【题解】2020 牛客 NOIP 赛前集训营-提高组（第四场）

写在前面

之前给牛客的提高组验过几次题，作为出题人还是第一次，中间有一些小锅，包括 B 题的样例不满足 y<=n 等等（实际上原来的第一组数据就是样例，后面修改重判了），但是最终还算比较完美 的结束了。

由于个人比较喜欢思维题，因此四道题的实现难度都不是很大，除了 C 题稍微有点套路（是数学题套数据结构）。

简单提要：A 题比较简单，读懂题意基本就能 AC 了；B 题其实是双向链 表，但是没有卡平衡树做法，因此不想动脑筋直接敲平衡树也是能 AC 的；C 题是线段树维护矩阵 / 多项式等，由于不想卡常时限给了 std 的 10 倍，因此不管维护什么都是能 AC 的（除非常数真的巨大）；D 题是一道难度较高的思维题，会 KMP 就能得到 40%，而要拿 100%需要发现某种类似倍增的结构。

赛中有四位选手 AC 了 C 题，有一位选手成功 AK，并且是唯一一位 AC 了 D 题的选手。（感谢你们>_<）

以下是详细的题解和 std。

Problem A. 语言

题意

每个字母有 A,N,V 三种可能的词性，$NP:=N\mid A+N\mid NP1+NP2$，$S:=NP+V+NP$，判断某个字符串能不能成为$S$。

做法

其实就是看能不能组成 $X\ldots XNVX\ldots XN$ 的结构，其中X可以是N或A。

直接暴力，复杂度为$O(n^2)$。期望通过 30%。

预处理可以构成前一个$X\ldots XN$的位置和后一个$X\ldots XN$的位置，复杂度为$O(n)$。

期望通过 100%。

Std

```
#include <bits/stdc++.h>
using namespace std;
char s[100100];
int w[30], ok[100100];
int main(void) {
    int T; scanf("%d",&T);
    assert(T <= 10);
    while (T--) {
        memset(ok,0,sizeof(ok));
        for (int i=0;i<26;i++) scanf("%d",&w[i]);
        for (int i=0;i<26;i++) assert(1 <= w[i] && w[i] <= 7);
        scanf("%s",s);
        for (int i=0;s[i];i++) assert('a' <= s[i] && s[i] <= 'z');
        int n = strlen(s);
        for (int i=0;i<n;i++) {
            if (w[s[i]-'a']&2) ok[i]=1;
            if (w[s[i]-'a']==4) break;
        }
        if (!(w[s[n-1]-'a']&2)) {
            printf("No\n");
            continue;
        }
        bool f=0;
        for (int i=n-2;i>=1;i--) {
```

```
            if ((w[s[i]-'a']&4) && ok[i-1]) {
                f=1;
                break;
            }
            if (w[s[i]-'a']==4) break;
        }
        if (f) printf("Yes\n");
        else printf("No\n");
    }
    return 0;
}
```

Problem B. 色球

题意

n个位置，m个操作。操作 1，把x个颜色为y的球放到z位置的顶部；操作 2，从z位置顶部取出x个球，输出最后一个球的颜色；操作 3，把位置u的球倒过来放到位置v顶部。

以下假设n与m同阶。

做法一

每个位置一个 vector 顺序记录放的球的颜色和个数，所有询问暴力进行。

复杂度$O(n^2)$，期望通过 30%~50%。

做法二

每个位置用双向链表顺序记录放的球的个数和颜色。

对于询问 1，往位置 z 的链表上加一个结点(x, y)。

对于询问 2，从位置 z 的链表头取，如果当前需要取$\boldsymbol{x}$个而当前结点拥有的球$x' < x$个，则整个结点删掉，x更新为$x - x'$。如果$x' \geq x$，那么输出当前的颜色y'，并且把 x'更新为$x' - x$。

对于询问 3，把位置u的链表头直接拼接到位置v的链表头上，位置u的链表尾作为位置v新的链表头。

考虑这个做法的复杂度，询问 1 和 3 很显然是$O(1)$的，询问 2 中每次碰到 $x' < x$的情况就会删掉一个结点，而每次询问最多碰到一次$x' \geq x$的情况。因为结点最多增加m次，因此整体的复杂度是$O(n)$。期望通过 100%。

Std

```
#include <bits/stdc++.h>
using namespace std;

const int maxn = 301001;
struct node {
    int c,num,ch[2];
} buf[maxn];
int n,m,h[maxn],t[maxn],tot;
int main(void) {

    scanf("%d%d",&n,&m);
    for (int i=0;i<m;i++) {
        char op[6];
        int x,y,z;
        scanf("%s%d%d",op,&x,&y);
        if (op[2]=='s') {
            scanf("%d",&z);
            buf[++tot]=(node){y,x,h[z],0};
            if (h[z])
buf[h[z]].ch[0]?(buf[h[z]].ch[1]=tot):(buf[h[z]].ch[0]=tot);
            else t[z]=tot;
            h[z]=tot;
        } else if (op[2]=='p') {
            while (buf[h[y]].num<x) {
                x-=buf[h[y]].num;
                int lch=buf[h[y]].ch[0]|buf[h[y]].ch[1];
                if (lch)
buf[lch].ch[0]==h[y]?(buf[lch].ch[0]=0):(buf[lch].ch[1]=0);
                h[y]=lch;
            }
            buf[h[y]].num-=x;
            printf("%d\n",buf[h[y]].c);
        } else {
            if (!h[x]) continue;
            if (h[y]) {
```

```
buf[h[y]].ch[0]?(buf[h[y]].ch[1]=h[x]):(buf[h[y]].ch[0]=h[x]);

buf[h[x]].ch[0]?(buf[h[x]].ch[1]=h[y]):(buf[h[x]].ch[0]=h[y]);
            } else {
                t[y]=h[x];
            }
            h[y]=t[x];
            h[x]=t[x]=0;
        }
    }
    return 0;
}
```

Problem C. 斐波

题意

给定序列$a_1, a_2, \ldots, a_n$

操作 1，把a_p修改为v;

操作 2，计算$\sum_{i=l}^{r} \sum_{j=i}^{r} f(\{a_i, a_{i+1}, .., a_j\})$

其中$f(S) = \sum_{T\subseteq S}[fib(\sum_{s\in T} s)]^2$。

做法一

考预处理fib的数列，修改操作直接修改，询问时按公式枚举答案相加。

复杂度 $O(qn^2 2^n)$或 $O(qn2^n)$。预计都是通过 10%。

做法二

首先观察到其实$g(n) = fib^2(n)$也是符合线性递推的，递推式为$g(n) = 2g(n-1) + 2g(n-2) - g(n-3)$

所以可以写出矩阵形式的递推式

$$\begin{pmatrix} g(n) \\ g(n-1) \\ g(n-2) \end{pmatrix} = \begin{pmatrix} 2 & 2 & -1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} g(n-1) \\ g(n-2) \\ g(n-3) \end{pmatrix}$$

即$\underset{gn}{\rightarrow} = A_g \underset{n-1}{\longrightarrow} = A^n \underset{g0}{\rightarrow}$。

这里$\overrightarrow{g0}$=$(g(0),g(-1),g(-2))^T$ 可以根据递推式反推出来，用这种形式比较方便之后的推导。

对于可重集合 S，$f(S)=\sum_{T\subseteq S}[fib(\sum_{s\in T}s)]^2$，可以把 f 也向量化，变成$\overrightarrow{f(S)}$=$\sum_{T\subseteq S}g\overrightarrow{\sum(T)}$ 最终答案就是 f 的第一项。

如果给 S 增加一个数 a，可以得到$\overrightarrow{f(S\cup\{a\})}$=$\overrightarrow{f(S)}$ $+\sum_{T\subseteq S}g\overrightarrow{\sum(T)+a}$

=$\overrightarrow{f(S)}$+$\sum_{T\subseteq S}A^{a}\,g\overrightarrow{\sum(T)}$

=$(I+A^{a})\overrightarrow{f(S)}$

因此假设 S=$\{a_1,a_2,\dots,a_n\}$，则$\overrightarrow{f(S)}$=$\prod_{i=1}^{n}\left(I+A^{a_i}\right)\overrightarrow{g0}$

令$B_i=I+A^{a_i}$，其实问题就是单点更新B_i，询问$\sum_{i=l}^{r}\sum_{j=i}^{r}\prod_{k=i}^{j}B_k$ 。

使用这个式子进行暴力，复杂度为 $O(3^3qn^2)$。期望通过 50%

做法三

线段树可以维护$\sum_{i=l}^{r}\sum_{j=i}^{r}\prod_{k=i}^{j}B_k$ 。

具体就是考虑合并两个区间 $[l,m]$，$[m+1,r]$时，增加的答案是$[i,m]$和$[m+1,j]$ $(l\le i\le m,m+1\le j\le r)$这些区间合并产生的。使劲地维护这些信息就可以了。

不使用线段树，但使用类似的思想，可以达到$O(3^3qn)$。期望通过 70%。

使用线段树，复杂度为$O(3^3q\log n)$。期望通过 100%。

生成函数的做法

首先要知道一个结论：

假设线性递推$g(n)=a_1g(n-1)+\dots+a_kg(n-k)$

令 $f(x)=x^k-a_1x^{k-1}-\dots-a_k$

令 $r(x)=x^n \bmod f(x)$

然后有 $g(n)=\sum_{i=0}^{k-1}g(i)*[x^i]r(x)$，其中$[x^i]$r(x)是 r(x)的$x^i$项的次数。

所以我们可以预处理每个$x^i \ mod\ f(x)$, (0≤i≤100000)。

线段树每个位置维护 $(x^{a_i})\ mod\ f(x)$，后面的做法基本是一样的。这样的好处

是常数会从3^3降到3^2。复杂度为$O(3^3 qlogn)$。期望通过 100%。

Std

```
#include <bits/stdc++.h>
#define ui unsigned int
#define ll long long
#define ull unsigned ll
using namespace std;
const int mod = 998244353;
const int maxn = 100100;
struct node {
    int x,y,z;
    node operator + (const node & a) const {
        return (node){(x+a.x)%mod,(y+a.y)%mod,(z+a.z)%mod};
    }
    node operator * (const node & a) const {
        int A=x*(ll)a.x%mod;
        int B=(x*(ll)a.y%mod+y*(ll)a.x%mod)%mod;
        int C=(x*(ll)a.z%mod+y*(ll)a.y%mod+z*(ll)a.x%mod)%mod;
        int D=(y*(ll)a.z%mod+z*(ll)a.y%mod)%mod;
        int E=z*(ll)a.z%mod;
        return (node){(int)(((A-D+mod)%mod-2ll*E%mod+mod)%mod),
            (int)((B+2ll*D%mod+3ll*E%mod)%mod),
            (int)((C+2ll*D%mod+6ll*E%mod)%mod)};
    }
} f[maxn], sum[maxn*4], msum[maxn*4], lsum[maxn*4], rsum[maxn*4];

void pushup(int rt) {
    int l=rt<<1, r=rt<<1|1;
    msum[rt]=msum[l]*msum[r];
    sum[rt]=sum[l]+sum[r]+rsum[l]*lsum[r];
    lsum[rt]=lsum[l]+msum[l]*lsum[r];
    rsum[rt]=rsum[r]+msum[r]*rsum[l];
}
void build(int l, int r, int rt) {
    if (l == r) {
        int u; scanf("%d",&u);
        assert(1 <= u && u <= 100000);
```

```
        sum[rt]=lsum[rt]=rsum[rt]=msum[rt]=f[u]+(node){1,0,0};
        return ;
    }
    int mid=(l+r)/2;
    build(l,mid,rt<<1);
    build(mid+1,r,rt<<1|1);
    pushup(rt);
}
void upd(int p, int u, int l, int r, int rt) {
    if (l == r) {
        sum[rt]=lsum[rt]=rsum[rt]=msum[rt]=f[u]+(node){1,0,0};
        return ;
    }
    int mid=(l+r)/2;
    if (p <= mid) upd(p,u,l,mid,rt<<1);
    else upd(p,u,mid+1,r,rt<<1|1);
    pushup(rt);
}
node ms,s,ls,rs;
void ask(int L, int R, int l, int r, int rt) {
    if (L <= l && r <= R) {
        if (l == L) {
            ms = msum[rt];
            s = sum[rt];
            ls = lsum[rt];
            rs = rsum[rt];
        } else {
            s = s + sum[rt] + rs*lsum[rt];
            ls = ls + ms*lsum[rt];
            rs = rsum[rt] + msum[rt]*rs;
            ms = ms * msum[rt];
        }
        return ;
    }
    int mid=(l+r)/2;
    if (L <= mid) ask(L,R,l,mid,rt<<1);
    if (mid < R) ask(L,R,mid+1,r,rt<<1|1);
}
int main(void) {
    int N = 100001;
    f[0]=(node){1,0,0};
    for (int i=1;i<=N;i++) f[i]=f[i-1]*(node){0,1,0};

    int n,q; scanf("%d%d",&n,&q);
```

```
    assert(1 <= n && n <= 100000);
    assert(1 <= q && q <= 100000);
    build(1,n,1);
    for (int i=0;i<q;i++) {
        int op,x,y;
        scanf("%d%d%d",&op,&x,&y);
        assert((op==1 && 1<=x&&x<=n && 1<=y&&y<=100000)
            || (op==2 && 1<=x&&x<=y&&y<=n));
        if (op == 1) {
            upd(x,y,1,n,1);
        } else {
            ask(x,y,1,n,1);
            int ans = (s.y+s.z)%mod;
            printf("%d\n",ans);
        }

    }
    return 0;
}
```

Problem D. 偶数

题意

定义“偶数”为数字的位数为偶数，且数字前一半和后一半完全一致的数。任意的“偶数”可以添加（至少一个）最少的数字使之成为新的“偶数”。一直按照这种方式添加，直到位数超过 n，然后多次询问最终“偶数”的$[l,r]$位模 998244353 是多少。

做法一

枚举下一个“偶数”的位数，必然是把u的某个后缀放到u后。因此确定位数后，可以$O(n)$的判断能不能成为新的偶数。得到最终的“偶数”后，暴力的处理每个查询。

复杂度 $O(n^2+qn)$。预计通过 10%。

做法二

假设u是vv，那么新的“偶数”必然是$vwvw$的形式，其中w是v的一个前缀，且是v的一个周期。可以证明w应该取v最短的周期。因此使用 KMP 求v的周期，每次得到w后，新的vw作为v继续求周期，直到v的长度超过n为止。

询问时预处理$s[i]$为区间$[1,i]$的答案，则$[l,r]$的答案为 $s[r]-s[l-1]\times 10^{\mathrm{r-l+1}}$。

复杂度$O(n+q)$。预计通过 40%。。

做法三

在做法二的基础上。如果 $len(w)$是 $len(v)$的一个因子，那么最终的 v将会是 $vww..w$。

比较难的是$len(w)$不是$len(v)$的因子的情况，这时候可以证明vw的最短的周期只能是v，因此新的v是vwv。（证明见最下）

我们令$v1=v,v2=vw,vi=v_{i-1}v_{i-2}(i>2)$。最终的“偶数”将是$v_\infty$的一个前缀。注意到这个性质不管$len(w)$是否是$len(v)$的因子都是成立的，且$vi$的长度一定不小于两倍的$v_{i-2}$，因此实际上只需要求$logn$次即可达到要求。

实现时维护$logn$个vi的长度和值（模 mod），求区间$[1,m]$时每次取最长的不超过m的vi填进去，维护好对应的长度和值（模 mod）即可。

复杂度$O(len(u)+qlogn)$ 。预计通过 100%。

证明：$\boldsymbol{w}$是 v 的最短周期，且$\boldsymbol{len(w)}$不是$\boldsymbol{len(v)}$的因子，则$\boldsymbol{vw}$的最短的周期是$\boldsymbol{v}$。

假设vw的最短的周期是x，$len(x)<len(v)$。

如果$len(x)\equiv len(v) mod\ len(w)$，$gcd(len(x),len(w))$也是$v$的周期，矛盾。

如果 $len(x) \not\equiv len(v) mod\ len(w)$，$gcd(len(w),(len(v)-len(x))\%len(w))$是

w的周期，从而是v的周期，矛盾。

Std

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
const int mod = 998244353;
const int maxn = 1000100;
char s[maxn];
int n, lim, nxt[maxn], fs[maxn], f[100], ten[100];
ll len[100];
void get_nxt() {
    int i=0, j=nxt[0]=-1;
    while (i<n) {
        while (j!=-1 && s[i]!=s[j]) j = nxt[j];
        nxt[++i] = ++j;
    }
}
int qpow(int a, ll n) {
    int ans = 1;
    for (;n;n>>=1,a=(ll)a*a%mod) if (n&1) ans=(ll)ans*a%mod;
    return ans;
}
int gao(ll m) {
    ll sum = 0;
    int ans = 0;
    for (int i=lim;i>=0;i--) {
        if (sum + len[i] <= m) {
            ans = (ans * (ll)ten[i] + f[i])%mod;
            sum += len[i];
        }
    }
    ans = (ans * (ll)qpow(10, m-sum) + fs[m-sum])%mod;
    return ans;
}
int main(void) {
    int T; scanf("%d",&T);
    assert(T <= 10);
    int sum_q = 0;
    while (T--) {
        scanf("%s",s);
```

```
        n = strlen(s);
        assert(n % 2 == 0 && 1 <= n && n <= 100000);

        n /= 2;
        for (int i=0;i<n;i++) assert('1' <= s[i] && s[i] <= '9' && s[i]
== s[i+n]);

        get_nxt();
        for (int i=1;i<=n;i++) {
            fs[i] = (fs[i-1]*10ll + s[i-1]-'0')%mod;
        }

        ll m; int q; scanf("%lld%d",&m,&q), sum_q += q;
        assert(1 <= m && m <= (ll)1e18);
        assert(1 <= q && q <= 100000);

        len[0] = n - nxt[n], f[0] = fs[len[0]];
        len[1] = n, f[1] = fs[n];
        lim = 1;
        for (int i=2;i<100;i++) {
            len[i] = len[i-1] + len[i-2];
            f[i] = (f[i-1]*(ll)qpow(10, len[i-2]) + f[i-2])%mod;
            if (len[i] >= m) { lim=i; break; }
        }
        for (int i=0;i<=lim;i++) ten[i] = qpow(10, len[i]);


        for (int t=0;t<q;t++) {
            ll l, r;
            scanf("%lld%lld",&l,&r);
            assert(1 <= l && l <= r && r <= m);

            int ans = (gao(r) - gao(l-1)*(ll)qpow(10, r-l+1)%mod + mod) %
mod;
            printf("%d\n", ans);
        }
    }
    assert(sum_q <= 100000);

    return 0;
}
```