**南京信息工程大学算法设计与分析（实习）报告**

实验名称 日期 2023.6 得分 指导教师 赵振杰 专业 计科

学院 计网院 年级 2021 班次 5 姓名 刘祥宇 学号 202183290006

1. **实验目的**
2. 1．实验目的
3. （1）熟悉动态规划算法
4. （2）理解动态规划算法求解问题的一般性过程
5. （3）针对特定问题，设计并编写动态规划程序
6. （4）能够分析动态规划算法的时间复杂度
7. **实验内容**
8. wzy在跑步，由于昨天下雨了，所以跑道上某些位置堆积了雨水。wzy并不希望自己踩到水坑，所以当他将要踩到水坑时候他会选择跳过去。
9. 具体的，跑道可以抽象为长度为n的序列，wzy从位置1出发，目的地是到达位置n。跑道上会有m个位置存在积水，他每次可以跳1到k单位长度正整数的距离，问wzy最少可以踩到几次水坑到达位置n。
10. 输入描述
11. 输入第一行三个正整数n，m，k，分别代表路径的长度，积水的个数以及wzy一次最远可以跨多远。
12. 第二行共m个正整数ai，分别代表第i个水坑的位置。
13. 输出描述
14. 输出共一行，一个正整数，表示wzy至少踩到几次水坑才能到达位置n。
15. 样例输入
16. 6 3 2
17. 2 3 5
18. 样例输出
19. 1
20. **实验步骤和结果**

#include<iostream>

using namespace std;

int n,m,k,T;

int a[10001],dp[10001];

int main(){

cin>>T;

while(T--){

while(cin>>n>>m>>k){

for(int i=1;i<=n;i++)

{

dp[i]=99999;

a[i]=0;}

for(int i=1;i<=m;i++)

{

int j;

cin>>j;

a[j]=1;

}//

if(a[1]==1)

dp[1]=1;

else

dp[1]=0;

for(int i=1;i<=n;i++)

{

for(int j=1;j<=k;j++)

{

if(i+j>n)

break;

if(dp[i]+a[i+j]<dp[i+j])

dp[i+j]=dp[i]+a[i+j];

}

}

//for(int i=1;i<=n;i++)

//cout<<dp[i]<<" ";

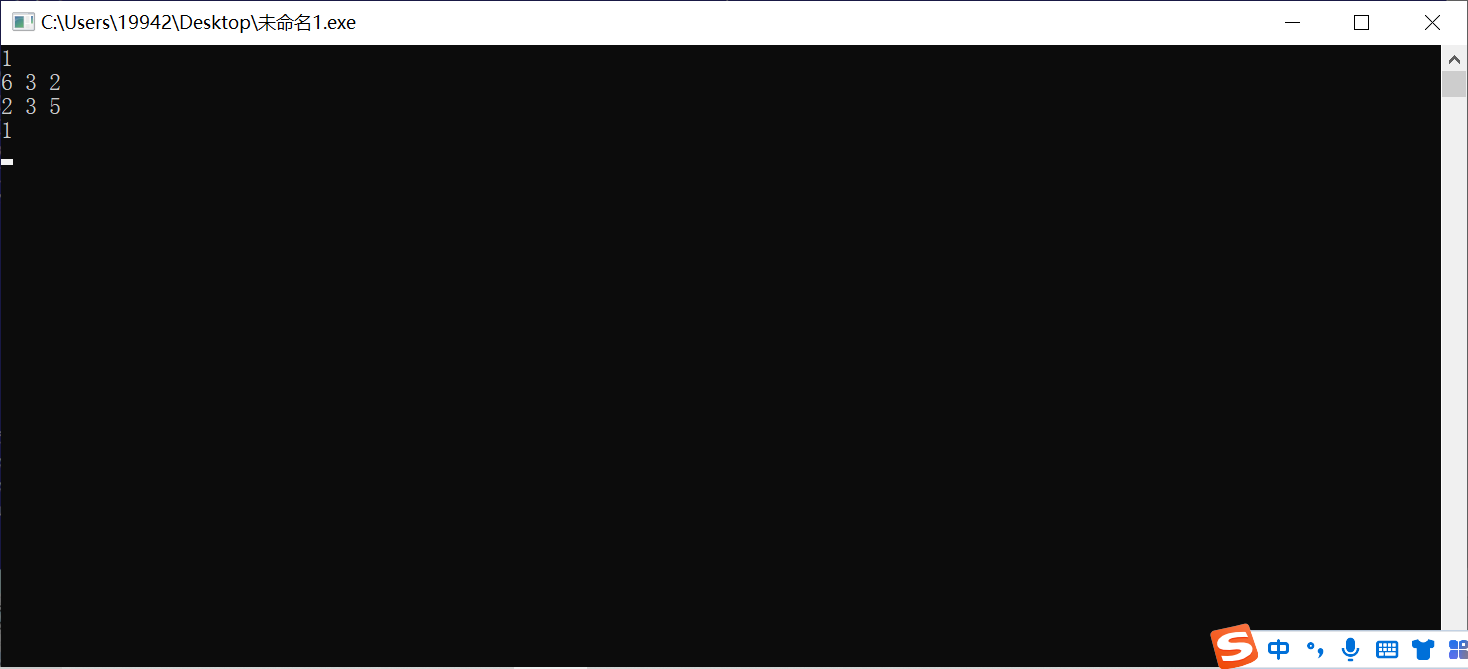
//cout<<endl;

cout<<dp[n]<<endl;}

}

return 0;

}



1. **总结**

整体思路如下

如果位置1（第一个点）有水塘，找到第一个无水塘的地方right，如果位置n（最后一个点）有水塘，找到最后一个无水塘的地方left，令begin=right，end=left（可以画个图理解一下）。此时整条路被分成了三段，中间两个断点分别是right（begin）和left（end)

易得：从位置i1到i2（i1，i2干燥，之间全是水塘）需要踩水塘的次数为：

（i2-i1-1）/k

对于第一段道路，若已知第一点有水塘，则wzy先到第一个点，此时踩了一次水塘，然后将位置1作为出发点，假设位置1无水塘（因为已经记录过踩水塘次数+1），找到第一个实际干燥的地方right，所需要经历的踩水塘次数为count+=（right-1-1）/k+1，这里的第一个1则表示假设干燥的位置1，最后的+1则表示踩了第一个水塘。第三段尾部同理，count+=（n-left-1）/k+1。

接下来是中间道路，赋予了新的begin和end（在上文），这两个点都是干燥的，因为begin和end的初始值分别为1和n，所以即使位置1和位置n也是干燥的，也能轻松合并情况。找到第一个干燥点后，下标赋值为left，往后遍历，找到第二个干燥点，下标赋值为right，利用公式，count+=（right-left-1）/k；然后将right赋值给left，再找left的下一个right，以此类推

算法时间复杂度为O(n）