit = maximum\_profit(inventory, order);

System.out.println("Total Profit is: "+ profit);

}

}

ALTERNATE : Java Solution using Priority Queue:

public class FindTheHighestProfit {

public int max\_profit(int[] inventory, int orders) {

int profit = 0;

PriorityQueue<Integer> pq = new PriorityQueue<>(inventory.length, Collections.reverseOrder());

for (int i : inventory) {

pq.add(i);

}

while (orders != 0 && !pq.isEmpty())

{

int top = pq.poll();

profit += top;

orders --;

pq.add(top-1);

}

return profit;

}

public static void main(String[] args) {

int[] inventory = {2,5}; //{10,2,8,4,6};

int order = 4; //20;

FindTheHighestProfit f = new FindTheHighestProfit();

int profit = f.max\_profit(inventory, order);

System.out.println("Total Profit is: "+ profit);

}

}