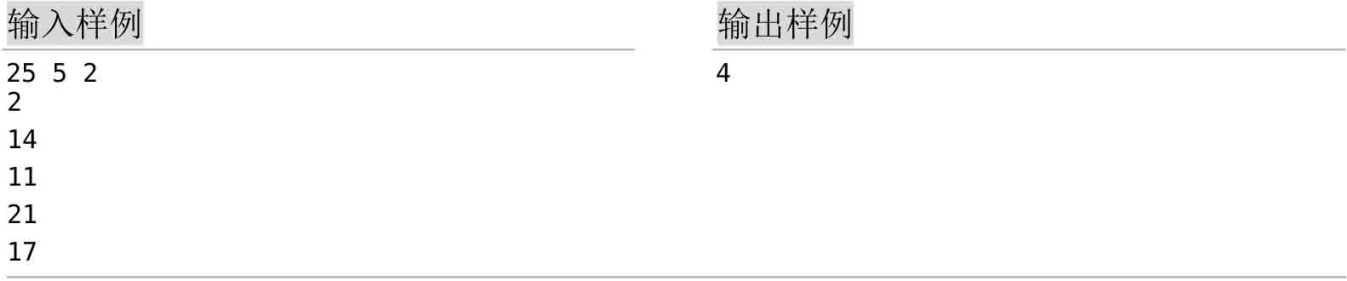
**POJ3258**

**题目描述（POJ3258）：**跳房子游戏指从河中的一块石头跳到另一块石头，这发生在一条又长又直的河流中，从一块石头开始，到另一块石头结束。长度为L（1≤L≤109），从开始到结束之间的石头数量为N（0≤N≤50 000），从每块石头到开始位置有一个整数距离di（0<di<L）。

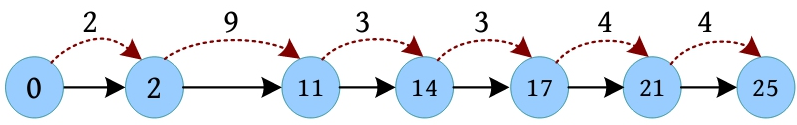
为了玩游戏，每头母牛都依次从起始石头开始，并尝试到达终点的石头，只能从石头跳到石头。当然，不那么灵活的母牛永远不会到达最后的石头，而是掉进河中。约翰计划移除几块石头，以增加母牛必须跳到最后的最短距离。不能删除起点和终点的石头，但约翰有足够的资源移除多达M块石头（0≤M≤N）。请确定在移除M块石头后，母牛必须跳跃的最短距离的最大值。

**输入：**第1行包含3个整数L、N和M。接下来的N行，每行都包含一个整数，表示从该石头到起始石头的距离。没有两块石头有相同的位置。

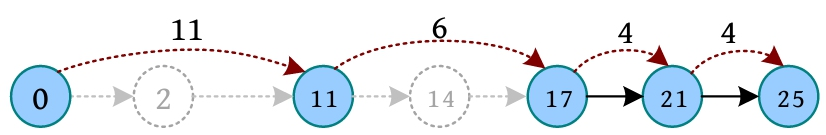
**输出：**单行输出移除M块石头后母牛必须跳跃的最短距离的最大值。



**题解：**根据输入样例，构建的图如下图所示。



在移除任何石头之前，跳跃的最短距离都是2（从0到2）。在移除2和14石头后，跳跃的最短距离是4（从17到21或从21到25）。



**1. 算法设计**

（1）如果移除的石头数等于总石头数（M=N），则直接输出L。

（2）增加开始（0）和结束（N+l）两块石头，到开始节点的距离分别为0和L。

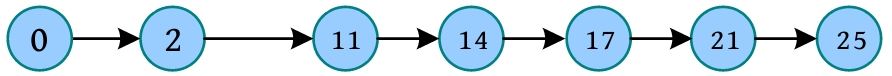
（3）对所有的石头都按照到开始节点的距离从小到大排序。

（4）令left=0，right=L，如果right-left>1，则mid=(right+left)/2，判断是否满足移除M块石头之后，任意间距都不小于mid。如果满足，则说明距离还可以更大，令left=mid；否则令right=mid，继续进行二分搜索。

（5）搜索结束后，left就是母牛必须跳跃的最短距离的最大值。

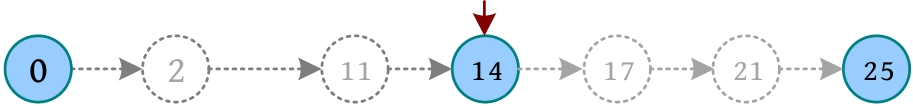
**2. 图解**

（1）根据输入样例，增加开始和结束两块石头，按照到开始节点的距离从小到大排序。



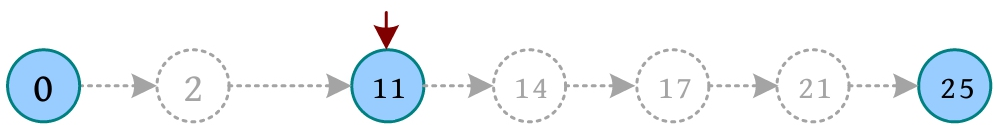
（2）令left=0，right=L=25，right-left>1，mid=(right+left)/2=12，判断是否满足移除两块石头之后，任意间距都不小于12。相当于将3块石头放置在开始位置和结束位置之间，且满足任意间距都不小于12。

用last记录前一块已放置石头的下标，初始时last=0，找第1个与last距离大于或等于12的位置，找到14，放置第1块石头，更新last=3。



继续找第1个与last距离大于或等于12的位置，未找到，说明无法满足条件。缩小距离，令right=mid=12，继续搜索。

（3）left=0，right=12，mid=(right+left)/2=6，判断是否满足移除两块石头之后，任意间距都不小于6。初始时last=0，找第1个与last距离大于或等于6的位置，找到11，放置第1块石头，更新last=2。

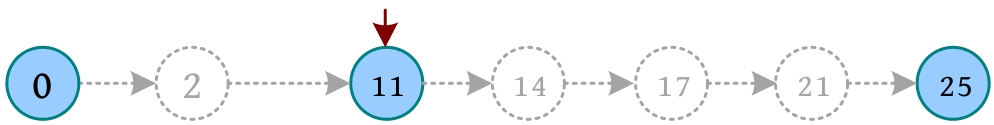


继续找第1个与last距离大于或等于6的位置，找到17，放置第2块石头，更新last=4。

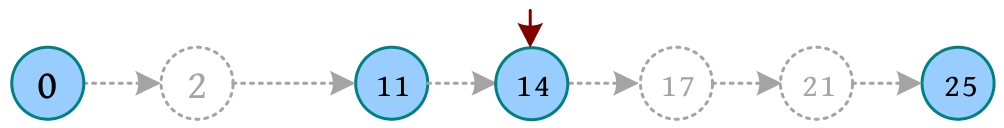


继续找第1个与last距离大于或等于6的位置，未找到，说明无法满足条件。缩小距离，令right=mid=6，继续搜索。

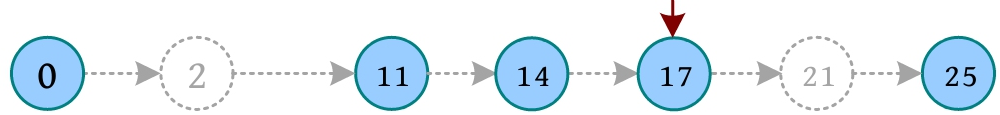
（4）left=0，right=6，mid=(right+left)/2=3，判断是否满足移除两块石头之后，任意间距都不小于3。初始时last=0，找第1个与last距离大于或等于3的位置，找到11，放置第1块石头，更新last=2。



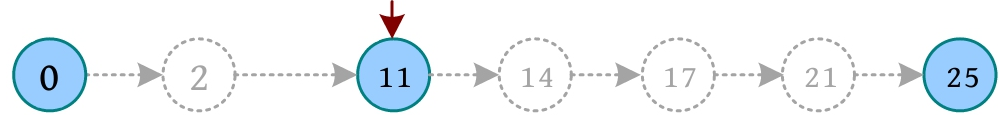
继续找第1个与last距离大于或等于3的位置，找到14，放置第2块石头，更新last=3。



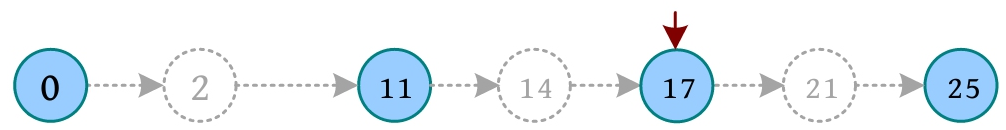
继续找第1个与last距离大于或等于3的位置，找到17，放置第3块石头，可以放置3块石头，满足条件。增加距离，令left=mid=3，继续搜索。



（5）left=3，right=6，mid=(right+left)/2=4，判断是否满足移除两块石头之后，任意间距都不小于4。初始时last=0，找第1个与last距离大于或等于4的位置，找到11，放置第1块石头，更新last=2。

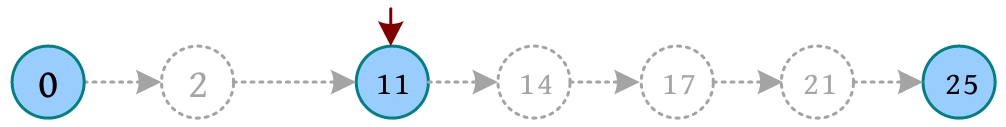


继续找第1个与last距离大于或等于4的位置，找到17，放置第2块石头，更新last=4。

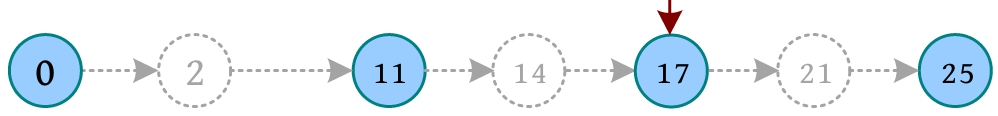


继续找第1个与last距离大于或等于4的位置，找到21，放置第3块石头，可以放置3块石头，满足条件。增加距离，令left=mid=4，继续搜索。

（6）left=4，right=6，mid=(right+left)/2=5，判断是否满足移除两块石头之后，任意间距都不小于5。初始时last=0，找第1个与last距离大于或等于5的位置，找到11，放置第1块石头，更新last=2。



继续找第1个与last距离大于或等于5的位置，找到17，放置第2块石头，更新last=4。

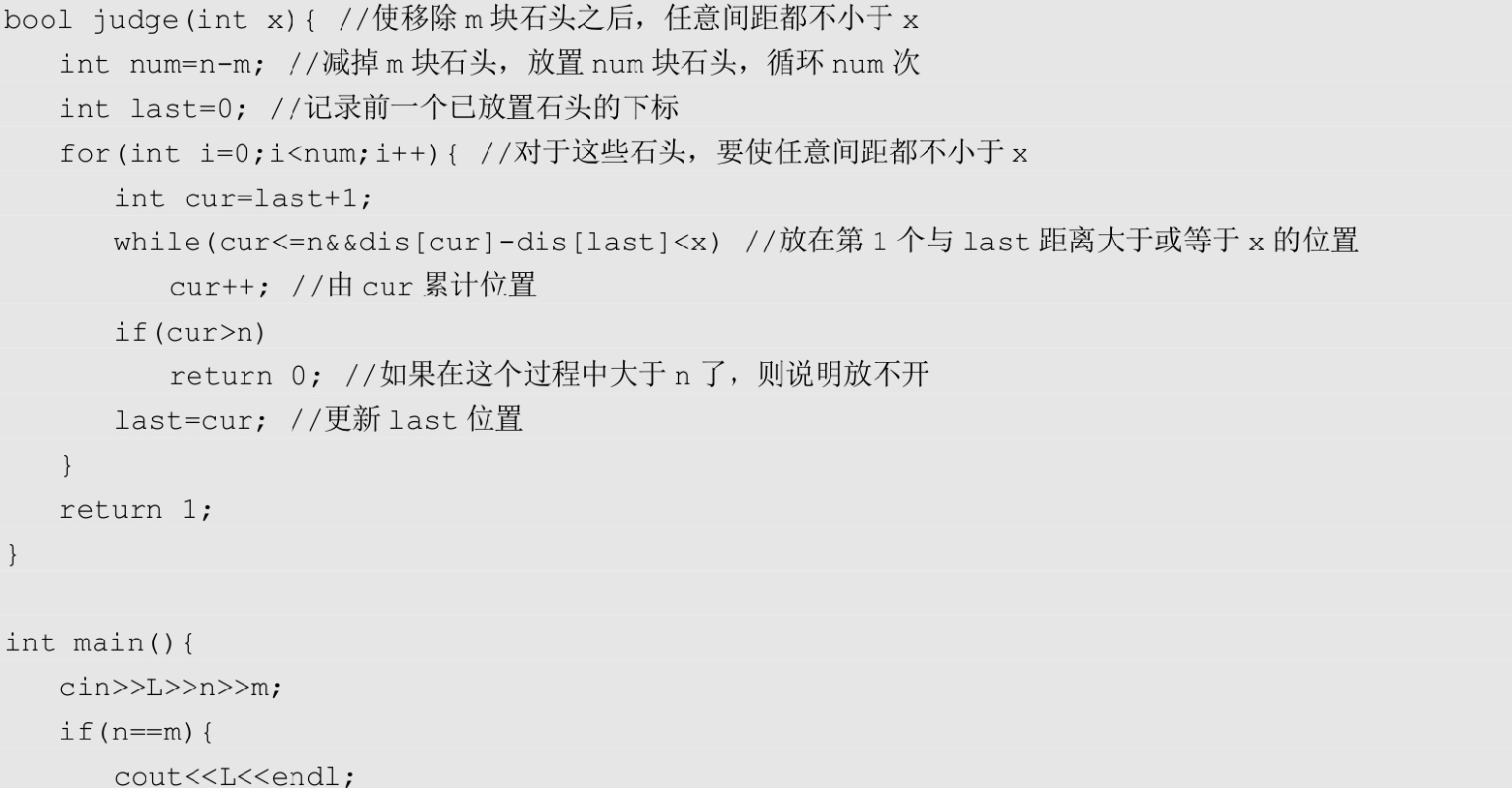


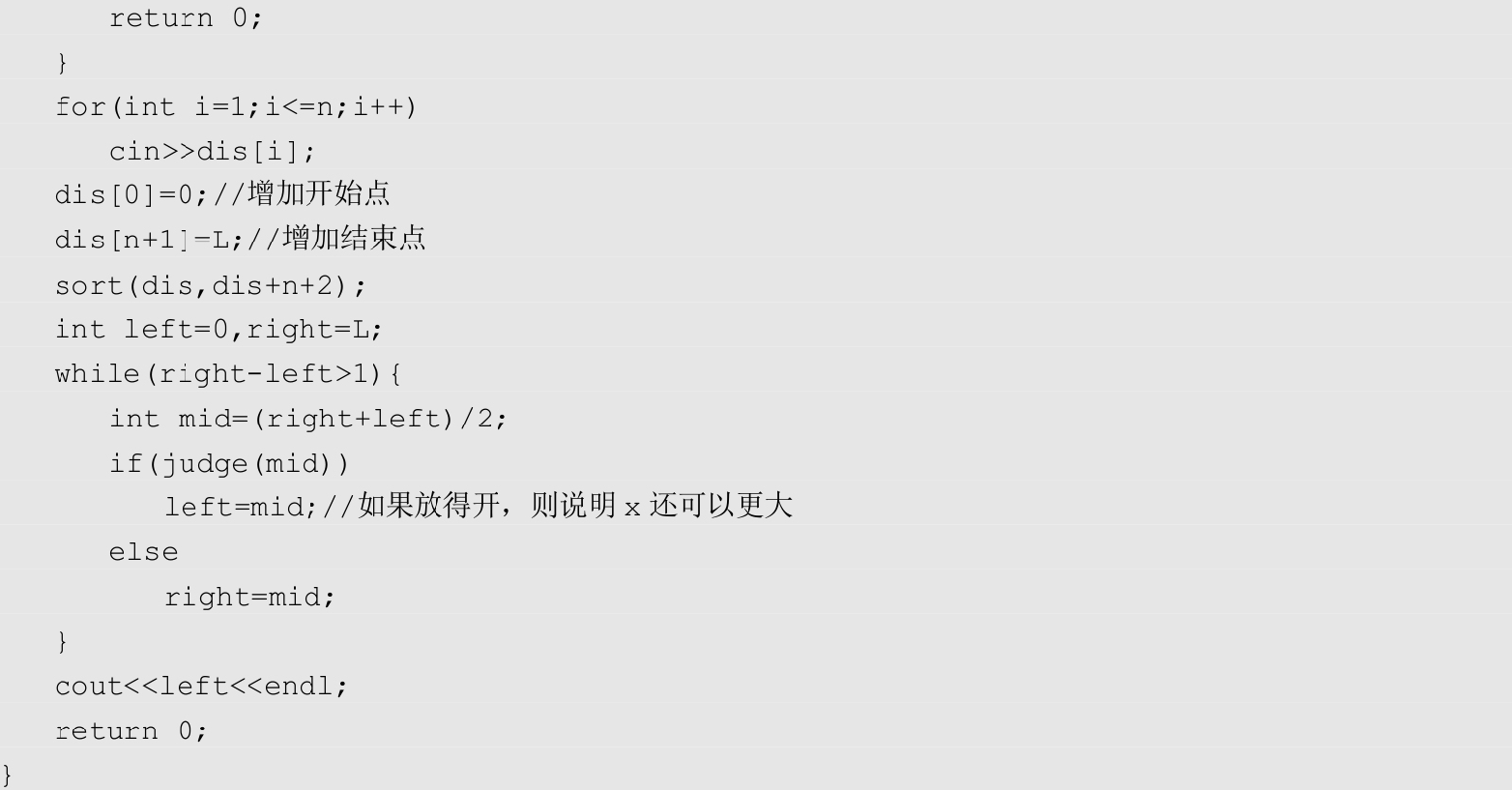
继续找第1个与last距离大于或等于5的位置，未找到，说明无法满足条件。缩小距离，令right=mid=5，继续搜索。

（7）left=4，right=5，此时right-left=1，算法结束，输出答案left=4。

**3. 算法实现**

判断函数相当于将n-m块石头放置在开始位置和结束位置之间，且任意间距都不小于x。



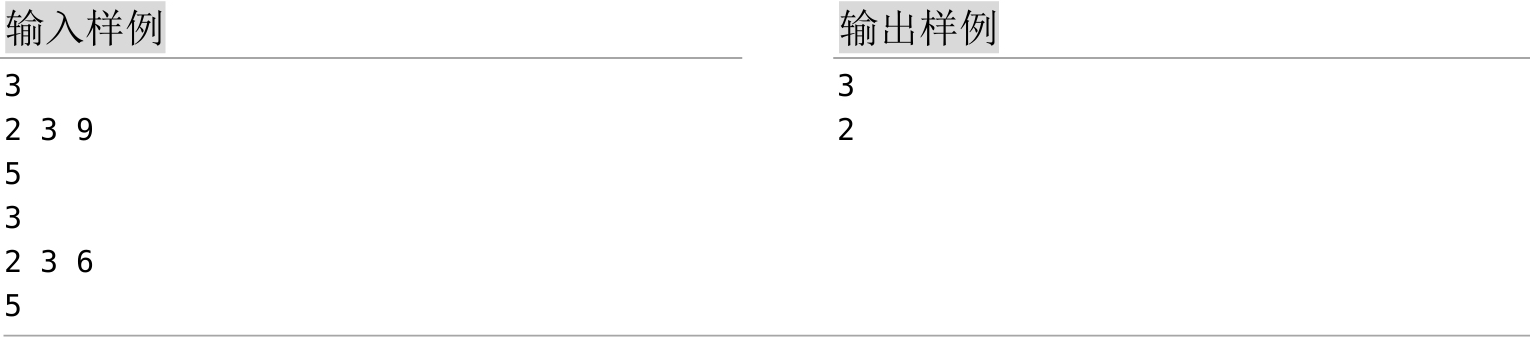


**POJ3104**

**题目描述（POJ3104）：**可以使用散热器烘干衣服。但散热器很小，所以它一次只能容纳一件衣服。简有n件衣服，每件衣服在洗涤过程中都带有ai的水。在自然风干的情况下，每件衣服的含水量每分钟减少1（只有当物品还没有完全干燥时）。当含水量变为零时，布料变干并准备好包装。在散热器上烘干时，衣服的含水量每分钟减少k（如果衣服含有少于k的水，则衣服的含水量变为零）。请有效地使用散热器来最小化烘干的总时间。

**输入：**第1行包含一个整数n（1≤n≤105）；第2行包含ai（1≤ai≤109，1≤i≤n）；第3行包含k（1≤k≤109）。

**输出：**单行输出烘干所有衣服所需的最少时间。



**题解：**假设烘干所有衣服所需的最少时间为mid，如果所有衣服的含水量a[i]都小于mid，则不需要用烘干机，自然风干的时间也不会超过mid。如果有的衣服a[i]大于mid，则让所有a[i]大于mid的衣服使用烘干机，让a[i]不大于mid的衣服自然风干即可。

假设衣服a[i]>mid，用了t时间的烘干机，对剩余的时间mid-t选择自然风干，那么a[i]=k×t+mid-t，t=(a[i]-mid)/(k-1)。只需判断这些a[i]大于mid的衣服使用烘干机的总时间有没有超过mid，如果超过，则不满足条件。

**1. 算法设计**

（1）按照a[i]从小到大排序。

（2）如果k=1，则直接输出a[n-1]，算法结束。

（3）进行二分搜索，l=1，r=a[n-1]，mid=(l+r)>>1，判断最少烘干时间为mid是否可行，如果可行，则r=mid-1，减少时间继续搜索；否则l=mid+1，增加时间继续搜索。当l>r时停止。

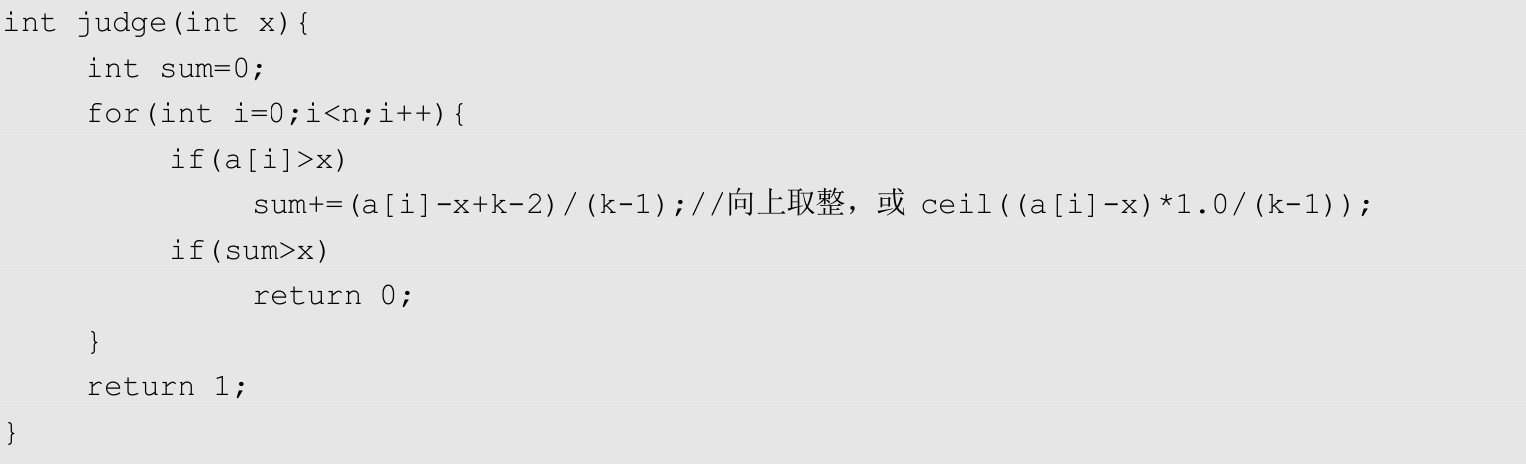
（4）判断最少烘干时间为mid是否可行。对所有a[i]>mid的衣服使用烘干机，用sum累加使用烘干机的时间，如果sum>mid，则说明不可行，返回0。当所有衣服都处理完毕时，返回1。

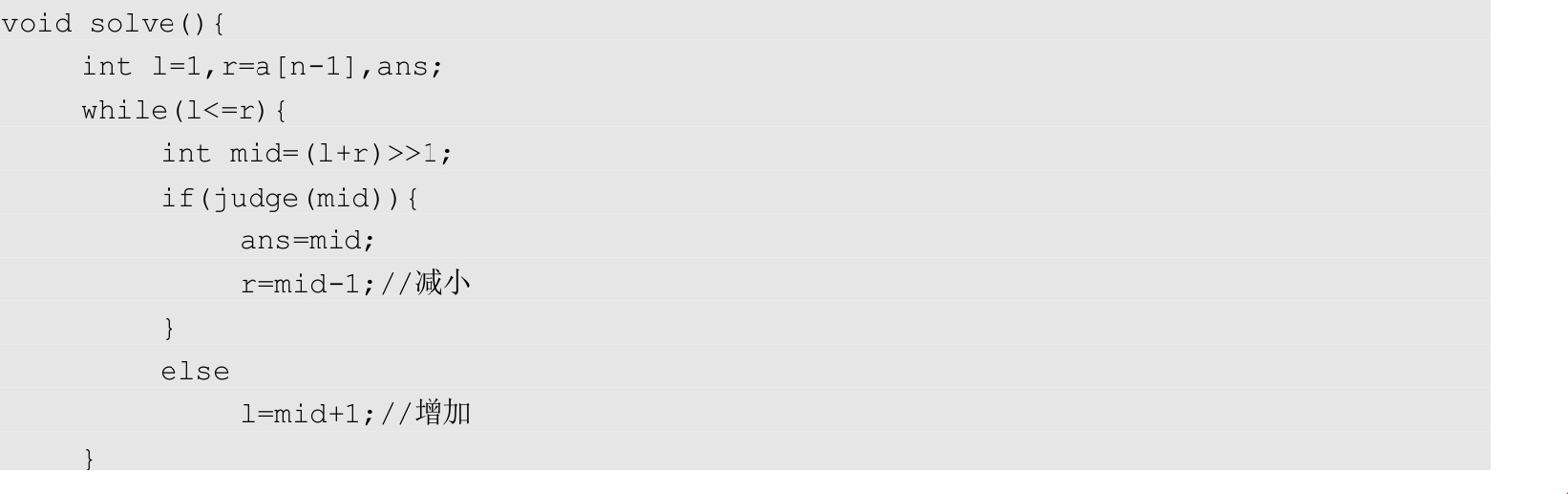
**要特别注意以下事项:**

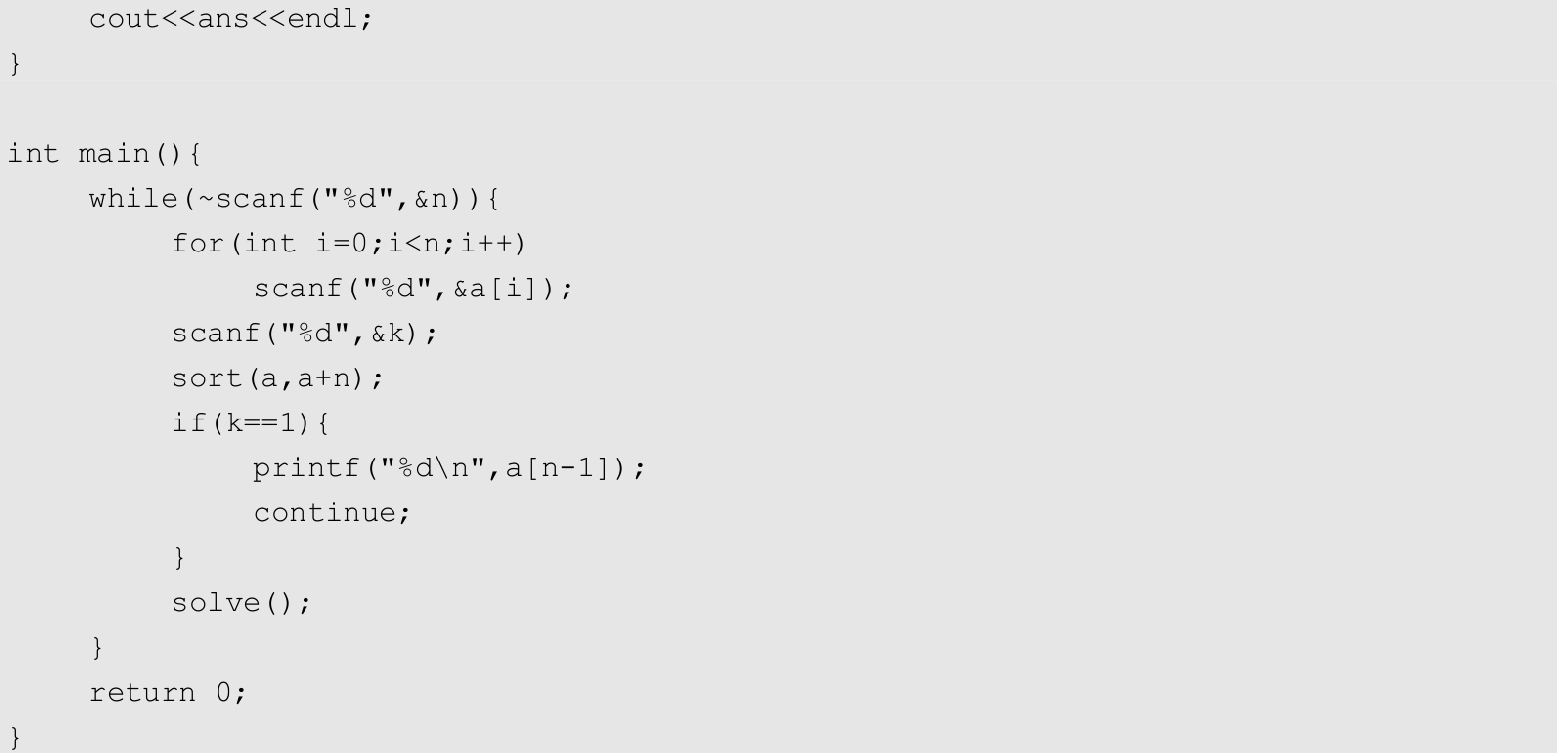
**（1）对t的结果需要向上取整，因为如果有余数，再用一次烘干机无非就是多1个时间，但是如果自然风干，则至少用1个时间。**

**（2）公式中的分母是k-1，因此在k=1时需要单独判断特殊情况，直接输出最大的含水量即可，不然会超时。**

**2. 算法实现**

****

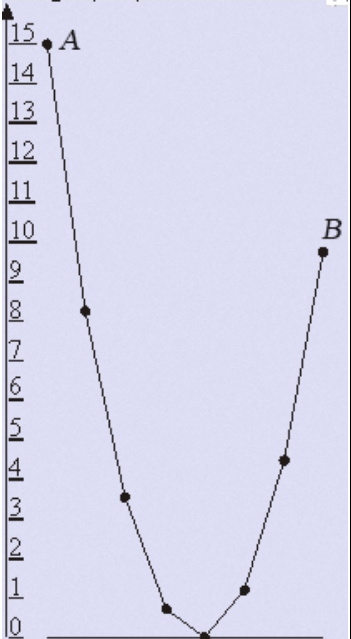
****

****

**POJ1759**

**题目描述（POJ1759）：**新年花环由N个灯组成，每个灯都悬挂在比两个相邻灯的平均高度低1毫米的高度处。最左边的灯挂在地面以上A毫米的高度处。必须确定最右侧灯的最低高度B，以便花环中的灯不会落在地面上，尽管其中一些灯可能会接触地面。灯的编号为1～N，并以毫米为单位表示第i个灯的高度为Hi，推导出以下等式：H1=A；Hi=(Hi-1+Hi+1)/2-1，1<i<N；HN=B；Hi≥0，1≤i≤N。

下图中所示的具有8个灯的花环，A=15和B=9.75。



**输入：**输入包含两个数字N和A。N（3≤N≤1000）表示花环中灯的数量，A（10≤A≤1000）表示地面上最左边的灯的高度（实数，以毫米为单位）。

**输出：**单行输出B，精确到小数点右边两位数，表示最右边灯的最低可能高度。



**题解：**

**1. 算法设计**

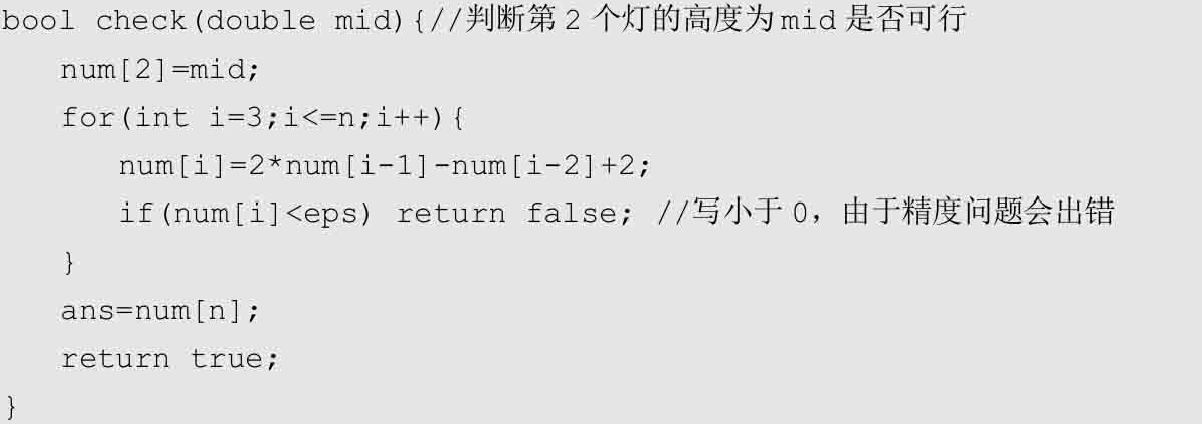
根据高度公式Hi=(Hi-1+Hi+1)/2-1，整理该公式得到Hi+1=2×Hi-Hi-1+2，也可以将其写成当前项与前面两项的关系表达式：Hi=2×Hi-1-Hi-2+2。

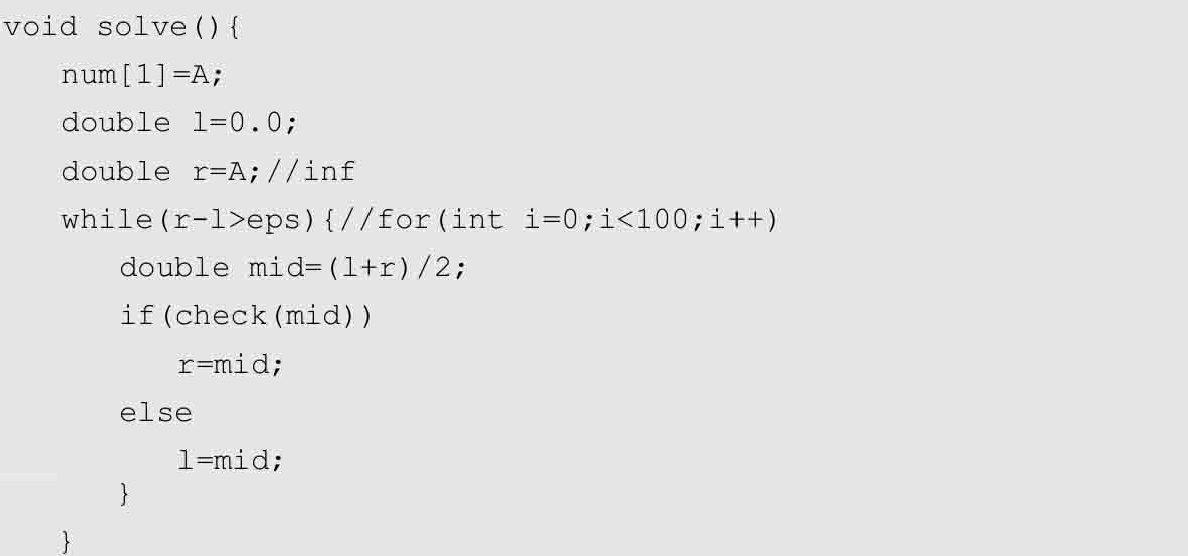
**（1）二分搜索。**初始时，num[1]=A，l=0.0，r=inf（无穷大，通常设为0x3f3f3f3f），mid=(l+r)/2。判断第2个灯的高度为mid是否可行，如果可行，则令r=mid，缩小高度搜索；否则l=mid，增加高度搜索。

**（2）判断mid是否可行。**令num[2]=mid，根据公式从左向右推导，num[i]=2×num[i-1]- num[i-2]+2，i=3…n。如果在推导过程中num[i]<eps，则说明不可行，返回false。**注意不要写小于0，否则由于精度问题会出错**。eps是一个较小的数，例如1e-7。

**（3）可以用r-l>eps判断循环条件，也可以搜索到较大的次数时停止，例如100次，运行100次二分搜索可以达到10-30的精度范围。**实际上对于输入样例，运行43次已经找到答案，为保险起见，尽量执行较多的次数，时间相差不大。

**2. 算法实现**



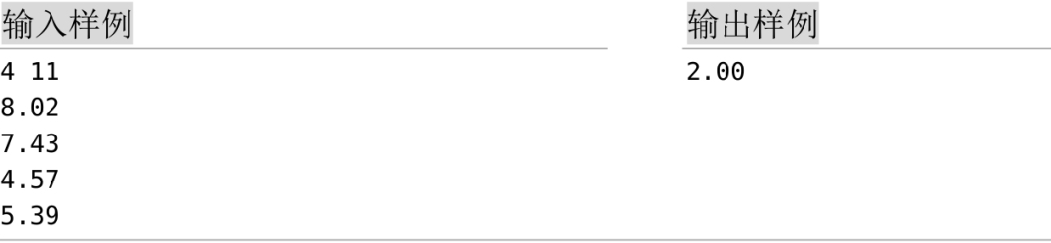


**POJ1064**

**题目描述（POJ1064）：**有N条电缆，长度分别为Li，如何从它们中切割出K条长度相同的电缆，每条电缆最长有多少米。

**输入：**输入的第1行包含两个整数N和K（1≤N,K≤10 000）。N是电缆的数量，K是要求切割的数量。后面是N行，每行一个数字Li（1≤Li≤100 000），表示每条电缆的长度。

**输出：**单行输出电缆切割的最大长度（在小数点后保留两位数字）。如果不能切割所要求数量的电缆，则输出“0.00”（不带引号）。



**题解：**本题求解切割出的K条电缆的最大可能长度，因为一条电缆有可能切割出多条，因此第K条的电缆长度并不是答案。可以假设最大长度为x，采用二分搜索求解答案。

**1. 算法设计**

（1）二分搜索。初始时，l=0.0，r=inf，r也可以被初始化为N条电缆中的最大长度。mid=(l+r)/2，判断切割出来电缆的长度为mid，是否可以切割K条。如果可以，则令l=mid，增加长度搜索，否则r=mid，减少长度搜索。

（2）判断mid是否可行。枚举N条电缆，累加每条电缆可以切割出的数量，注意该数量要取整(int)(L[i]/mid)，如果数量大于或等于K，则表示可行。

（3）可以用r-l>eps判断循环条件，也可以在搜索较大的次数时停止，例如100次。结束时返回l。

（4）输出答案。本题要求保留两位小数，切割后不可四舍五入，因此可以扩大100倍取下限，然后缩小100倍，舍去2位小数之后的数字。但是存在特殊情况，例如1.599 999 99，这样的数近似于1.60，可以加上一个特别小的数处理该问题，因此返回答案ans加上eps（1e-7）即可。还有一种解决办法是直接返回r作为答案，因为循环条件r-l>eps，r比l大eps。

**2. 算法实现**

