题目要求：<32-P2678-跳石头-题目要求.docx>

## 分析

这道题如果要使用暴力搜索直接求解会严重超时。实际上，我们可以发现，这个所谓的最短跳跃距离显然不能超过一个范围，而这个范围题目上已经给了出来。也就是说，答案是有一个确定的范围限制的，我们就可以考虑一种另外的方法去解决——枚举答案，并去验证答案是否可行。

实际上，枚举答案有时候也会超时。这就好比说你要从一本英汉词典上查一个单词，你从头到尾一页一页的翻着找，这样找可以保证一定能找到，但是最坏情况你要把整本词典都翻一遍，那就麻烦了。

有什么改进的方法吗？当然有。

考虑把这个词典从中间分开，看一下中间那一页的主要单词都是啥，然后去判断我要找的单词应该在左半部分还是右半部分，再去那一部分考虑怎么找就好了。同样的，在另一部分也是要进行划分并且判断的操作。这样一直进行下去，便能很快的找到答案，而且根本不需要翻过整个词典来。

可以证明，如果一页一页的找，最多要找n次，但是用这个方法，最多找floor(log2n)次。

我们把这个方法叫做“二分答案”。顾名思义，它用二分的方法枚举答案，并且枚举时判断这个答案是否可行。但是，二分并不是在所有情况下都是可用的，使用二分需要满足两个条件。一个是有界，一个是单调。

二分答案应该是在一个单调闭区间上进行的。也就是说，二分答案最后得到的答案应该是一个确定值，而不是像搜索那样会出现多解。二分一般用来解决最优解问题。刚才我们说单调性，那么这个单调性应该体现在哪里呢？

可以这样想，在一个区间上，有很多数，这些数可能是我们这些问题的解，换句话说，这里有很多不合法的解，也有很多合法的解。我们只考虑合法解，并称之为可行解。考虑所有可行解，我们肯定是要从这些可行解中找到一个最好的作为我们的答案， 这个答案我们称之为最优解。

最优解一定可行，但可行解不一定最优。我们假设整个序列具有单调性，且一个数x为可行解，那么一般的，所有的x'(x'<x)都是可行解。并且，如果有一个数y是非法解，那么一般的，所有的y'(y'>y)都是非法解。

那么什么时候适用二分答案呢？注意到题面：使得选手们在比赛过程中的最短跳跃距离尽可能长。如果题目规定了有“最大值最小”或者“最小值最大”的东西，那么这个东西应该就满足二分答案的有界性（显然）和单调性（能看出来）。

那就好办了。我们二分跳跃距离，然后把这个跳跃距离“认为”是最短的跳跃距离，然后去以这个距离为标准移石头。使用一个judge判断这个解是不是可行解。如果这个解是可行解，那么有可能会有比这更优的解，那么我们就去它的右边二分。为什么去右边？答案是，这个区间是递增的 ，而我们求的是最短跳跃距离的最大值，显然再右边的值肯定比左边大，那么我们就有可能找到比这更优的解，直到找不到，那么最后找到的解就有理由认为是区间内最优解。反过来，如果二分到的这个解是一个非法解，我们就不可能再去右边找了。因为性质，右边的解一定全都是非法解。那么我们就应该去左边找解。整个过程看起来很像递归，实际上，这个过程可以递归写， 也可以写成非递归形式，我个人比较喜欢使用非递归形式。

下一个问题，这个judge怎么实现呢？judge函数每个题有每个题的写法，但大体上的思想应该都是一样的——想办法检测这个解是不是合法。拿这个题来说，我们去判断如果以这个距离为最短跳跃距离需要移走多少块石头，先不必考虑限制移走多少块，等全部拿完再把拿走的数量和限制进行比对，如果超出限制，那么这就是一个非法解，反之就是一个合法解，很好理解吧。

可以去模拟这个跳石头的过程。开始你在i(i=0)位置，我在跳下一步的时候去判断我这个当前跳跃的距离，如果这个跳跃距离比二分出来的mid小，\*\*那这就是一个不合法的石头，应该移走。\*\*为什么？我们二分的是最短跳跃距离，已经是最短了，如果跳跃距离比最短更短岂不是显然不合法，是这样的吧。移走之后要怎么做？先把计数器加上1，再考虑向前跳啊。去看移走之后的下一块石头，再次判断跳过去的距离，如果这次的跳跃距离比最短的长，那么这样跳是完全可以的，我们就跳过去，继续判断，如果跳过去的距离不合法就再拿走，这样不断进行这个操作，直到i = n+1，为啥是n+1？河中间有n块石头，显然终点在n+1处。（这里千万要注意不要把n认为是终点，实际上从n还要跳一步才能到终点）。

模拟完这个过程，我们查看计数器的值，这个值代表的含义是我们以mid作为答案需要移走的石头数量，然后判断这个数量 是不是超了就行。如果超了就返回false，不超就返回true。

整道题我已经说完了。。。如果实在难以理解，请看代码。

1. #include <iostream>
2. #include <cstdio>
3. #include <cstring>
4. #include <cctype>
5. #define maxn 500010
6. using namespace std;
7. int d,n,m;
8. int a[maxn];
9. int l,r,mid,ans;
10. inline int read(){*//我喜欢快读*
11. int num = 0;
12. char c;
13. bool flag = false;
14. while ((c = getchar()) == ' ' || c == '\n' || c == '\r');
15. if (c == '-') flag = true;
16. else
17. num = c - '0';
18. while (isdigit(c = getchar()))
19. num = num \* 10 + c - '0';
20. return (flag ? -1 : 1) \* num;
21. }
22. bool judge(int x){*//judge函数，x代表当前二分出来的答案*
23. int tot = 0;*//tot代表计数器，记录以当前答案需要移走的实际石头数*
24. int i = 0;*//i代表下一块石头的编号*
25. int now = 0;*//now代表模拟跳石头的人当前在什么位置*
26. while (i < n+1){*//千万注意不是n，n不是终点，n+1才是*
27. i++;
28. if (a[i] - a[now] < x)*//判断距离，看二者之间的距离算差值就好*
29. tot++;*//判定成功，把这块石头拿走，继续考虑下一块石头*
30. else
31. now = i;*//判定失败，这块石头不用拿走，我们就跳过去，再考虑下一块*
32. }
33. if (tot > m)
34. return false;
35. else
36. return true;
37. }
38. int main(){
39. d = read();*//d代表总长度，也就是右边界*
40. n = read();*//n块石头*
41. m = read();*//限制移走m块，思考的时候可别被这个m限制*
42. for (int i=1;i<=n;i++)
43. a[i] = read();
44. a[n+1] = d;*//敲黑板划重点，再强调一遍，n不是终点*
45. l = 1;*//l和r分别代表二分的左边界和右边界*
46. r = d;
47. while (l <= r){*//非递归式二分正常向写法，可理解为一般框架*
48. mid = (l+r) / 2;*//这再看不出是啥意思可以退群了*
49. if (judge(mid)){*//带入judge函数判断当前解是不是可行解*
50. ans = mid;
51. l = mid + 1;*//走到这里，看来是可行解，我们尝试看看是不是有更好的可行解*
52. }
53. else
54. r = mid - 1;*//噫，你找了个非法解，赶紧回到左半边看看有没有可行解*
55. }
56. cout << ans << endl;*//最后的ans绝对是最优解*
57. return 0;
58. }

## 分析2

题目的意思很简单，就是最多移走m块石头，使得选手跳跃的最短距离最大即可。

本人暴力枚举的时候，自己编译时，发现只有样例过了，自己编的数据一个都没过（内心有点酸~~~），后来才发现其实是可以二分答案的，原因是题目要求找到最短距离的最大值，这种题目用二分做最合适。

主要思路 ：从起点出发，先选定一段距离mid，若前面的石头B与你所站着的石头A的距离小于mid，就把B搬掉，记录一下；如果不，就把B留下，再跳到石头B上。照这个步骤多次循环后，如果搬掉的石头多了，就把距离mid定小点；如果少了，就把mid定大点。

1. #include<bits/stdc++.h>
2. using namespace std;
3. int main(){
4. int l,n,m,left,right,mid,now,i,a[50005],s,ans;
5. cin>>l>>n>>m;
6. for(i=1;i<=n;i++)
7. cin>>a[i];   *//输入*
8. left=0;right=l;
9. while(left<=right){
10. mid=(right+left)/2;
11. now=0;s=0;
12. for(i=1;i<=n;i++){
13. if(a[i]-a[now]<mid)
14. s++;
15. else now=i;}
16. if(s<=m){
17. ans=mid;
18. left=mid+1;}
19. else right=mid-1;}
20. cout<<ans;
21. return 0;
22. }

## 分析3

此题无论是题目，还是代码，思路都非常简洁明了。一道很好的又很经典的二分查找题。

首先假如枚举，那不用说，直接TLE。

然而很多人就卡在如何二分上面。这就非常奇怪了，因为一旦理解了暴力的判断是如何达到的，那二分也就一目了然了。

首先将石头位置排个序，以便处理方便。这一步也是必须要做的，因为若不排序，那判断将非常的困难。而下面这一步，也是此题之精髓——

我们对于一个长度x，想看看它是否可以符合删除石头数小于等于m，可以这样做：

从位置的小到大扫遍所有石头，用一个变量存储上一个跳到的点。第一个与这上一个点的距离大于等于x的石头即是下一个跳到的点。这里用了一点贪心的思想：因为如果不跳到第一个符合条件的点上，那么整个队列的稀疏度就会提高，最终需要删除的石头也会更多。因为我们要取最优状态，所以要保证跳过的石头数最少。当然，如果某个石头到终点的距离小于x，那它不能被统计到——所以得删去后面这些无法跳到的石头。我自认为这应该也是一个坑点（虽然我第一遍就判断了）。

这样，便求出了这个x是否可行，如果可行，那就往右边二分，但要记得范围要包括x；若不行，则往左边二分，右限制不包括x。然后，二分到左右边界相等，输出即可。

1. #include<cstdio>
2. #include<algorithm>
3. using namespace std;
4. int sto[100000];*//开大一点，保险*
5. int main()
6. {
7. int s,n,m;
8. scanf("%d%d%d",&s,&n,&m);
9. int zuo=1,you=s,mid;*//所有边界为1、s*
10. for(int i=0;i<n;i++)scanf("%d",&sto[i]);
11. sort(sto,sto+n);*//从小到大排序*
12. int sg,cnt,ii;
13. while(zuo!=you)
14. {
15. mid=(zuo+you+1)>>1;*//位运算加速*
16. sg=cnt=0;*//初始化*
17. for(ii=0;ii<n;ii++)
18. {
19. if(s-sto[ii]<mid)break;*//如解析中所述，若再跳x已超过终点，则不可取此点，它后面的也显然不可取*
20. if(sto[ii]-sg<mid)cnt++;*//跳过*
21. else sg=sto[ii];*//贪心，直接跳到*
22. }
23. cnt+=n-ii;*//统计最后被删除的点数*
24. if(cnt<=m)zuo=mid;
25. else you=mid-1;*//二分边界更新，具体请见解析*
26. }
27. printf("%d",zuo);*//输出*
28. return 0;
29. }

## 我的代码：

附上文件路径：<32-P2678-跳石头.cpp>

1. #include<stdio.h>
2. #define N 50005
3. int len,n,m;
4. int main(void)
5. {
6. scanf("%lld%d%d",&len,&n,&m);
7. int map[N],dis[N];
8. map[0]=0;
9. map[n+1]=len;
10. for(int i=1;i<=n;i++)
11. scanf("%d",&map[i]);
12. int left=0,right=len,mid,check=0;
13. while(left<right-1)
14. {
15. check=0;
16. mid=(left+right)/2;
17. for(int i=1;i<=n+1;i++)
18. dis[i]=map[i]-map[i-1];
19. for(int i=1;i<=n+1;i++)
20. {
21. if(dis[i]<mid)
22. {
23. dis[i+1]+=dis[i];
24. check++;
25. }
26. }
27. if(check<=m) left=mid;
28. else right=mid;
29. }
30. if(n==0) printf("%d",right);
31. else printf("%d",left);
32. }