Algorithms Laboratory (CS29203)
Assignment 6: Heap & priority queue
Department of CSE, IIT Kharagpur

20th October 2022

Question-1 (50 points)

You are going to the popular BBQ Nation buffet restaurant in Kolkata where there are multiple options available that can be eaten without any restriction. However recently they have realized that keeping the dessert/sweet items in the buffet is getting really costly, and they have to impose some restrictions on it. So you can't eat unlimited sweet items any more, but you still want to maximize the number of sweets that you are allowed to eat. In the restaurant, different types of sweets are placed in different containers, and you can pick different sweet items from different containers with some restrictions. Each container has some specific number of sweets in it. Let us say that an array container[i] stores the number of sweets in the i-th container. While entering the restaurant, you are also given an integer number k. The rule is that, you can choose any container[i] and take $\lfloor container[i]/2 \rfloor$ sweets from it, and repeat this process exactly k times. You are allowed to take sweets from the same container more than once while following the above rule. Since your goal is to eat as much sweet items as possible, you would like to leave the minimum possible total number of sweets remaining in the containers after applying the k operations.

For example, let's say the container array is $container = [5, 4, 9]$, and $k = 2$. In this case, the minimum remaining sweets can be 12. First, you can take sweets from the last container. That means you will take $\lfloor 9/2 \rfloor = 4$ sweets from it. So the resulting container array will be $[5, 4, 5]$. Next you take from the first container, so you will take $\lfloor 5/2 \rfloor = 2$ sweets from it. So the resulting container array will be $[3, 4, 5]$. Hence the total number of remaining sweets is $3 + 4 + 5 = 12$. Note that you can't take sweets more than twice in this case since $k = 2$.

Your task is to solve this problem using heap data structure. *Hint: Think of a greedy approach by each time selecting a container with the naximum number of sweets. Use a priority queue (or max heap) to maintain the sweet counts in the containers.*

Example:

```
Input: container = 57, 45, 62, 76
       k = 3
Reamining total sweets: 143
```

Question-2 (50 points)

Consider two arrays arr1 and arr2 containing m and n integer elements (both unsorted). We define "*IndexSum*" to be a sum of the form arr1[i] + arr2[j] for some (valid) indices i and j. You are also given a positive integer k, and you have to find the k smallet *IndexSums* of the two arrays. Note that it is not important which i and j correspond to an *IndexSum*. Basically if you permute arr1 and arr2, these indices change, but the k smallest *IndexSums* remain the same. However, if the same *IndexSum* is realized by two or more different index pairs (for the same permutations of arr1 and arr2), these sums are to be reported separately.

A naive idea is to store the $m * n$ number of *IndexSums* in an array 'A', sort 'A', and report the k smallest elements in the sorted 'A'. This requires $O(mn \log(mn))$ time. Using min-heaps, you can do much better than this.

Convert 'A' to a min-heap. For k times, print the minimum followed by deleting the minimum. This improves the running time to $O(mn + k \log(mn))$, which is $O(mn)$ (we choose $k \leq m/\log m$ and $k \leq n/\log n$). Your task is to write a code to print the k smallest elements using the idea of min-heap as just stated. You may either hard code the input arrays, or read these from file (no need to take as user input).

Example:

```
Input:

m = 64
Array arr1:
647 225 200 820 789 338  72 274 407 577 306 167 928  40 417  86 751 384 697 144
137 823 241 986 665 468 225 121 372 143  86 737  86   4 557 874 341 628 148 748
923 173 633 852 212  50 656 681 153 353 824   8 176 783 993 559 970 936 399  61
797 203 797 882

n = 60
Array arr2:
206 354 757 547 700 623  14 623 514 646 194 444 414 849 125 566 202 948 292  96
732 285 374 702 940 772 762 737 974 559 620 898 631  96 445 331 437 177 672 951
822 866 395 955 715 520 240 636 187 532 731 637 535 823 339 475 314 819 931   7

k = 10

Output:
11 15 18 22 47 54 57 64 68 75
```