青蛙过河

1.1 问题描述

在河上有一座独木桥，一只青蛙想沿着独木桥从河的一侧跳到另一侧。在桥上有一些石子，青蛙很讨厌踩在这些石子上。由于桥的长度和青蛙一次跳过的距离都是正整数，我们可以把独木桥上青蛙可能到达的点看成数轴上的一串整点：0,1,...,L（其中 L 是桥的长度）。坐标为 0 的点表示桥的起点，坐标为 L 的点表示桥的终点。青蛙从桥的起点开始，不停的向终点方向跳跃。一次跳跃的距离是 S 到 T 之间的任意正整数（包括 S,T）。当青蛙跳到或跳过坐标为 L 的点时，就算青蛙已经跳出了独木桥。

题目给出独木桥的长度 L，青蛙跳跃的距离范围 S,T，桥上石子的位置。你的任务是确定青蛙要想过河，最少需要踩到的石子数。

对于 30%的数据，L<=10000

对于全部的数据，L<=109

输入格式

输入的第一行有一个正整数 L（1 <= L <= 10^9），表示独木桥的长度。第二行有三个正整数 S，T，M，分别表示青蛙一次跳跃的最小距离，最大距离，及桥上石子的个数，其中 1 <= S <= T <= 10，1 <= M <= 100。第三行有 M 个不同的正整数分别表示这 M 个石子在数轴上的位置（数据保证桥的起点和终点处没有石子）。所有相邻的整数之间用一个空格隔开。

输出格式

输出只包括一个整数，表示青蛙过河最少需要踩到的石子数。

样例输入

```
10
2 3 5
2 3 5 6 7
```

样例输出

```
2
```

1.2 问题分析

路径压缩法，虽然桥很长，但上面最多只有 100 个石子，想到能否用石子 DP，而应该是不行的。由于石子排布非常的疏，我们还会发现，如果两个石子相隔甚远，那他们中间的 f[i]大部分将会是同一个数，能否把两个石子的距离缩短，使之还与原来等效？要是行的话怎么缩？王乃岩同学考试时做了一个方法能够过全部数据，用的滚动数组存储，下面列出了他的程序。我自己也写了个程序，和他不尽相同：我令 L=stone[i]-stone[i-1]（stone[i]代表按坐标由小到大顺序排列的石块坐标），当 L 能够被 t 整除时（L%t==0），令 k=t；当 L 不能被 t 整除时（L%t!=0），令 k=L%t。然后令 k+=t，最后判断如果 k>L，那么 map[]中 stone[i]和 stone[i-1]两石头的距离就被等效成为 L（也就是没变）；如果 k<=L，那么 map[]数组中 stone[i]和 stone[i-1]两石头的距离就被等效成为 k，可以看出来，这样处理完，两石子最大间距为 2*t，大大的缩短了数组，再按解一进行 DP，就可以通过了。

1.3 算法实现

```
#include <stdio.h>
```

```
#include <string.h>
long stone[101];
int map[100001];
int f[100001];
long L;
int S,T,M;

void quickSort(int l, int r){
  int i=l,j=r;
  long temp=stone[i];
  while(i<j){
    while(i<j && stone[j]>temp) j--;
    if(i<j){
      stone[i]=stone[j]; i++;
    }
    while (i<j && stone[i]<temp) i++;
    if(i<j){
      stone[j]=stone[i]; j--;
    }
  }
  stone[i]=temp;
  if (i-1>l) quickSort(l,i-1);
  if (i+1<r) quickSort(i+1,r);
}

int main(){
  int i,j;
  long l,k,p=0,min;
  scanf("%ld%d%d%d",&L,&S,&T,&M);
  for (i=1;i<=M;i++) scanf("%ld",&stone[i]);
  memset(map,0,sizeof(int)*100001);
  memset(f,0,sizeof(int)*100001);
  quickSort(1,M);
  stone[0]=0;
  p=0;
  for(i=1;i<=M;i++){
    l=stone[i]-stone[i-1];
    if(l%T==0) k=T;
    else k=l%T;
    k=k+T;
    if(l<k) k=l;
    p=p+k;
    map[p]=1;
  }
```

```
  for(i=1;i<=p+T;i++){
    min=1000;
    for(j=i-T;j<=i-S;j++) if(j>=0 && f[j]<min)
        min=f[j];
    f[i]=min+map[i];
  }
  min=1000;
  for(i=p+1;i<=p+T;i++) if(f[i]<min)
      min=f[i];
  printf("%d\n",min);
  return 0;
}
```

1.4 复杂性分析

简单动态规划，f[i]代表青蛙跳到 i 点时所可能踩到的最少石子数，所以有 f[i]=min{f[k]+map[i]}(i-s≤k≤i-t)，其中 map[i]代表 i 上是否有石子，有是 1，否则 0。算法复杂度 O(n^2)。

路径压缩的 dp

路径压缩方法：

设 s 与 t 的最小公倍数为 LCM,则

当相邻的两粒石子距离 S>LCM 时

情况与相距 LCM+(S%LCM)情况一样

dp 就很简单了

dp[i]表示跳到 i 个位置（压缩后的）最少踩到的石子数

则有 dp[i]=min{dp[i-j} + f[i], s<=j<=t

当第 i 格有石子时 f[i]=1 否则 f[i]=0

我们不难发现其实这题可以应用单调队列（虽然没有必要）

目前我的方法是用一个缓存队列保存 i-s+1 至 i 格的情况，每次将第 i 格算出后加入缓存，并将缓存中第 i-s+1 格入单调队列

这样时间复杂度由 O(n^2)降至 O(n)

1.5 总结

这道题在没看数据规模之前以为是一道简单的 DP，但是数据开到十亿，无论在时间还是空间复杂度都过大，所以就要进行优化了。选择简单的动态规划方法会使复杂度太高，所以我选择了路径压缩法，调用队列使复杂度大大降低。