string

点此看题

由题意可知 N 得为奇数，S 才存在，所以先特判 N 为偶数的情况。

由题意可知 S 的长度为 $\lfloor \frac{n}{2} \rfloor$，令 $M = \lfloor \frac{n}{2} \rfloor$。

因为只插入一个字符，所以如果存在 S，则 U 的前 M 个字符或后 M 个字符中一定有一边是 S。

所以可以用 substr 函数分别截取前 M 个字符和后 M 个字符再依次匹配检查是否合法。

sort

点此看题

能否较快找到进行完整冒泡排序的轮数，以及剩下的交换次数?

剩下的交换次数必定很小，我们能否还原经过 k 次完整的冒泡排序之后的序列，然后暴力把剩下的交换次数用完?

我们显然可以通过二分找到完整的交换轮数 x，具体地，记 $d_i = \sum_{j=1}^{i-1}[a_j > a_i]$，那么，假设我们当前假定的轮数为 x，那么位置 i 所使用的的交换次数就是 $\min(x, d_i)$，用这个来检测 mid 的合法性即可。

对于如何还原出原来的序列，对于 $d_i \geq x$ 的部分，实际上它就会跑到 $i - x$ 的位置去，因为左边会有 x 个比他更大的数字越过他。这种数字最简单，我们先把他们安定好，然后用剩下的数字来填坑。显然，对于 $d_i < x$ 的数字，他们左边已经没有比他们更大的数字，因此，这些数字在剩下的坑里面肯定是单增的。从左往右，从小往大填就行了。

最后再用一轮冒泡排序就行了。时间复杂度 $\mathcal{O}(N \log K)$.

zxy的题解

jzm的题解

set

$Q \leq 600$：$O(Q^3)$大暴力；

$Q \leq 5000$：$O(Q^2 \log Q)$暴力，用multiset维护$A, B, \max(a_x + a_y, b_x + b_y)$；

正解：link

easy

点此看题

3300 的题啊，就差临门一脚了 $...$

直接做有点难，我们**观察操作结构设计图论模型**，因为这是相邻两个数配对的问题，那么如果两个数配对我们新建一个点表示它们配对后的数，然后把它们和新点连一条边，发现最后是一颗二叉树的结构。

定义某点的深度为从根到它向右走的次数，不难发现每个叶子的贡献是 $2^{dep_i} \cdot a_i$，问题是最大化贡献。

区间 dp 没有前途，可以先设计一个朴素 dp，设 $dp[i][x]$ 表示第 i 个点的深度是 x 剩下的最大数：

$dp[i][x] = dp[i-1][x-1] + 2^x \cdot a_i, x > 1$

$dp[i][1] = \max(dp[i-1][y]) + 2 \cdot a_i$

初始化 $dp[1][0] = a_1$，这个 dp 可以很显然的降维，设 $dp[i]$ 表示前 i 个点剩下的最大数：

$dp[i] = \sum_{j=1}^{i-1} dp[j] + cal(j+1, i)$

其中 $cal(l, r) = \sum_{i=1}^{r-l+1} a_{i+l-1} \cdot 2^i$，其实熟练的选手可以直接设计这个一维 dp，因为他的本质是枚举最后一段"链"，所以可以考虑所有情况。

但是本题有一个极其特殊的地方：`答案对1e9+7取模`，翻译一下就是不允许你使用取 $\max$ 之类的操作，那么自然就不能 dp 了。我们考虑 dp 转贪心，它的主要思路是**考虑转移点的性质然后直接取转移点。**

考虑已经得到了 $dp[i-1]$ 的转移点集合 s，考虑如果 a_i 是负数那么往 s 里面添加转移点 i，否则把 i 合并到前一个转移段是最优的，并且我们发现如果 i 的转移段权值为正，那么就可以直接往前合并。

简单的证明一下：不难发现这样操作是当前最优的，我们考虑这样操作有没有后效性，由于合并的条件是条件是当前转移段权值为正，那么无论当且的合并怎样都不会影响后面的合并的。

但是这道题还要我们解决区间问题（*[OID脏话]*），可以套路地离线，然后固定右端点，维护每一个左端点的答案，用并查集维护转移段，再维护转移段的权值前缀和。不难发现第一个转移段可能取不满，但是这并不会对合并造成影响，所以段还是以前的那些段，那么可以直接计算权值。

注意因为 dp 初始化的原因第一段的权值是要除以二的，因为第一个元素一开始直接作为根。

其实 dp 转贪心能算做优化 dp 的方法，考虑转移点的性质即可。

```
#include <cstdio>
#include <vector>
using namespace std;
const int MOD = 1e9+7;
const int M = 100005;
#define int long long
const int inf = 2e9;
int read()
{
  int x=0,flag=1;char c;
  while((c=getchar())<'0' || c>'9') if(c=='-') flag=-1;
  while(c>='0' && c<='9') x=(x<<3)+(x<<1)+(c^48),c=getchar();
  return x*flag;
}
int n,m,a[M],ans[M],pr[M],sf[M],fa[M],f[M],l[M],pw[M];
struct node
{
  int x,id;
};vector<node> q[M];
int find(int x)
{
  if(x!=fa[x]) fa[x]=find(fa[x]);
  return fa[x];
}
```

```
void merge(int x,int y)
{
  if(l[x]>31 || f[x]+(f[y]<<l[x])>inf) f[x]=inf;
  else f[x]+=f[y]<<l[x];
  l[x]+=l[y];
  fa[y]=x;
}
int cal(int l,int r)
{
  return (sf[l]-sf[r+1]*pw[r-l+1])%MOD;
}
signed main()
{
  n=read();m=read();pw[0]=1;
  for(int i=1;i<=n;i++)
    pw[i]=pw[i-1]*2%MOD;
  for(int i=1;i<=n;i++)
    f[i]=a[i]=read(),fa[i]=i,l[i]=1;
  for(int i=n;i>=1;i--)
    sf[i]=(sf[i+1]*2+a[i])%MOD;
  for(int i=1;i<=m;i++)
  {
    int l=read(),r=read();
    q[r].push_back(node{l,i});
  }
  for(int i=1;i<=n;i++)
  {
    int x=find(i);
    while(find(x-1) && f[x]>0)
      merge(find(x-1),x),x=find(i);
    pr[x]=(pr[find(x-1)]+f[x])%MOD;
    for(auto t:q[i])
    {
      ans[t.id]=(pr[x]-pr[find(t.x)])*2
      +cal(t.x,find(t.x)+l[find(t.x)]-1);
      ans[t.id]=(ans[t.id]%MOD+MOD)%MOD;
    }
  }
  for(int i=1;i<=m;i++)
    printf("%lld\n",ans[i]);
}
```