### Maximum Diamonds

There are **N** bags with diamonds in them. The i'th of these bags contains **A[i]**diamonds. If you drop a bag with **A[i]** diamonds, it changes to **A[i]/2**diamonds and you gain **A[i]**diamonds. Dropping a bag takes 1 minute. Find the **maximum number of diamonds** that you can take if you are given **K minutes.**

**Example 1:**

**Input:**

**N =** 5, **K =** 3

**A[] =** {2, 1, 7, 4, 2}

**Output:**

14

**Explanation:**

The state of bags is:

2 1 7 4 2

You take all diamonds from Third bag (7).

The state of bags becomes: 2 1 3 4 2

Take all diamonds from Fourth bag (4).

The state of bags becomes: 2 1 3 2 2

Take all diamonds from Third bag (3).

The state of bags becomes: 2 1 1 2 2

Hence, number of Diamonds = 7+4+3 = 14.

**Example 2:**

**Input:**

**N =** 3, **K =** 2

**A[] =** {7, 1, 2}

**Output:**

10

**Explanation:**You take all diamonds from First bag (7).  
The state of bags becomes: 3 1 2   
Take all diamonds from again First bag (3).  
The state of bags becomes: 1 1 2  
You can take a **maximum** of 10 diamonds.

**Expected Time Complexity:** O(NlogN)  
**Expected Auxiliary Space:** O(N)

**Constraints:**  
1 <= N <= 105  
0 <= K,A[i] <= 105

**Topic Tags**

[**Mathematical**](https://practice.geeksforgeeks.org/explore/?category%5b%5d=Mathematical) [**Queue**](https://practice.geeksforgeeks.org/explore/?category%5b%5d=Queue) [**STL**](https://practice.geeksforgeeks.org/explore/?category%5b%5d=STL) [**Data Structures**](https://practice.geeksforgeeks.org/explore/?category%5b%5d=Data%20Structures) [**Algorithms**](https://practice.geeksforgeeks.org/explore/?category%5b%5d=Algorithms) [**Heap**](https://practice.geeksforgeeks.org/explore/?category%5b%5d=Heap)

import java.io.\*;

import java.util.\*;

class GFG {

public static void main(String args[]) throws IOException {

BufferedReader read =

new BufferedReader(new InputStreamReader(System.in));

int t = Integer.parseInt(read.readLine());

while (t-- > 0) {

String S[] = read.readLine().split(" ");

int N = Integer.parseInt(S[0]);

int K = Integer.parseInt(S[1]);

int[] A = new int[N];

String S1[] = read.readLine().split(" ");

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

A[i] = Integer.parseInt(S1[i]);

Solution ob = new Solution();

System.out.println(ob.maxDiamonds(A,N,K));

}

}

}

// } Driver Code Ends

//User function Template for Java

class Solution {

static long maxDiamonds(int[] A, int N, int K) {

long sum=0;

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

for(int num : A)pq.add(num);

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

int temp=pq.poll();

sum+=temp;

pq.add(temp/2);

}

return sum;

}

};