

# 2020牛客暑期多校训练营（第一场）

[比赛网址](#)

## 训练结果

- 时间:2020.7.12 12:00~17:00
- rank:43/1116
- 完成情况：4/6/10

## 题解

### A.B-Suffix Array

#### 题意

定义一个字符串的 B 函数为字符串到相同长度非负整数序列的映射，第  $i$  个整数表示字符串中在  $i$  前面与  $i$  字符相同的字符之间的最小距离，如果前面没有和自己一样的字符则记为 0。求每个后缀的 B 序列的排名。字符串只包含 a[]b 两个字母

#### 题解

solve by wxg

两个字符串的B函数如何比较大小呢？

首先，他们的公共前缀字符串的B函数一定是一样的，第一个不一样的字母一定可以比出大小，如果最长公共前缀只包含一个字母，那么和前一个字母不同的串排名小，如果包含两个字母，那么和前一个字母相同的串排名小。

只需要写一个 cmp 函数，用哈希求一下最长公共前缀，然后比一下，之后直接调用 sort 函数即可，注意 ab 和 ba 没有区别，所以我们比较的时候要把字符串首转化为字母相同的串。

```
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cstring>
#define ll long long
#define ull unsigned 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;
}
int n,m,pos[2],head1[100055],v1[100055],v2[100055];
int a[100055],b[100055],c[100055],sum[100055];
ull pw[100055],hs[100055],hs2[100055];
char s[100055];
ull query1(int x,int l)
{
    return hs[x+l-1]-hs[x-1]*pw[l];
}
ull query2(int x,int l)
{
    return hs2[x+l-1]-hs2[x-1]*pw[l];
}
bool cmp(int i,int j)
{
    if(s[i]==s[j])
    {
        int mid,l=1,r=n-max(i,j)+1,res=0;
        int len=r;
        while(r>=l)
        {
            mid=l+r>>1;
            if(query1(i,mid)==query1(j,mid))
                res=mid,l=mid+1;
            else r=mid-1;
        }
        if(len==res)
        {
            if(j>i) return 0;
            else return 1;
        }
        else
        {
            if(sum[i+res-1]-sum[i-1]==0||sum[i+res-1]-sum[i-1]==res)
            {
                if(s[i+res-1]!=s[i+res]) return 1;
                else return 0;
            }
            else
            {
                if(s[i+res-1]!=s[i+res]) return 0;
                else return 1;
            }
        }
    }
    else
    {
        int mid,l=1,r=n-max(i,j)+1,res=0;
        int len=r;
```

```

        while(r>=l)
        {
            mid=l+r>>1;
            if(query1(i,mid)==query2(j,mid))
                res=mid,l=mid+1;
            else r=mid-1;
        }
//    cout<<i<<" "<<j<<" "<<res<<endl;
    if(len==res)
    {
        if(j>i) return 0;
        else return 1;
    }
    else
    {
        if(sum[i+res-1]-sum[i-1]==0||sum[i+res-1]-sum[i-1]==res)
        {
            if(s[i+res-1]!=s[i+res]) return 1;
            else return 0;
        }
        else
        {
            if(s[i+res-1]!=s[i+res]) return 0;
            else return 1;
        }
    }
}
}
}
int main()
{
    pw[0]=1;
    for(int i=1;i<=100000;i++)
        pw[i]=pw[i-1]*31;
    while scanf("%d",&n)!=EOF
    {
        scanf("%s",s+1);
        for(int i=1;i<=n;i++)
        {
            c[i]=i;
            a[i]=s[i]-'a',b[i]=a[i]^1;
            sum[i]=sum[i-1]+a[i];
            hs[i]=hs[i-1]*31+a[i];
            hs2[i]=hs2[i-1]*31+b[i];
        }
        sort(c+1,c+1+n,cmp);
//        cout<<query1(1,1)<<" "<<query2(2,1)<<" ";
//        cout<<cmp(3,5)<<" ";
        for(int i=1;i<=n;i++)
            printf("%d ",c[i]);
        puts("");
    }
}

```

```
return 0;  
}
```

## B.Infinite Tree

### 题意

一棵无限大的树，每个点  $i$  与  $\frac{i}{\text{mindiv}(i)}$  连边，其中  $\text{mindiv}(i)$  表示  $i$  的最小约数，给定每个阶乘点  $i!$  的权值  $w_i$  求由  $1$  到  $n$  的阶乘组成的虚树的带权重心，即求  $\min_u \sum_{i=1}^n w_i \text{dis}(u, i!)$

### 题解

补题byfyh

题解都说要建一个虚树然后维护啥的，没用那么复杂的做法，方法跟 cf#614div2 的 F 有点像

暴力还原出树的路径来，因为阶乘的关系  $i!$  的路径和  $(i-1)!$  只有在  $i$  的地方不一样，所以我们只需要处理每一个  $i$  的路径即可，用欧拉筛预处理每个  $i$  的  $\text{mindiv}(i)$

现在尝试求  $u=1$  时候的答案:

首先对于  $i < j$  由于都是阶乘，后者都是包含前者的，所以前者怎么走  $(i, n]$  区间的都会跟着走，所以  $i$  每跳一步（设  $\text{tmp}=i, \text{tmp}$  到  $\frac{\text{tmp}}{\text{mindiv}(\text{tmp})}$  一直到  $1$ ），每走一步累加的权值是  $\sum_{k=i+1}^n w_k$ ，这个用前缀和维护即可。

然后开始考虑从  $1$  号节点出发走向哪里使得答案变小，设当前的区间为  $[l, r]$  表示我当前点所维护阶乘所在的子树区间，那么每走一步其实就是在看  $\text{sum}[l, r]$  和  $\text{sum}[1, l-1] + \text{sum}[r+1, n]$  的大小关系决定改怎么走，总复杂度  $O(n \log n)$

```
#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 int read()  
{  
    int 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;  
}  
const int maxN=100000;  
int n,End,len,prime[maxN+10],mdiv[maxN+10];
```

```
LL sum[maxN+10],ans;
PII a[maxN*25];
void init()
{
    for(int i=2;i<=maxN;i++)
    {
        if(!mdiv[i])prime[++End]=i,mdiv[i]=i;
        for(int j=1;j<=End && i*prime[j]<=maxN;j++)
        {
            mdiv[i*prime[j]]=prime[j];
            if(i%prime[j]==0)break;
        }
    }
}
bool cmp(PII a,PII b){
    return a.X==b.X ? a.Y<b.Y : a.X>b.X;
}
int main()
{
    init();
    while(scanf("%d",&n)!=EOF)
    {
        for(int i=1;i<=n;i++)sum[i]=read(),sum[i]+=sum[i-1];
        len=0;ans=0;
        for(int i=2;i<=n;i++)
        {
            int tmp=i;
            while(tmp>1)
            {
                a[++len]=make_pair(mdiv[tmp],i);
                tmp/=mdiv[tmp];
                ans+=sum[n]-sum[i-1];
            }
        }
        sort(a+1,a+len+1,cmp);
        int l=1,r=n;
        for(int i=1;i<=len;++i)
        {
            if(2ll*(sum[r]-sum[l-1])<sum[n])break;
            if(a[i].Y>r)continue;
            int k=max(l,a[i].Y);
            if(2ll*(sum[r]-sum[k-1])>sum[n])ans-=2ll*(sum[r]-sum[k-1])-
sum[n],l=k;
            else r=k-1;
        }
        printf("%lld\n",ans);
    }
    return 0;
}
```

## D.Quadratic Form

### 题意

给一个正定对称矩阵 $A$ ,和一个 $n$ 维向量 $b$ ,让你求 $n$ 维实向量,其中满足 $\sum_{i=1}^n \sum_{j=1}^n A_{ij} x_i x_j \leq 1$   $\sum_{i=1}^n b_i x_i$ 最大  
求 $(\sum_{i=1}^n b_i x_i)^2$ 的分数形式 $P \cdot Q^{-1} \pmod{998244353}$

### 题解

补题byfyh

题目那个两个求和式子是一个二次型,写成这样的形式 $x^T A x \leq 1$   $\max\{b^T \cdot x\}$  设 $a \cdot (x^T A x) = 1$ ,其中 $a \geq 1$ .

那么设 $x' = \sqrt{a} x$ ,原式即为 $(\sqrt{a} x)^T A (\sqrt{a} x) = (x')^T A x' = 1$

$b^T \cdot x' = b^T \cdot (\sqrt{a} x) \geq b^T \cdot x$  二次型值一定要为1

故变成: 约束条件 $x^T A x \leq 1$  目标函数 $b^T \cdot x$  考虑拉格朗日数乘法

$L(x_1, \dots, x_n) = b^T \cdot x - \lambda (x^T A x - 1)$  对每一个 $x_i$ 求偏导: $\frac{\partial L}{\partial x_i} = b_i - 2\lambda \sum_{j=1}^n A_{ij} x_j$

$\frac{\partial L}{\partial x_1} \dots \frac{\partial L}{\partial x_n} = b - 2\lambda A \cdot x$  以下简记 $2\lambda = \mu$   
 $x = \frac{1}{\mu} A^{-1} b$ ,带入二次型等于1

$\frac{1}{\mu} b^T (A^{-1})^T A \frac{1}{\mu} A^{-1} b = 1$  其中 $(A^{-1})^T = A^{-1}$  继续化简得 $\mu = \sqrt{b^T A^{-1} b}$  回代入 $ans = (b^T x)^2 = (b^T \frac{1}{\mu} A^{-1} b)^2 = b^T A^{-1} b$   
求逆矩阵即可,复杂度 $O(n^3)$

```
#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 int read()
{
    int 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;
}
const int maxn=210,MOD=998244353;
int n;
LL a[maxn][maxn<<1],b[maxn];
LL quickpow(LL a,int N,int mod)
```

```

{
    LL res=1,tmp=a;
    while(N)
    {
        if(N&1)res=(res*tmp)%mod;
        tmp=(tmp*tmp)%mod;
        N>>=1;
    }
    return res;
}
LL Inv(LL a){return quickpow(a,MOD-2,MOD);}
void gauss()
{
    for(int i=1;i<=n;i++)//按照列来枚举，当前之前i-1列全消完了
    {
        int k=i;
        for(int j=i+1;j<=n;j++)if(a[j][i]>a[k][i])k=j;//找一个系数绝对值最大的放在当前行，方便消元
        LL del=a[k][i];
        for(int j=i;j<=2*n;j++)swap(a[i][j],a[k][j]);
        for(int j=i;j<=2*n;j++)a[i][j]=(a[i][j]*Inv(del))%MOD;
        for(int j=1;j<=n;j++)
            if(j!=i)
            {
                del=a[j][i];
                for(int k=i;k<=2*n;k++)a[j][k]=(a[j][k]-
a[i][k]*del+(LL)MOD*(LL)MOD)%MOD;
            }
    }
}
int main()
{
    while(scanf("%d",&n)!=EOF)
    {
        for(int i=1;i<=n;i++)
        {
            for(int j=1;j<=n;j++)a[i][j]=read();
            for(int j=n+1;j<=2*n;j++)a[i][j]=0;
            a[i][i+n]=1;
        }
        for(int i=1;i<=n;i++)b[i]=read();
        LL ans=0;
        gauss();
        for(int i=1;i<=n;i++)
            for(int j=1;j<=n;j++)ans=(ans+a[i][j+n]*b[i]%MOD*b[j])%MOD;
        printf("%lld\n",ans);
    }
    return 0;
}

```

## H.Minimum-cost Flow

### 题意

给定一个网络，每次询问将每条边的容量设置为 $\frac{u_i}{v_i}$ 问在源点注入1流量，流尽后的最小费用，若流不完输出NaN.

### 题解

solve by fyh hxm

先考虑无解的情况 $\lceil \frac{v_i}{u_i} \rceil > \maxflow$ 是无解的，因为每一次都是流一次 $\frac{u_i}{v_i}$ 最多可以流 $\maxflow$ 次，如果 $\frac{u_i}{v_i} * \maxflow$ 还留不满1显然就是无解。

回忆最小费用最大流的做法，实质就是每次增广之前，在还剩余流量的弧跑最短路，沿着最短路增广，也就是前 $k$ 次增广的费用其实就是前 $k$ 次的最小费用，利用这个性质我们在算最小费用时把每次答案都记录下来即可。

```
#include<iostream>
#include<cmath>
#include<queue>
#include<cstdio>
#include<cstring>
#include<algorithm>
#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 BUG(s,n) for (int i = 1; i <= (n); i++) cout<<s[i]<<' '; puts("");
using namespace std;
const int maxn = 55,maxm = 220,INF = 1000000000;
inline int read(){
    int out = 0,flag = 1; char c = getchar();
    while (c < 48 || c > 57) {if (c == '-') flag = -1; c = getchar();}
    while (c >= 48 && c <= 57) {out = (out << 3) + (out << 1) + c - '0'; c = getchar();}
    return out * flag;
}
int n,m,K,h[maxn],ne = 2,S,T,Q;
struct EDGE{int to,nxt,f,w;}ed[maxm];
inline void build(int u,int v,int f,int w){
    ed[ne] = (EDGE){v,h[u],f,w}; h[u] = ne++;
    ed[ne] = (EDGE){u,h[v],0,-w}; h[v] = ne++;
}
int d[maxn],minf[maxn],p[maxn],inq[maxn],ans[110],len;
queue<int> q;
int mincost(){
```

```

int flow = 0, cost = 0;
while (true){
    for (int i = 0; i <= T; i++) d[i] = INF, inq[i] = 0;
    d[S] = 0; minf[S] = INF;
    q.push(S);
    int u;
    while (!q.empty()){
        u = q.front(); q.pop();
        inq[u] = false;
        Redge(u) if (ed[k].f && d[to = ed[k].to] > d[u] + ed[k].w){
            d[to] = d[u] + ed[k].w; p[to] = k; minf[to] =
min(minf[u], ed[k].f);
            if (!inq[to]) q.push(to), inq[to] = true;
        }
    }
    if (d[T] == INF) break;
    flow += minf[T];
    cost += minf[T] * d[T];
    ans[++len] = cost;
    u = T;
    while (u != S){
        ed[p[u]].f -= minf[T];
        ed[p[u] ^ 1].f += minf[T];
        u = ed[p[u] ^ 1].to;
    }
}
return flow;
}
int gcd(int a, int b){
    return b == 0 ? a : gcd(b, a % b);
}
int main(){
    int a, b, w, u, v;
    while (scanf("%d%d", &n, &m) != EOF)
    {
        ne = 2;
        for (int i = 1; i <= n; i++) h[i] = 0;
        for (int i = 0; i < m; i++) a = read(), b = read(), w = read(), build(a, b, 1, w);
        S = 1; T = n; len = 0;
        int FLOW = mincost();
        Q = read();
        while (Q--){
            u = read(); v = read();
            double tmp = (double)v / (double)u;
            if (tmp > (double)FLOW) printf("NaN\n");
            else
            {
                int ans1 = ans[v/u] * u, ans2 = v;
                if (v % u) ans1 += (ans[v/u + 1] - ans[v/u]) * (v % u);
                printf("%d/%d\n", ans1 / gcd(ans1, ans2), ans2 / gcd(ans1, ans2));
            }
        }
    }
}

```


```
    }  
  }  
}  
return 0;  
}
```

## 训练实况

12:00 读题  
12:00 wxg说结论 hxm开写  
12:30 hxm过F  
12:36 推出J wxg过J hxm想出了错误的做法 开写C fyh想I wxg想A  
13:10 hxm wa C 放弃C wxg开写A  
14:35之前 尝试想B未果，想出H  
14:35 wxg 过A fyh写H wxg hxm想B  
15:30 fyh过H

## 训练总结

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

Permanent link: [https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:die\\_java:front\\_page\\_summertrain1&rev=1594979014](https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:die_java:front_page_summertrain1&rev=1594979014) 

Last update: 2020/07/17 17:43