

[2019 Multi-University Training Contest 1]

[比赛网址](#)

训练结果

- rank: 算了这个别说了
- 完成情况 : 3/6/13

题解

B Operation

补题

题意

给了一个序列，要求实现两种操作

1. 给定 \$l,r\$ 求 \$a[l..r]\$ 种选出其中的一些值的最大异或和
2. 在序列的后面加一个 \$x\$

题解

开始想到线段树套线性基，发现时间和空间都爆了。后发现我们可以记录 \$a[1..dots i]\$ 的线性基，添加时候则从高位到低位，尽量用当前的基去替换之前的基，这样能使所有的基离 \$r\$ 更近。查询的时候只用位置大于 \$l\$ 的基。

```
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cstring>
#define ll long long
using namespace std;
int read()
{
    int k=0,f=1;char c=getchar();
    for(;!isdigit(c);c=getchar()) if(c=='-') f=-1;
    for(;isdigit(c);c=getchar()) k=k*10+c-'0';return k*f;
}
const int N=1000055;
int T,n,m,a[N],pos[N][35],base[N][35],Base[35],Pos[35];
void build(int p,int x)
```

```
{  
    for(int i=29;i>=0;i--)  
    {  
        if(x&(1<<i))  
        {  
            if(!Base[i])  
            {  
                Base[i]=x,Pos[i]=p;  
                return;  
            }  
            else  
            {  
                if(p>Pos[i])  
                    swap(p,Pos[i]),swap(x,Base[i]);  
            }  
            x^=Base[i];  
        }  
    }  
}  
int main()  
{  
    for(T=read();T;T--)  
    {  
        memset(Base,0,sizeof(Base));  
        memset(Pos,0,sizeof(Pos));  
        n=read();m=read();  
        for(int i=1;i<=n;i++)  
            a[i]=read();  
        for(int i=1;i<=n;i++)  
        {  
            build(i,a[i]);  
            for(int j=0;j<=29;j++)  
                base[i][j]=Base[j],pos[i][j]=Pos[j];  
        }  
        int lans=0;  
        for(int i=1;i<=m;i++)  
        {  
            int l,r,opt;  
            opt=read();  
            if(!opt)  
            {  
                l=(read()^lans)%n+1;  
                r=(read()^lans)%n+1;  
                if(l>r) swap(l,r);  
                int ans=0;  
                for(int j=29;j>=0;j--)  
                {  
                    if(pos[r][j]>=l)  
                    {  
                        if(j>0)  
                            pos[r][j]=pos[r][j-1];  
                        else  
                            pos[r][j]=l;  
                    }  
                }  
            }  
        }  
    }  
}
```

```

        if((ans^base[r][j])>ans)
            ans=ans^base[r][j];
    }
}
printf("%d\n",lans=ans);
}
else
{
    l=read()^lans;
    n++;a[n]=l;
    build(n,a[n]);
    for(int j=0;j<=29;j++)
        base[n][j]=Base[j],pos[n][j]=Pos[j];
}
}
return 0;
}

```

D Vacation

solved by wxg

题意

有 \$n\$ 辆汽车在过绿灯，每辆车给了距离红绿灯的位置，长度和最大速度。不能超车。假设每辆车都按最优方案驾驶，问最后一辆车过红绿灯的时间

题解

可以想象，最后几辆车一定是贴在一起通过的，我们只需要枚举 \$x\$ 来计算最后 \$x\$ 辆车贴贴后通过时间，取一个最大值即可。

```

#include<iostream>
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;
int read()
{
    int k=0,f=1;char c=getchar();
    for(;!isdigit(c);c=getchar()) if(c=='-') f=-1;
    for(;isdigit(c);c=getchar()) k=k*10+c-'0';return f*k;
}
const int N=100055;
int n,m,l[N],s[N],v[N];
int main()
{

```

```
while(scanf("%d",&n)!=EOF)
{
    for(int i=0;i<=n;i++)
        scanf("%d",&l[i]);
    for(int i=0;i<=n;i++)
        scanf("%d",&s[i]);
    for(int i=0;i<=n;i++)
        scanf("%d",&v[i]);
    double ans=1.*s[0]/v[0],sum=0;
    for(int i=1;i<=n;i++)
    {
        sum+=l[i];
        ans=max(ans,((sum+s[i])/v[i]));
    }
    printf("%.8f\n",ans);
}
return 0;
}
```

E Path

口胡 by fyh |written by wxg

题意

给了一个图，去掉一个边的代价为边权，问为使 \$1-n\$ 的最短路变长，最小代价是多少。

题解

求出最短路径的新图，可以看出答案为最小割

```
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cstring>
#include<queue>
#define ll long long
#define int long long
using namespace std;
int read()
{
    int k=0,f=1;char c=getchar();
    for(;!isdigit(c);c=getchar()) if(c=='-') f=-1;
    for(;isdigit(c);c=getchar()) k=k*10+c-'0';return k*f;
}
```

```
const int N=200055;
const ll inf=1ll<<60;
typedef pair<int,int> P;
int T,n,m,s,t;
int tot=1,iter[N],head[N],to[N],nextt[N],w[N],d[N],vis[N];
queue<int> q;
struct graph
{
    int tot=0,to[N],nextt[N],head[N],w[N];
    int u[N],v[N],c[N];
    ll dis[N];
    priority_queue <P,vector<P>,greater<P> > q;
    void add(int a,int b,int c)
    {
        to[++tot]=b;
        w[tot]=c;
        nextt[tot]=head[a];
        head[a]=tot;
    }
    void dij()
    {
        for(int i=2;i<=n;i++)
            dis[i]=1ll<<60;
        dis[1]=0;
        q.push(P(0,1));
        while(!q.empty())
        {
            P k=q.top();q.pop();
            int u=k.second;if(dis[u]<k.first) continue;
            for(int i=head[u];i;i=nextt[i])
                if(dis[to[i]]>dis[u]+w[i])
                {
                    dis[to[i]]=dis[u]+w[i];
                    q.push(P(dis[to[i]],to[i]));
                }
        }
    }
}G;
void add(int a,int b,int c)
{
    to[++tot]=b;
    w[tot]=c;
    nextt[tot]=head[a];
    head[a]=tot;
}
bool bfs()
{
    for(int i=1;i<=n;i++) d[i]=-1;
    d[s]=0;q.push(s);
    while(!q.empty())
    {
```

```
int u=q.front();q.pop();
for(int i=head[u];i;i=nextt[i])
    if(w[i]>0&&d[to[i]]==-1)
        d[to[i]]=d[u]+1,q.push(to[i]);
}
return d[t]!=-1;
}
int dfs(int u,int f)
{
    if(f==0||u==t) return f;
    for(int &i=iter[u];i;i=nextt[i])
    {
        if(d[to[i]]==d[u]+1&&w[i]>0)
        {
            int ff=dfs(to[i],min(f,w[i]));
            if(ff>0)
            {
                w[i]-=ff;w[i^1]+=ff;
                return ff;
            }
        }
    }
    return 0;
}
int dinic()
{
    int fl=0,f;
    while(1)
    {
        if(!bfs()) return fl;
        for(int i=1;i<=n;i++)
            iter[i]=head[i];
        while(f=dfs(s,inf)) fl+=f;
    }
    return fl;
}
main()
{
    for(T=read();T;T--)
    {
        G.tot=0;
        for(int i=1;i<=n;i++)
            G.head[i]=0;
        for(int i=1;i<=tot;i++)
            head[i]=0;
        tot=1;
        n=read();m=read();
        for(int i=1;i<=m;i++)
        {
            int a,b,c;
```

```

        a=read();b=read();c=read();
        G.u[i]=a;G.v[i]=b;G.c[i]=c;
        G.add(a,b,c);
    }
    G.dij();
    for(int i=1;i<=m;i++)
    {
        if(G.dis[G.u[i]]+G.c[i]==G.dis[G.v[i]])
        {
            add(G.u[i],G.v[i],G.c[i]);
            add(G.v[i],G.u[i],0);
        }
    }
    s=1;t=n;
    printf("%lld\n",dinic());
}
return 0;
}

```

F Typewriter

solved by wxg

题意

给了一个字符串，你现在要花最小代价构造出该字符串，有两种构造方法

1. 花 \$p\$ 代价在当前字符串后添加一个字符。
2. 花 \$q\$ 代价在当前字符串后添加一个当前字符串的字串

题解

\$dp[i]\$ 表示构造出长度 \$i\$ 的字符串的最小代价，有两种转移

1. \$dp[i]=dp[i-1]+p\$
2. \$dp[i]=dp[j]+q\$，\$j\$ 为满足 \$s[1\ldots j]\$ 中存在 \$s[j+1\ldots i]\$ 的子串

问题是 \$j\$ 怎么求？

首先随着 \$i\$ 增加 \$j\$ 是单调的

我们可以构建 \$s[1\ldots j]\$ 的后缀自动机，当 \$dp\$ 到 \$i\$ 时，我们在后缀自动机上匹配 \$s[i]\$，若匹配长度与 \$j\$ 的和大于等于 \$i\$，说明 \$j\$ 不用增加，反之则增加 \$j\$ 直到大于等于为止。

```

#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cstring>

```

```
#define ll long long
using namespace std;
int read()
{
    int k=0,f=1;char c=getchar();
    for(;!isdigit(c);c=getchar()) if(c=='-') f=-1;
    for(;isdigit(c);c=getchar()) k=k*10+c-'0';return k*f;
}
const int N=400055;
int n,m,ch[N][26],mxlen,P,Q;
ll dp[N];
int last,tot,link[N],len[N],size[N];
char s[N];
void init()
{
    for(int i=0;i<=tot;i++) memset(ch[i],0,sizeof(ch[i]));
    last=tot=0;len[0]=0;link[0]=-1;
}
void extend(int c)
{
    int cur=++tot,p=last,len[cur]=len[last]+1,size[cur]=1;
    for(;p!=-1&&!ch[p][c];p=link[p]) ch[p][c]=cur;
    if(p==-1) link[cur]=0;
    else
    {
        int q=ch[p][c];
        if(len[p]+1==len[q]) link[cur]=q;
        else
        {
            int clone=++tot;
            len[clone]=len[p]+1;
            link[clone]=link[q];
            memcpy(ch[clone],ch[q],sizeof(ch[q]));
            for(;p!=-1&&ch[p][c]==q;p=link[p])
                ch[p][c]=clone;
            link[q]=clone;link[cur]=clone;
        }
    }
    last=cur;
}
int get(int p,int c,int opt)
{
    if(ch[p][c])
    {
        p=ch[p][c];
        if(opt==1) mxlen++;
    }
    else
    {
        while(p!=-1&&!ch[p][c])
```

```

        p=link[p];
        if(p==-1) p=0,mxlen=0;
        else mxlen=len[p]+1,p=ch[p][c];
    }
    if(opt) return mxlen;
    return p;
}
int main()
{
    while(scanf("%s",s+1)!=EOF)
    {
        int l=strlen(s+1);
        init();scanf("%d%d",&P,&Q);
        int inslen=0, pos=0;mxlen=0;
        for(int i=1;i<=l;i++)
        {
            dp[i]=dp[i-1]+P;
            if(get(pos,s[i]-'a',1)+inslen>=i)
            {
                dp[i]=min(dp[i],dp[inslen]+Q);
                pos=get(pos,s[i]-'a',0);
            }
            else
            {
                while(get(pos,s[i]-'a',1)+inslen<i)
                {
                    extend(s[++inslen]-'a');
                }
                pos=get(pos,s[i]-'a',0);
                if(inslen<i)
                    dp[i]=min(dp[i],dp[inslen]+Q);
            }
            // cout<<inslen<<" "<<mxlen<<endl;
        }
        printf("%lld\n", dp[l]);
    }
    return 0;
}

```

I String

补题 by hxm

题意

给定一个只包含小写字母的字符串，要求选出一个长度为\$K\$的子序列，满足字典序最小，同时子序列中每个字母的出现次数都在一个各自给定的范围\$[L,R]\$内

题解

贪心。当前位置能选字典序小的就选字典序小的。对于子序的第*i*个位置，从*a*开始枚举到*z*尝试将当前位置后第一个该字符放入，然后看看后面剩下的子串能不能至少满足能构成长度为*K*的合法子串。

具体查看如下：1、剩下能用字符能否至少构成长度为*K* 2、剩下每个字符数量能否至少达到下界这个下界即要查看每个字符的剩余字符数是否足够，还要查看子序列剩余字符数能否提供所有的字符达到下界。

然后就完了。【很不明白比赛是怎么没调出来，，】

```
#include<algorithm>
#include<iostream>
#include<cstdlib>
#include<cstring>
#include<cstdio>
#include<vector>
#include<queue>
#include<cmath>
#include<map>
#include<set>
#define LL long long int
#define REP(i,n) for (int i = 1; i <= (n); i++)
#define Redge(u) for (int k = h[u].to; k; k = ed[k].nxt)
#define cls(s,v) memset(s,v,sizeof(s))
#define mp(a,b) make_pair<int,int>(a,b)
#define cp pair<int,int>
using namespace std;
const int maxn = 100005,maxm = 100005,INF = 0x3f3f3f3f;
inline int read(){
    int out = 0,flag = 1; char c = getchar();
    while (c < 48 || c > 57){if (c == '-') flag = 0; c = getchar();}
    while (c >= 48 && c <= 57){out = (out << 1) + (out << 3) + c - 48; c = getchar();}
    return flag ? out : -out;
}
vector<int> pos[26];
int pi[26],K,n;
int sum[maxn][26];
int cnt[maxn],L[maxn],R[maxn];
char s[maxn],ans[maxn];
void init(){
    for (int i = 0; i < 26; i++){
        cnt[i] = 0; pos[i].clear(); pi[i] = 0;
    }
}
int main(){
    while (~scanf("%s",s + 1)){
        init();
        K = read(); n = strlen(s + 1);
        for (int i = 0; i < 26; i++) L[i] = read(),R[i] = read();
```

```

for (int i = 1; i <= n; i++){
    int u = s[i] - 'a';
    pos[u].push_back(i);
    for (int j = 0; j < 26; j++) sum[i][j] = sum[i - 1][j];
    sum[i][u]++;
}
//cout << sum[2][0] << endl;
int u = 1, tag = true;
for (int i = 1; i <= K; i++){
    tag = false;
    for (int j = 0; j < 26; j++){
        //printf("[%d,%d]\n", i, j);
        if (cnt[j] >= R[j]) continue;
        while (pi[j] < pos[j].size() && pos[j][pi[j]] < u) pi[j]++;
        if (pi[j] >= pos[j].size()) continue;
        int tmp = 0;
        //puts("LXT");
        for (int k = 0; k < 26; k++) tmp += min(R[k] -
cnt[k], sum[n][k] - sum[pos[j][pi[j]] - 1][k]);
        if (tmp < K - i + 1) continue;
        //puts("LXT");
        int flag = true; tmp = 0;
        for (int k = 0; k < 26; k++){
            if (sum[n][k] - sum[pos[j][pi[j]] - 1][k] < L[k] -
cnt[k]) flag = false;
            if (k != j){
                if (L[k] > cnt[k]) tmp += L[k] - cnt[k];
            }
        }
        if (tmp > K - i) continue;
        //puts("LXT");
        if (!flag) continue;
        cnt[j]++;
        ans[i] = j + 'a';
        u = pos[j][pi[j]++] + 1;
        tag = true;
        //puts("pick");
        break;
    }
    if (!tag) {puts("-1"); break;}
}
if (tag){
    for (int i = 1; i <= K; i++) putchar(ans[i]);
    puts("");
}
}
return 0;
}

```

K Function

补题 by fyh

题意

求 $\sum_{i=1}^n \gcd(\lfloor \sqrt{i} \rfloor, i)$, $n \leq 10^{21}$

题解

很自然的对立方数进行划分

$\sum_{i=1}^n \gcd(\lfloor \sqrt{i} \rfloor, i) = \sum_{i=1}^n \sum_{j=(i-1)^3+1}^{i^3} \gcd(j, i)$ 我们发现这个式子出现了很多次

$$\sum_{i=1}^n \gcd(i, a) = \sum_{x|a} x \sum_{i=1}^n [\gcd(i, a) == x] = \sum_{x|a} x \sum_{i=1}^n [\lfloor \frac{i}{x} \rfloor \equiv 0 \pmod{x}]$$

$\sum_{i=1}^n \sum_{x|a} x \sum_{i=1}^n [\lfloor \frac{i}{x} \rfloor \equiv 0 \pmod{x}] = \sum_{k|x} k \sum_{i=1}^n [\lfloor \frac{i}{k} \rfloor \equiv 0 \pmod{\frac{k}{x}}] = \sum_{k|x} k \sum_{i=1}^n [\lfloor \frac{i}{k} \rfloor \equiv 0 \pmod{1}] = \sum_{k|x} k$

设 $xk=d$, 则上述式子等于

$\sum_{d|a} d \sum_{i=1}^n [\lfloor \frac{i}{d} \rfloor \equiv 0 \pmod{1}] = \sum_{d|a} d \sum_{i=1}^n [\lfloor \frac{i}{d} \rfloor = \lfloor \frac{i}{d} \rfloor]$ 故 $\sum_{i=1}^n \gcd(i, a) = \sum_{d|a} (\sum_{i=1}^n [\lfloor \frac{i}{d} \rfloor = \lfloor \frac{i}{d} \rfloor]) \phi(d)$, 对于原式右半部分的内容我们便可以通过数论分块在 $O(\sqrt{n})$ 的时间内解决。

继续展开左边的式子

$\sum_{i=1}^n \sum_{d|i} d \sum_{i=1}^n [\lfloor \frac{i}{d} \rfloor = \lfloor \frac{i}{d} \rfloor] = \sum_{d|a} d \sum_{i=1}^n [\lfloor \frac{i}{d} \rfloor = \lfloor \frac{i}{d} \rfloor]$

设 $xd=i$, 可继续写成

$\sum_{d=1}^n d \sum_{i=1}^n [\lfloor \frac{i}{d} \rfloor = \lfloor \frac{i}{d} \rfloor] = \sum_{d=1}^n d \sum_{i=1}^n [\lfloor \frac{xd}{d} \rfloor = \lfloor \frac{xd}{d} \rfloor] = \sum_{d=1}^n d \sum_{i=1}^n [\lfloor \frac{xd}{d} \rfloor = \lfloor \frac{xd}{d} \rfloor]$ 左边那个整除余数是 d , 右边余数是 $d-1$, 再进一步展开得

$\sum_{d=1}^n d \sum_{i=1}^n [\lfloor \frac{xd}{d} \rfloor = \lfloor \frac{xd}{d} \rfloor] = \sum_{d=1}^n d (3dx^2 + 3x + 1)$ 接下来就是平方和公式和等差数列求和,

设 $y = \lfloor \sqrt{d} \rfloor$, 得:

$\sum_{d=1}^n d \sum_{i=1}^n [\lfloor \frac{xd}{d} \rfloor = \lfloor \frac{xd}{d} \rfloor] = \sum_{d=1}^n d \sum_{y=1}^{\lfloor \sqrt{d} \rfloor} (y(y+1)(2y+1))$ y 也是一个整除形式, 故依旧可以用数论分块维护, 通过 $O(\sqrt{n})$ 预处理出 $\sum \phi(i)$ 和 $\sum \phi(i)$, 就可以在 $O(\sqrt{n})$ 的时间内处理每一组询问了。总时间复杂度 $O(\sqrt{n})$

注意, 读入要用 `_int128`, 但是在开数组的时候都要开 `int` 否则会爆空间。

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

```

#define mem(a,b) memset(a,b,sizeof(a))
typedef long long LL;
typedef pair<int,int> PII;
#define X first
#define Y second
inline __int128 read()
{
    __int128 x=0,f=1;char c=getchar();
    while(!isdigit(c)){if(c=='-')f=-1;c=getchar();}
    while(isdigit(c)){x=x*10+c-'0';c=getchar();}
    return x*f;
}
__int128 n,ans;
const int MOD=998244353,maxN=10000000;
int
T,prime[maxN+10],len,A,sqrt3N,phi[maxN+10],phii[maxN+10],pre[maxN+10],inv2=4
99122177;
bool vis[maxN+10];
void calc()
{
    phi[1]=1;phii[1]=1;
    for(int i=2;i<=maxN;i++)
    {
        if(!vis[i])prime[++len]=i,phi[i]=i-1;
        for(int j=1;j<=len;j++)
        {
            if(i*prime[j]>maxN)break;
            vis[i*prime[j]]=1;
            if(i%prime[j]==0){
                phi[i*prime[j]]=phi[i]*prime[j];
                break;
            }
            else phi[i*prime[j]]=phi[i]*(prime[j]-1);
        }
    }
    for(int i=1;i<=maxN;i++)phii[i]=((LL)i*(LL)phi[i])%MOD;
    for(int
i=1;i<=maxN;i++)pre[i]=((LL)pre[i-1]+(LL)phi[i])%MOD,phii[i]=((LL)phii[i-1]+
(LL)phii[i])%MOD;
}
__int128 sqrt3(__int128 N)
{
    __int128 l=0,r=1e9;
    while(r-l>1)
    {
        __int128 mid=(l+r)/2;
        if(mid*mid*mid<N)l=mid;
        else r=mid;
    }
    return (r*r*r<=N) ? r : l;
}

```

```
int main()
{
    calc();
    scanf("%d",&T);
    while(T--)
    {
        n=read();
        sqrt3N=(int)sqrt3(n);
        __int128 ans=0;
        for(int d=1;d*d<=sqrt3N;d++)
            if(sqrt3N%d==0)
            {
                ans=(ans+(n/d-
((__int128)sqrt3N*(__int128)sqrt3N*(__int128)sqrt3N-1)/d)%MOD*phi[d])%MOD;
                if(d*d!=sqrt3N)
                {
                    int t=sqrt3N/d;
                    ans=(ans+(n/t-
((__int128)sqrt3N*(__int128)sqrt3N*(__int128)sqrt3N-1)/t)%MOD*phi[t])%MOD;
                }
            }
        for(int l=1,r=0;l<=sqrt3N-1;l=r+1)
        {
            int x=(sqrt3N-1)/l;
            r=min(sqrt3N-1,(sqrt3N-1)/((sqrt3N-1)/l));
            LL tmp1=(phii[r]-phii[l-1]+MOD)%MOD,tmp2=(pre[r]-
pre[l-1]+MOD)%MOD;
            ans=(ans+tmp1*(LL)inv2%MOD*x%MOD*(x+1)%MOD*(2*x+1))%MOD;
            ans=((ans+tmp2*(x+1)%MOD*x%MOD*inv2%MOD*3%MOD)%MOD+x*tmp2%MOD)%MOD;
        }
        cout<<(LL)ans<<endl;
    }

    return 0;
}
```

训练实况

0~1h fyh摸鱼了15min,hxm读E wxg秒AD fyh写E，疯狂CE+RE 把数据量调大之后变成WA wxg&hxm思考F B

1~2h fyh仍在尝试E hxm读M与I,fyh放弃了E wxg开始写F

2~3h fyh读K L,与hxm交流I,澄清题意后口糊出I做法 wxgAF,hxm开始写M并写完，但是发现误解题意，后放弃 wxg打算重新写E

3~4h wxgAE ,fyh推K无果，开始写I

4~5h wxg,hxm讨论B，口糊出来，没有写 fyh&hxm调I无果

结果[]wxg全场最佳

训练总结

总结&反思：模板的整理是个大问题，本次用的全部是高中时候的板子[]fyh调板子题调了一年，问题出在哪最后也不知道（重写大法好）！没写出来是因为代码能力差+脑子持续掉线+细节没想清楚

改进

- 一个题至少要两个读，之后再核对题目大意是否一致，否则白给
- 板子的整理

From:
<https://wiki.cvbbacm.com/> - CVBB ACM Team



Permanent link:
https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:die_java:front_page_springtraining5&rev=1589554930

Last update: 2020/05/15 23:02