



中国计算机学会
China Computer Federation



链表基本操作及应用

重庆育才中学 周祖松



主要内容

- 1、链表基本操作
- 2、链表快速查找——跳跃表
- 3、链表在搜索中的应用——舞蹈链
- 4、链表在解题中的应用举例



链表的特点

- 用一组任意的存储单元存储线性表的数据元素
- 利用指针实现了用不相邻的存储单元存放逻辑上相邻的元素
- 每个数据元素 a_i ，除存储本身信息外，还需存储其直接后继的信息
- 结点
 - 数据域：元素本身信息
 - 指针域：指示直接后继的存储位置

结点

| | |
|-----|-----|
| 数据域 | 指针域 |
|-----|-----|

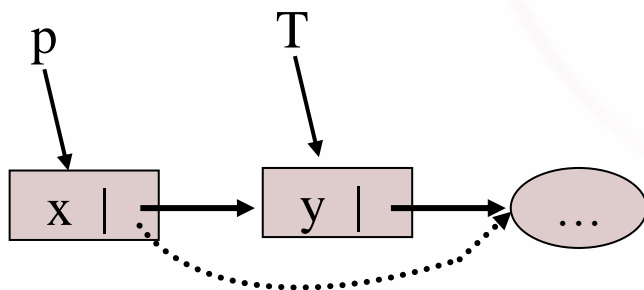
单向链表删除操作图示

是删除节点p的下一个节点T，保证p和T是存在的。

$p^{next} := T^{next};$

Dispose(T);

//竞赛中常常省略这一句



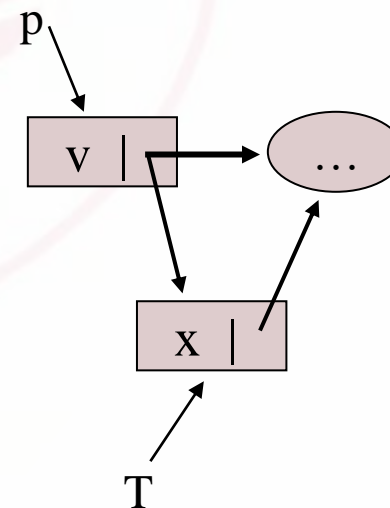
单向链表插入操作图示

在节点p之后插入一个数x，保证p是存在的。

new(T); $T.v = x;$

$T^{next} := p^{next};$

$p^{next} := T;$

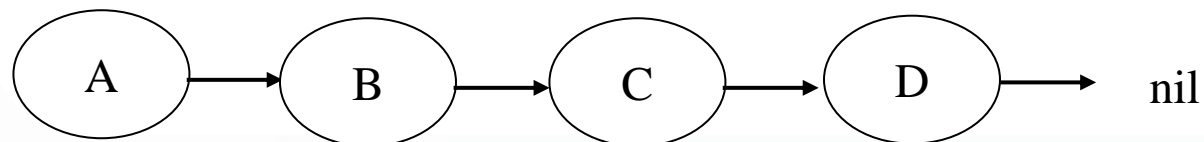




- 1、先开一个足够大的数组，即先静态产生足够的节点。
- 2、取一个节点即为取一个数组元素。
- 3、“指针”的“地址”现在是数组的下标，即数组的第几个元素。
- 4、“指针”为数组下标值，一般为整数型。

| | | | | | | | |
|---|---|---|----|---|---|---|-----|
| A | . | B | D | . | . | C | ... |
| 3 | . | 7 | -1 | . | . | 4 | ... |

| | | | | | | | |
|---|---|---|---|---|---|---|-----|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | ... |
|---|---|---|---|---|---|---|-----|





链表的“数组实现”样例

题目描述：

无聊时，东东写了个 $N(<10^6)$ 位的正整数，然后从开头删除数字，但并不是连续删除，而是根据刚被删除的数字的值 x 向后跳 x 个数字，再去删除之后的第 $x+1$ 个数字。如果数到数字的尾部，则环绕到开头。

请问最后被删数字是什么？如：数字221034

221034 \rightarrow 21034 \rightarrow 2134 \rightarrow 214 \rightarrow 21 \rightarrow 1

输入格式：

第一行一个整数 N

第二行一个长为 N 的数字串

输出格式：

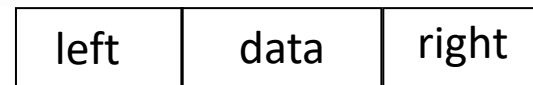
一个整数



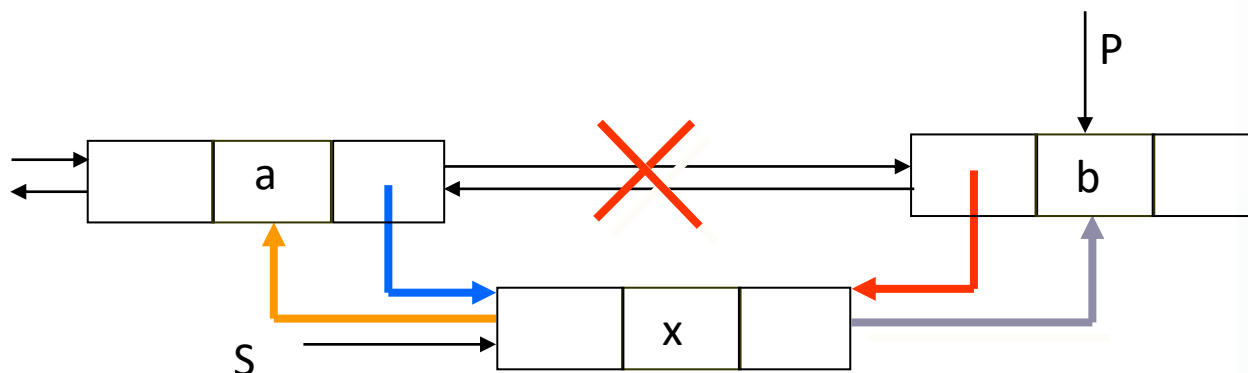
```
//开两个数组，next中记录的是下一个节点“指针”
int d[1000000],next[1000000],n,now; char c;
int main() {
    cin>>n;
    for (int i=0; i<n; i++){
        cin>>c; d[i]=c-'0';
        next[i]=i+1; //初始化“指针”
    }
    next[n-1]=0; //环绕—循环链表
    now=n-1; //第一个被删除数的前面“指针”
    for(int i=0; i<n-1; i++) {
        int x=d[next[now]]; //是删除的数
        next[now]=next[next[now]]; //删除一个节点
        for(;x>0;x--) now=next[now]; //向后移动x个节点
    }
    cout<<(d[next[now]]); //打印最后一个节点
    return 0;
}
```

双向链表

❖ 定义



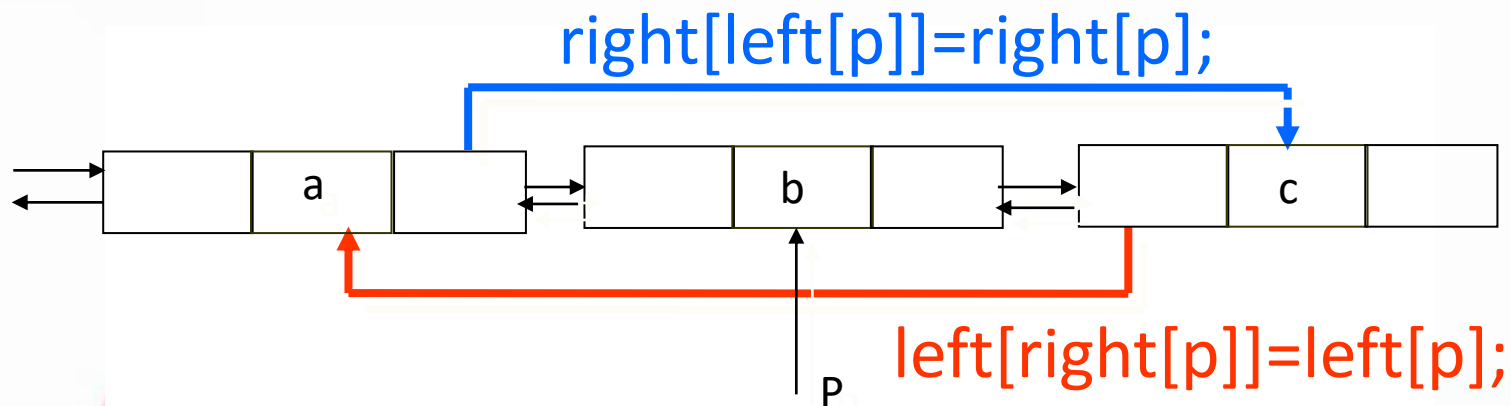
❖ 插入



```
data[s]=x;  
left[s]=left[p];  
right[left[p]]=s;  
right[s]=p;  
left[p]=s;
```


双向链表

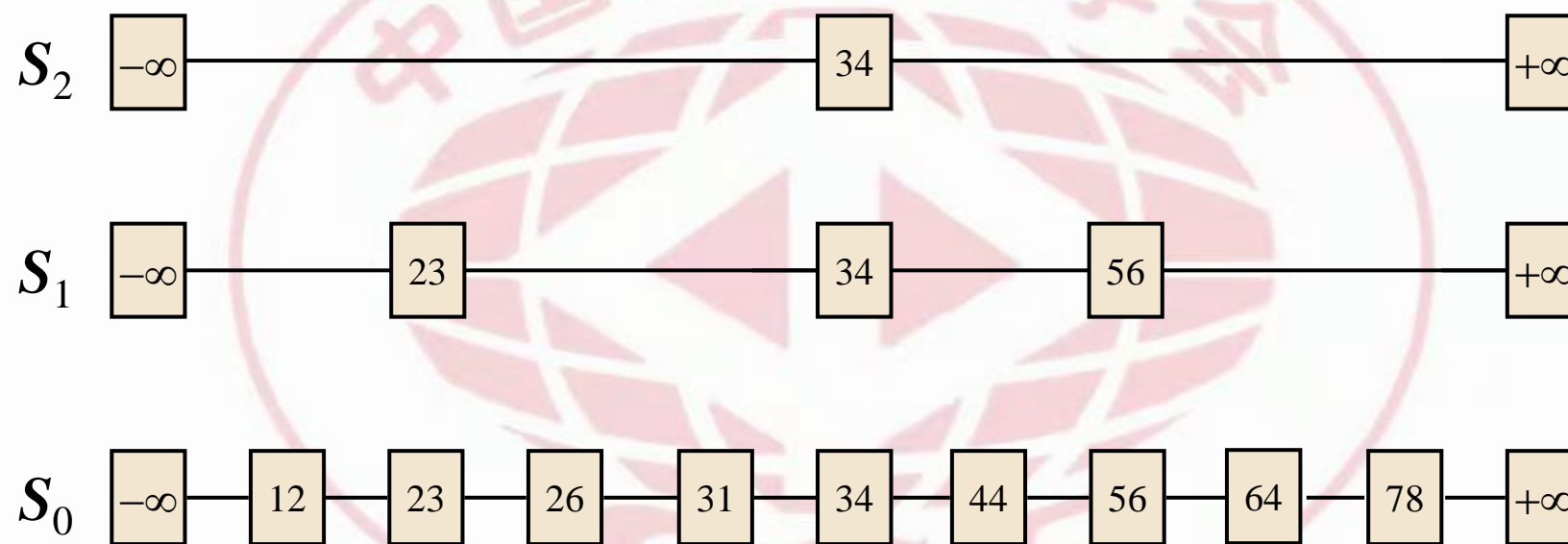
❖ 删除



❖ 恢复

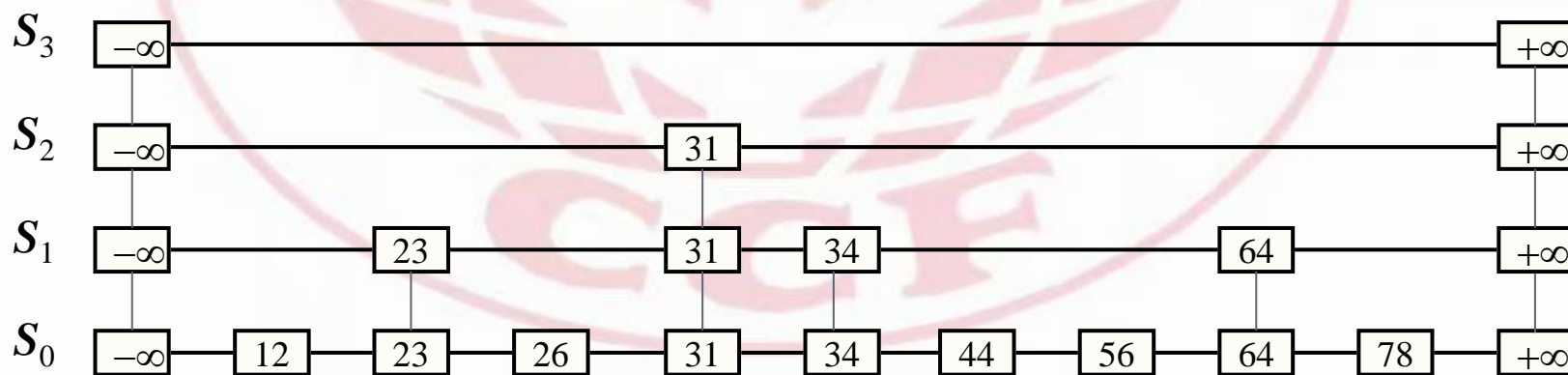
$\text{right}[\text{left}[p]] = p; \quad \text{left}[\text{right}[p]] = p;$

– 如何在链表上实现快速查找？



跳跃表

1. 跳跃表由多层链构成 ($S_0, S_1, S_2, \dots, S_h$)
2. 每条链必须包含两个特殊元素: $+\infty$ 和 $-\infty$
3. S_0 包含所有的元素, 并且所有链中的元素按照升序排列。
4. 如果一个元素出现在 第 i 层的链表中, 则它在 i 之下的链表也都会出现。
5. 每个节点包含两个指针, 分别指向下一个元素和下一层元素。



在跳跃表中查找 x 的操作如下：

我们从表的第一个位置的顶端开始

假设当前位置为 p ，我们比较 x 和 p 后面一个位置的值 $y \leftarrow key(next(p))$

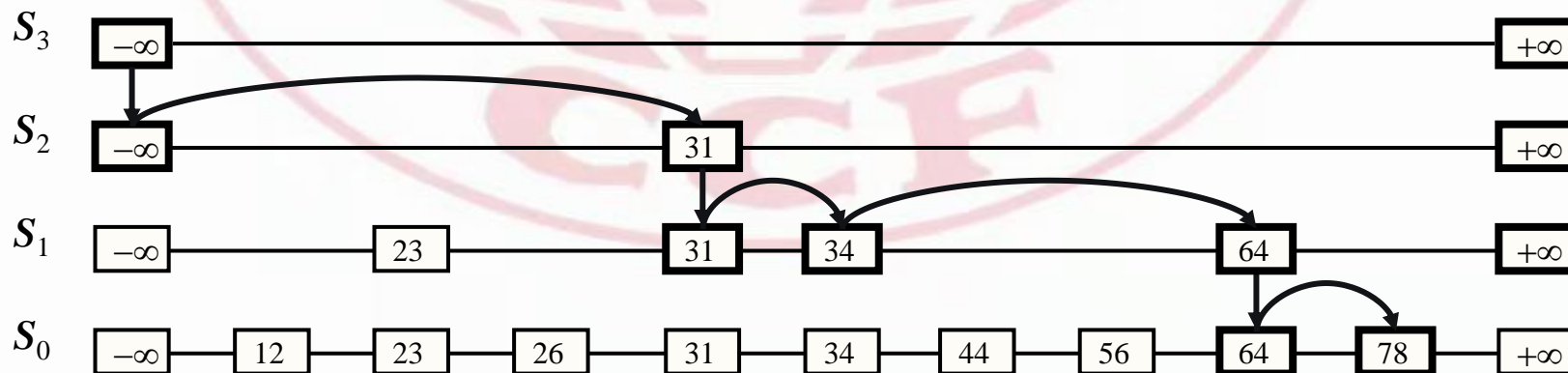
$x = y$: 返回 p 后面一个位置 ($next(p)$)

$x > y$: 继续向前搜

$x < y$: 跳到下一层

如果已经到了最底层还要往下跳，返回 x 不存在

例如：查找 78





插入操作

在跳跃表中插入一个元素 x 的操作：

- 1、查找插入的位置和插入对应元素。
- 2、先确定该 x 的层数 K （用丢硬币方式，完全随机）
- 3、然后在 $S_0, S_1, S_2 \dots, S_k$ 各个层的链表都插入 x 元素。



删除操作

在跳跃表中删除一个元素 x 的操作：

- 1、查找 x 元素的位置，如果未找到，则退出。
- 2、将该元素所在整列从表中删除
- 3、将多余的“空链”删除



跳跃表的时空效率

- 空间复杂度: $O(n)$ (期望)
- 跳跃表高度: $O(\log n)$ (期望)
- ✓ 相关操作的时间复杂度:
 - 查找: $O(\log n)$ (期望)
 - 插入: $O(\log n)$ (期望)
 - 删除: $O(\log n)$ (期望)



跳跃表总结

- 1、跳跃表是用于解决查找问题的数据结构
- 2、原理简单易于理解，代码实现容易
- 3、效率与平衡树相当



例：坐座位

题目大意：数轴上有 N 个座位依次分布在1至 N 个整数点上。有 N 个人排好队，要坐到1到 N 个座位上，第 N 个人在数轴的整点0处，第 $N-1$ 个人在数轴的整点 -1 处，……第1个人在数轴的整点 $-N+1$ 处。第 i 个人要去的位置是 S_i （每个人都有唯一的一个要去的位置）。如果没有被挡住，人每秒会向右移动一步到达下一个整点。当第 i 个人到达它的座位 S_i 时，它需要花费 T_i 秒时间把行李放到头顶的行李架上，在此 T_i 秒中，在他左边的所有的被挡住的人都不能移动，要等第 i 个人放好行李坐好后才能动。问至少要多少秒之后，所有的人都能坐到自己的座位上？（ $T_i \leq 10^9$ ， $N \leq 2 \times 10^5$ ）

输入：单个整数 N ，后面有 N 行，没行有两个整数 S_i 和 T_i 。

输出：所有人坐好的时间。

| 样例输入 | 样例输出 | 说明 |
|-------------------------|------|--|
| 3 2 5 3 10 1 5 | 19 | 说明：第1个人要到第2个座位，第2个人要去第3个座位，第3个人要去第1个座位，第一秒他们都会移动一步，第3个人到达他的座位，坐下要5秒，再过3秒，1和2都到达自己的位置，1坐下5秒，2坐下10秒，共10秒，时间为：1+5+3+10=19 |



例：0-1矩阵

给定一个由0和1组成的N行M列的矩阵，是否能找到一个行的集合，使得集合中每一列都恰好包含一个1？

例如：如下的矩阵

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

包含（第1、4、5行）的集合是其中的一个解



算法实现：

如果矩阵 A 是空的，问题解决；成功终止。

否则，选择一列 c （确定的）。

选择一行 r ，满足 $A[r, c]=1$ （不确定的）。

把 r 包含进部分解。

对于所有满足 $A[r, j]=1$ 的 j ，

从矩阵 A 中删除第 j 列；

对于所有满足 $A[i, j]=1$ 的 i ，

从矩阵 A 中删除第 i 行。

在不断减少的矩阵 A 上递归地重复上述算法。



例：区间第k大数总和

给出 n 和 k ，求 $1 \sim n$ 排列中每个区间的第 k 大数的总和是多少，区间长度小于 k 时贡献为 0。

【输入文件】

第一行两个整数 N 和 K ($1 \leq N, K \leq 10^5$)。

第二行是 N 个用空格隔开的 $1 \sim N$ 的整数。

【输出文件】

输出为一个整数，表示总和。

【输入输出样例】

| | |
|-----------|----------|
| sum. in | sum. out |
| 5 2 | 30 |
| 1 2 3 4 5 | |



例：暴力排序

给出一个序列。如果不满足 $a[i-1] \leq a[i] \leq a[i+1]$ ，则称数 $a[i]$ 是无序的。现在每一次把序列中所有无序的数删去，剩下的数合成新的序列，直到无法删除为止。输出最后剩下的序列。

【输入文件】

第一行一个整数 T N 。

后面有 T 行，为 T 组数据，每行 N 个用空格隔开的整数。

【输出文件】

输出 T 行，每行第一个数表示剩下数的个数，后面是剩下的数。

【输入输出样例】

| sort.in | sort.out |
|-----------|-------------|
| 5 5 | |
| 1 2 3 4 5 | 5 1 2 3 4 5 |
| 5 4 3 2 1 | 0 |
| 1 2 3 2 1 | 2 1 2 |
| 1 3 5 4 2 | 2 1 3 |
| 2 4 1 3 5 | 3 2 3 5 |



例：

题目大意：有一串序列，你要使得最终的序列正负交替，有两个操作：

1. 用c的代价使任意部分元素+1或-1。
2. 用r代价删除一个数。

问你最后用的代价最少是多少。

【输入文件】

第一行为三整数 N, C, R 。

第二行为N个用空格隔开的整数。

【输出文件】

输出代价最少值。

【输入输出样例】

```
sort.in
4 10 50
8 8 2 -2
```

```
sort.out
80
```



例：中位数

给你n个数，每次插入一个数，当插入数的数量为奇数的时候，我们就输出中位数。

【输入文件】

第一行一个整数 N ($1 \leq N, B \leq 105$)。

第二行是 N 个用空格隔开的整数。

【输出文件】

输出为一个整数，表示总和。

【输入输出样例】

median.in

9

1 2 3 4 5 6 7 8 9

9

9 8 7 6 5 4 3 2 1

23

23 41 13 22 -3 24 -31 -11 -8 -7

3 5 103 211 -311 -45 -67 -73 -81

-99 -33 24 56

median.out

1 2 3 4 5

9 8 7 6 5

23 23 22 22 13 3 5 5 3 -3 -7 -3



例：二叉查找树（BST）

题目大意：给你1-N个整数组成的序列。按给定序列的顺序建立一个二叉查找树，把第一个整数作为根，然后依次插入后面的整数。

每个结点X的插入过程为insert(X, root):

```
insert( number X, node N )
{
    C++;
    if X小于结点N的值
        if N没有左孩子
            新建一个节点，把X作为N左孩子
        else
            insert(X, N左子树)
    else X大于结点N的值
        if N没有右孩子
            新建一个节点，把X作为N右孩子
        else
            insert(X, N右子树)
}
```

你的任务是：每次把序列的一个整数插入到二叉查找数后，到目前为止计数累加器C的值是多少？

输入：一个整数N, 表示序列有多少个整数。接下来有N个1到N的整数序列。

输出：N个数，表示每插入一个数时计数器C的值是多少。

数据范围：1 ≤ N ≤ 300000



【输入输出样例】

| | | |
|--|--|---|
| <p>样例1:</p> <p>4</p> <p>1</p> <p>2</p> <p>3</p> <p>4</p> | <p>样例2</p> <p>5</p> <p>3</p> <p>2</p> <p>4</p> <p>1</p> <p>5</p> | <p>样例3</p> <p>8</p> <p>3</p> <p>5</p> <p>1</p> <p>6</p> <p>8</p> <p>7</p> <p>2</p> <p>4</p> |
| <p>输出</p> <p>0</p> <p>1</p> <p>3</p> <p>6</p> <p>输出说明：（第一次插入根1，不执行过程<code>insert(X,N)</code>，所以C是0，第二次插入2， C加1，所以插入2完毕后，输出C的值是1，第三次插入3，依次执行了<code>insert(3, 1)</code>、<code>insert(3,2)</code>，所以C 加2，变成3； 第四次插入4，依次执行了<code>insert(4, 1)</code>、<code>insert(4,2)</code>、<code>insert(4,3)</code>， 所以C 加3，变成6。</p> | <p>输出</p> <p>0</p> <p>1</p> <p>2</p> <p>4</p> <p>6</p> | <p>输出</p> <p>0</p> <p>1</p> <p>2</p> <p>4</p> <p>7</p> <p>11</p> <p>13</p> <p>15</p> |



例：靴子

奔奔有M双靴子，每双靴子都有一个高度和重量，奔奔穿上不同重量靴子每步走的距离不同，第i双靴子的高度和每步走的距离分别为 s_i 和 d_i 。

奔奔从家到学校的距离为N（可以看成是N个格子），刚下了雨，已知每个格子水的深度，问穿第i双靴子是否可以从家走到学校，如果格子上水的深度大于靴子的高度，则穿这双靴子不能踩在这个格子上。

【输入文件】

第一行两个整数 N和B ($1 \leq N, B \leq 105$)。

第二行是 N个用空格隔开的整数，表示第i个格子水的深度 f_i ($0 \leq f_i \leq 109$)。

后面有B行，表示每双靴子的高度 s_i 和走的距离 d_i 。

【输出文件】

输出为B个数，第i行表示，奔奔穿第i双靴子是否能走到学校，1表示可以，0表示不可以该序列的最大前缀长度，使得该前缀的所有盘子洗干净后，能按小号在下，大号在上的规则堆叠。

【输入输出样例】

| boot.in | boot.out |
|-----------------|----------|
| 8 7 | 0 |
| 0 3 8 5 6 9 0 0 | 1 |
| 0 5 | 1 |
| 0 6 | 0 |
| 6 2 | 1 |
| 8 1 | 1 |
| 10 1 | 1 |
| 5 3 | |
| 150 7 | |



例：猜数游戏

小博想一个数 $x+0.5$ ，其中 x 是某个 0 到 N 之间的整数，小乔猜。小博每次会回答猜高了或猜低了。小乔创建了一个包含 1 到 N 的 N 个整数序列，按序列中的数的顺序依次猜数。

小乔会跳过所有不必要的猜测：如果小乔将要猜某个数 i ，而之前已经猜过了某个 $j < i$ 并且小博回答“高”，他不会再猜 i ，而是继续猜序列中的下一个数。同理，如果他将要猜某个数 i ，而他之前已经猜过了某个 $j > i$ 并且小博回答“低”，小乔不会再猜 i ，而是继续猜序列中的下一个数。

如果我们将所有小博回答的“高”或“低”拼接成一个字符串 S ，求字符串 S 中出现“高低”的次数。

小博知道小乔将要使用这一策略；此外，他还知道小乔将要使用的排列。然而，小博尚未决定选用哪个值 x 。帮助小博对于每个值 x 求出他会说“高低”的次数。

【输入格式】 输入的第一行包含 N 。第二行包含小乔的长为 N 的排列。

【输出格式】 对于从 0 到 N 的每一个 x ，输出一行，包含小博会说“高低”的次数。

【输入输出样例】

| 输入 | 输出 | 样例解析： |
|-----------|----|-------------------------------|
| 5 | 0 | 对于 $x=0$ ，会说“高高”，总计零次“高低”。 |
| 5 1 2 4 3 | 1 | 对于 $x=2$ ，会说“高低低高高”，总计一次“高低”。 |
| | 1 | 对于 $x=3$ ，会说“高低低高低”，总计两次“高低”。 |
| | 2 | |
| | 1 | |
| | 0 | |



例：克隆

题目大意：开始有1个克隆人，现在有5种操作：

learn ci pi， 让ci克隆人学会技能pi;

rollback ci， 删除ci克隆人学会的最近一个技能并记下删除记录;

relearn ci， 让ci克隆人重新学习之前删除的技能;

clone ci， 再克隆一个ci;

check ci， 输出ci克隆人当前最近学的技能。

输入N个操作，输出询问结果

【输入输出样例】

| 输入 | 输出 |
|------------|-------|
| 9 10 | 5 |
| learn 1 5 | 7 |
| learn 1 7 | basic |
| rollback 1 | |
| check 1 | |
| clone 1 | |
| relearn 2 | |
| check 2 | |
| rollback 1 | |
| check 1 | |



例：种树

A城市有一个巨大的圆形广场，市政府决定沿圆形广场外圈种一圈树。有 n 个种树的位置，顺时针编号1到 n 。并且每个位置都有一个美观度 A_i ，如果在这里种树就可以得到这 A_i 的美观度。但两棵树决不能种在相邻的位置（ i 号位置和 $i+1$ 号位置叫相邻位置，1号和 n 号也算相邻位置）。最终市政府提供了 m 棵树苗并要求全部种上，请你帮忙设计种树方案使得美观度总和最大。如果无法将 m 棵树苗全部种上，给出无解信息。（ $m \leq n \leq 2 \times 10^5$ ， $1000 \leq A_i \leq 10000$ ）。

【输入格式】

输入的第一行包含两个正整数 n ， m 。

第二行 n 个整数，第 i 个代表 A_i 。

【输出格式】

输出一个整数，表示最佳植树方案可以得到的美观度。如果无解输出 Error!。

【输入输出样例】

| 输入 | 输出 |
|----------------------|----|
| 7 3 1 2 3 4 5 6 7 | 15 |



中
Chi

Description

$n \times n$ 的字符的矩阵， Q 次选择子正方形逆时针旋转90度

第 i 行第 j 列的位置称为 (i, j)

求最后的字符矩阵

Input

第一行 n, Q

接下来 n 行表示初始矩阵

接下来 Q 行

每行 i_k, j_k 表示子正方形的左上角，边长为 s_k

$n \leq 1000$

$Q \leq 2000$

$1 \leq i_k \leq n - s_k + 1$

$1 \leq j_k \leq n - s_k + 1$

$2 \leq s_k \leq n$

Output

n 行，表示最后的矩阵



Sample Input

```
4 1
abcd
efgh
ijkl
mnop
2 2 2
```

Sample Output

```
abcd
egkh
ifjl
mnop
```


例：温室生长

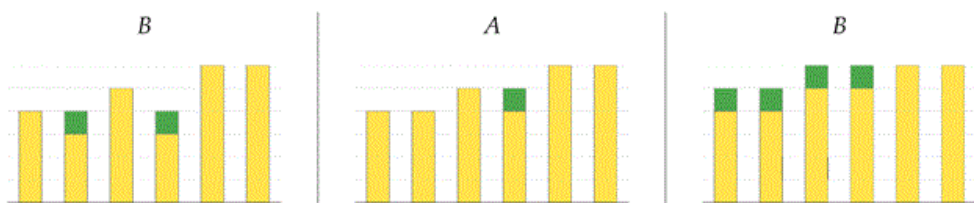
在温室中有 n 个直线排列的向日葵植株，从左向右编号为1到 n 。在向日葵的左右两端有2个照射器为向日葵的生长提供光和热，分别为A和B。每天只有1个照射器被打开，使所有向日葵转向光源，并使部分向日葵生长。向日葵会生长当且仅当其朝向的相邻植株比它更高，其每天的生长高度为1厘米。请注意，一个植株的生长将使其背后的植株立刻开始生长。给出向日葵的初始高度和接下来 m 天的光照计划，请计算所有向日葵最终的高度。

【输入文件】

第一行有2个整 n 和 m ，表示植株数和天数($1 \leq n, m \leq 300\,000$)。
接下来一行包括 n 个整数 h_1, h_2, \dots, h_n ，表示从左到右向日葵的初始高度 ($1 \leq h_k \leq 10^9$)。
接下来一行包括一个仅含字母A/B长度为 m 的字符串，表示从第一天开始的光照计划。

【输出文件】

输出 n 个整数，表示从左到右每株向日葵最终的高度。

| 样例输入 | 样例输出 | |
|-----------------------------|-------------|--|
| 6 5 4 3 5 3 6 6 BABAA | 5 5 6 6 6 6 |  |
| | | 样例数据前三天的生长情况 |