Data structures and algo in java - Day 23

public class day23

{

public static void main(String[] args)

{

// int n = 28;

// findSquareRoot(n);

// int m = 125;

// int n = 3;

// int ans = findNthSquareRoot(m,n);

// System.out.println(ans);

// int m = 100;

// int n = 2 ;

// int ans = findCeilingNthRoot(m, n);

// System.out.println(ans);

int arr [] = {3,6,7,11};

int limit = 8;

kokoEatingBananas(arr, limit);

}

public static void findSquareRoot(int n)

{

int low = 1;

int high = n;

int ans = 1;

if(n==0 || n==1)

{

System.out.println(n);

}

while(low<=high)

{

int mid = (low+high)/2;

if(mid\*mid<=n) // or (mid<=n/mid)

{

ans = mid;

low = mid+1;

}

else

{

high = mid -1;

}

}

System.out.println(ans);

}

public static int findNthSquareRoot(int m,int n)

{

int low = 1;

int high = m ;

while(low<=high)

{

int mid = (low+high)/2;

if(midPowerN(mid,n)==m)

{

return mid;

}

else if(midPowerN(mid,n)<m)

{

low = mid + 1;

}

else

{

high = mid -1;

}

}

return -1;

}

public static int midPowerN(int mid , int n)

{

int ans = 1;

while(n>0)

{

if(n%2==1)

{

ans = ans\*mid;

n = n-1;

}

else

{

mid = mid\*mid;

n=n/2;

}

}

return ans;

}

public static int findCeilingNthRoot(int m, int n)

{

int low = 1;

int high = m ;

int ans = -1;

while(low<=high)

{

int mid = (low+high)/2;

int midpowN = findmidPowN(mid,n);

if(midpowN == m)

{

return mid;

}

else if (midpowN<m)

{

low = mid +1;

}

else {

ans = mid;

high = mid -1;

}

}

return ans;

}

public static int findmidPowN(int mid , int n)

{

int ans = 1;

while(n>0)

{

if(n%2==1)

{

ans = ans\*mid;

n=n-1;

}

else

{

mid = mid\*mid;

n=n/2;

}

}

return ans;

}

public static void kokoEatingBananas(int arr[], int limit)

{

int low = 1;

int high = getMax(arr);

int ans = 1;

while(low<=high)

{

int mid = (low+high)/2;

int bananaPerHour = findHours(arr,mid); //6

if(bananaPerHour<=limit)

{

ans = mid;

high = mid -1;

}

else

{

low = mid +1;

}

}

System.out.println(ans);

}

public static int findHours(int arr[],int mid)

{

int hours = 0;

for(int i=0;i<arr.length;i++)

{

hours += (int) Math.ceil((double) arr[i] / mid);

}

return hours;

}

public static int getMax(int arr[])

{

int max = arr[0];

for(int i=0;i<arr.length;i++)

{

if(arr[i]>max)

{

max = arr[i];

}

}

return max;

}

}