

2019 Multi-University Training Contest 1

比赛情况

比赛时间

2020-05-13 13:00-18:00

提交记录

E: Wrong Answer 2020-05-13 13:23:37

E: Wrong Answer 2020-05-13 13:28:32

E: Accepted 2020-05-13 13:32:38

D: Accepted 2020-05-13 13:38:53

B: Wrong Answer 2020-05-13 13:55:48

B: Accepted 2020-05-13 14:00:20

题解

A-Blank

补题 by Zars19

code:[HDU6578](#)

一个长度为 N 的序列被数字 $\{0,1,2,3\}$ 填充，有 M 个限制条件，限制区间 $[l_i, r_i]$ 中有 x_i 种不同数字。求方案数

题解 DP 用 $f[i][j][k][l]$ ($i \geq j \geq k \geq l$) 四维分别记录数字最后一次出现的位置，obviously 它可以向 $f[i+1][j][k][l]$, $f[i+1][i][k][l]$, $f[i+1][i][j][l]$ 和 $f[i+1][i][j][k]$ 转移。而限制条件在循环到每一个 i 时对 $r=i$ 的条件进行处理（将不满足者清零）。考虑到空间限制使用滚动数组。

B-Operation

solved by infinity37

<http://acm.hdu.edu.cn/showproblem.php?pid=6579>

题目大意

给你一个数列 a 有两种操作

1: 查询 a_l 到 a_r 间选若干数可计算出的最大异或和

2: 在数列 a 后新增一个数

数据范围

数列长 $n \leq 5e5$

操作数 $m \leq 5e5$

$1 \leq a_i \leq 2^{30}$

要求强制在线

题解

警惕多组数据

求子集的最大异或和我们可以求助线性基，但是此题要求某个范围内的线性基，而且数列有可能修改。

首先对于区间问题我们可以转化为 $(1, l-1)$ 和 $(1, r)$ 两个区间，可以发现，前缀线性基是很好求的，因为对于到第 i 位的线性基和 $i-1$ 位的线性基只差一个插入，所以我们可以快速的求出前缀线性基。

但是如何把 $(1, l-1)$ 的影响刨除呢？我们可以考虑在记录保留的数时，记录更偏右的数，然后在查询时查询只让位置在 i 之后的数产生影响即可。

官方题解是线段树维护线性基.....等周末学习一个。

代码

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <algorithm>
using namespace std;
const int N = 1e6+5;
const int M = 33;
int pre[N][M];
int sel[N][M];
int n;
int lastans = 0;
void append(int x) {
    x = x^lastans;
    n++;
    int cur = n;
    for (int i = 31; i >= 0; i--) {
        pre[n][i] = pre[n-1][i];
        sel[n][i] = sel[n-1][i];
```

```

}
for (int i = 31;i>=0;i--) {
    if ((x >> i) & 1) {
        if (!pre[n][i]) {
            pre[n][i] = x;
            sel[n][i] = cur;
            break;
        } else {
            if (cur > sel[n][i]) {
                swap(pre[n][i], x);
                swap(sel[n][i], cur);
            }
            x ^= pre[n][i];
        }
    }
}
}
}
}
void getans(int l,int r) {
    l = (l^lastans)%n+1;
    r = (r^lastans)%n+1;
    if (l>r) swap(l,r);
    int ans = 0;
    for(int i = 31;i>=0;i--) {
        if (sel[r][i] < l) continue;
        ans = max(ans,ans^pre[r][i]);
    }
    printf("%d\n",ans);
    lastans = ans;
}
int main()
{
    int cas,pn,m,opt,x,y;
    scanf("%d",&cas);
    while(cas--) {
        n = 0;
        lastans = 0;
        scanf("%d%d",&pn,&m);
        for (int i = 1;i<= pn;i++) {
            scanf("%d",&x);
            append(x);
        }
        //lastans = 0;
        for (int i = 1;i<= m;i++) {
            scanf("%d%d",&opt,&x);
            if (opt == 0) {
                scanf("%d",&y);
                getans(x,y);
            } else {
                append(x);
            }
        }
    }
}

```

```
}  
}  
}
```

D-Vacation

solved by Zars19

[D-Vacation](#)

code:[HDU6581](#)

给出当前车辆、以及位置在当前车辆之前的所有 n 辆车的长度、前端距停车线距离、最大速度 l_i, s_i, v_i 所有车都不可以超车，问当前车辆最快多久到达停车线。

题解：后来听说好多人是 $O(n \log n)$ 做的，不过其实没有必要，可以做到 $O(n)$ 假定最终形态是当前车辆刚好到达停车线而它前面的车辆一字排开，计算时间然后取最大值即可。

E-Path

solved by _wzx27

<http://acm.hdu.edu.cn/showproblem.php?pid=6582>

给一个图问割掉总权值最小的某些边使得最短路变长，先 dijkstra 跑一遍最短路，枚举边集把 $w + \text{dis}[u] = \text{dis}[v]$ 的边加入网络流图中，跑一遍最大流即可。

HDU 5889 几乎一样的题，看到以为能切，没想到因为 $\text{long}; \text{long}$ 的问题 WA 了两发。

```
#include<bits/stdc++.h>  
#define ll long long  
#define int long long  
#define pii_ pair<int,int>  
#define mp_ make_pair  
#define pb push_back  
#define fi first  
#define se second  
#define rep(i,a,b) for(int i=(a);i<=(b);i++)  
#define show1(a) cout<<#a<<" = "<<a<<endl  
#define show2(a,b) cout<<#a<<" = "<<a<<" ; "<<#b<<" = "<<b<<endl  
using namespace std;  
const ll INF = 1LL<<60;  
const int inf = 1<<30;  
const int maxn = 2e5+5;  
inline void fastio() {ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);}  
int n,m,tot,tot2,head[maxn],head2[maxn],cur[maxn],layer[maxn],dis[maxn];  
struct edge  
{  
    int u,v,w,nxt;  
}es[maxn<<1],es2[maxn<<1];
```

```

void add2(int u,int v,int w,int he[],edge e[],int &t)
{
    e[++t].u = u;
    e[t].v = v;
    e[t].w = w;
    e[t].nxt = he[u];
    he[u] = t;
}
void add(int u,int v,int w,int he[],edge e[],int &t)
{
    e[++t].u = u;
    e[t].v = v;
    e[t].w = w;
    e[t].nxt = he[u];
    he[u] = t;
}
void dijkstra()
{
    priority_queue<pii_> q;
    rep(i,1,n) dis[i] = INF;
    dis[1] = 0;
    q.push(mp_(0,1));
    while(!q.empty()){
        pii_ now = q.top();q.pop();
        int u = now.se;
        for(int i=head2[u];~i;i=es2[i].nxt){
            int v=es2[i].v,w=es2[i].w;
            if(dis[v]>dis[u]+w){
                dis[v] = dis[u]+w;
                q.push(mp_(-dis[v],v));
            }
        }
    }
}
bool bfs()
{
    memset(layer,0,sizeof(layer));
    queue<int>q;
    q.push(1);
    layer[1] = 1;
    while(q.size())
    {
        int u = q.front();q.pop();
        for(int i=head[u];i!=-1;i=es[i].nxt)
        {
            int v = es[i].v,w = es[i].w;
            if(w>0 && !layer[v])
            {
                layer[v] = layer[u]+1;
                q.push(v);
            }
        }
    }
}

```

```
    }
}
return layer[n]>0;
}
ll dfs(int u,ll flow)
{
    if(u==n) return flow;
    for(int& i=cur[u];i!=-1;i=es[i].nxt)
    {
        int v=es[i].v,w=es[i].w;
        if(w>0 && layer[v]==layer[u]+1)
        {
            int d = dfs(v,min(flow,w));
            if(d>0)
            {
                es[i].w -= d;
                es[i^1].w += d;
                return d;
            }
        }
    }
    return 0;
}
ll Dinic()
{
    ll ans = 0;
    while(bfs())
    {
        memcpy(cur,head,sizeof(cur));
        while(ll d=dfs(1,INF)) ans+=d;
    }
    return ans;
}
signed main()
{
    fastio();
    int _;
    for(cin>>_;;_--){
        memset(head,-1,sizeof(head));
        memset(head2,-1,sizeof(head2));
        tot = -1, tot2 = -1;
        cin>>n>>m;
        while(m--){
            int u,v,w;
            cin>>u>>v>>w;
            add2(u,v,w,head2,es2,tot2);
        }
        dijkstra();
        rep(i,0,tot2){
            int u=es2[i].u,v=es2[i].v,w=es2[i].w;
```

```

        if(dis[u]+w==dis[v])
    {add(u,v,w,head,es,tot);add(v,u,0,head,es,tot);}
    }
    cout<<Dinic()<<endl;
}

return 0;
}

```

F-Typewriter

补题 by infinity37

题目大意

给定一个字符串，给定两种方式构造它，第一种方式是添加任意字符，花费 p ；另一种方式是复制已存在的字符到当前串后，花费为 q 。问构造此字符串的最小花费。

数据范围

$|S| \leq 2 \times 10^5$ $1 \leq p, q \leq 2^{31}$

题解

一读题，老dp了（）

有两种转移方式，一种是 $f_i = f_{i-1} + p$

还有一种是 $f_i = f_j + q$ 。这个 j 要是确定 $j+1$ 到 i 这一段是 $1-j$ 中的一个子串。

转移有了，那么我们就需要一种方法来确定这个 j 。这是找之前存在过的子串，所以我们自然就想到了后缀自动机。具体的使用方式是这样的，设置一个指针 j 代表目前在后缀自动机插入到哪个字符，对于一个 i 确定目前的 $j+1$ 到 i 能否匹配，如果不能匹配就插入 $j+1$ 位置的字符，这样加入一直到可以匹配为止，然后用当前的指针作为 j 更新方程即可。

代码

```

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <algorithm>
using namespace std;
const int N = 2e5+10;
typedef long long ll;
struct SAM {

```

```
int len[N<<1],fa[N<<1],son[N<<1][26];
int size,last;
void init() {
    size = last = 1;
    memset(son[1],0,sizeof(son[1]));
}
void insert(char c) {
    int s = c-'a';
    int p = last,np = ++size;
    memset(son[np],0,sizeof(son[np]));
    last = np;
    len[np] = len[p]+1;
    for (;p && !son[p][s]; p = fa[p])
        son[p][s] = np;
    if (!p) { fa[np] = 1; }
    else {
        int q = son[p][s];
        if (len[p] + 1 == len[q]) { fa[np] = q; }
        else {
            int nq = ++size;
            len[nq] = len[p] + 1;
            memcpy(son[nq],son[q],sizeof(son[q]));
            fa[nq] = fa[q];
            fa[q] = fa[np] = nq;
            for (; son[p][s] == q; p = fa[p])
                son[p][s] = nq;
        }
    }
}
int get_next(int s,char ch,int &lens) {
    int id = ch-'a';
    if (son[s][id]) {
        lens++;
        return son[s][id];
    } else {
        while ( s && !son[s][id]) s = fa[s];
        if ( !s ) {
            s = 1;
            lens = 0;
        } else {
            lens = len[s]+1;
            s = son[s][id];
        }
        return s;
    }
}
int getlen(int s,char ch,int &lens) {
    int id = ch-'a';
    if (son[s][id]) {
        lens++;
    }
}
```

```

        } else {
            while (s && !son[s][id]) s = fa[s];
            if (!s) {
                s = 1;
                lens = 0;
            } else {
                lens = len[s]+1;
                s = son[s][id];
            }
        }
        return lens;
    }
}
sam;
ll f[N];
char s[N];
int main()
{
    while (scanf("%s",s+1)!=EOF) {
        sam.init();
        int p,q;
        scanf("%d%d",&p,&q);
        int len = strlen(s+1);
        int j = 0;
        int next = 1;
        int mlen = 0;
        for (int i = 1;i <= len;i++) {
            f[i] = f[i-1] + p;
            if (sam.getlen(next,s[i],mlen) + j >= i) {
                f[i] = min(f[i],f[j] + q);
                next = sam.get_next(next,s[i],mlen);
            } else {
                while (sam.getlen(next,s[i],mlen) + j < i) {
                    sam.insert(s[j+1]);
                    j = j+1;
                }
                next = sam.get_next(next,s[i],mlen);
                if (j < i)
                    f[i] = min(f[i],f[j] + q);
            }
        }
        printf("%lld\n",f[len]);
    }
    return 0;
}

```

I-String

补题 by _wzx27

<http://acm.hdu.edu.cn/showproblem.php?pid=6586>

为什么当时没想出来这个贪心的策略啊。。。贪心的选 k 位中的每一位，选之前判断一下后面是不是可能满足条件

```
#include<bits/stdc++.h>
#define ll long long
#define pii_ pair<int,int>
#define mp_ make_pair
#define pb push_back
#define fi first
#define se second
#define rep(i,a,b) for(int i=(a);i<=(b);i++)
#define show1(a) cout<<#a<<" = "<<a<<endl
#define show2(a,b) cout<<#a<<" = "<<a<<" "; cout<<#b<<" = "<<b<<endl
using namespace std;
const ll INF = 1LL<<60;
const int inf = 1<<30;
const int maxn = 3e5+5;
inline void fastio() {ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);}

char s[maxn],ans[maxn];
int bk[maxn][26],K,loc[26],L[26],R[26],ps;
vector<int> when[26];
bool check(int a,int pos)
{
    if(R[a]<=0) return 0;
    int tot = 0;
    rep(i,0,25){
        if(i!=a && bk[pos+1][i]<L[i]) return 0;
        tot += L[i];
    }
    if(L[a]>0) tot--;
    if(tot+ps+1>K) return 0;
    return 1;
}
int main()
{
    fastio();
    while(cin>>s+1){
        ps = 0;
        memset(bk,0,sizeof(bk));
        memset(loc,0,sizeof(loc));
        cin>>K;
        int len = strlen(s+1);
        for(int i=len;i>=1;i--){
            memcpy(bk[i],bk[i+1],sizeof(bk[i]));
            bk[i][s[i]-'a']++;
        }
        rep(i,0,25) when[i].clear();
        rep(i,1,len) when[s[i]-'a'].pb(i);
        rep(i,0,25) cin>>L[i]>>R[i];
    }
}
```

```

int pos = 0;
int flag = 0;
while(ps<K){
    flag = 0;
    rep(i,0,25){
        while(loc[i]<when[i].size() && when[i][loc[i]]<=pos) loc[i]++;
    }
    rep(i,0,25){
        if(when[i].size()==0) continue;
        int now = when[i][loc[i]];
        if(check(i,now)){
            flag = 1;
            pos = now;
            ans[ps] = i+'a';
            loc[i]++;
            R[i]=max(0,R[i]-1);
            L[i]=max(0,L[i]-1);
            ans[ps++] = 'a'+i;
            //cout<<ps-1<<" : "<<('a'+i)<<endl;
            break;
        }
    }
    if(!flag) {cout<<-1<<endl;break;}
}
ans[ps]='\0';
if(flag) cout<<ans<<endl;
}
return 0;
}

```

M-Code

补题 by Zars19

M-Code

code:

```

#include<iostream>
#include<cstdio>
#include<cstring>
#include<cstdlib>
#include<cmath>
#include<algorithm>
#define LL long long
#define INF 0x3f3f3f3f
#define eps 1e-10
using namespace std;
const int N=105;

```

```
int read()
{
    int x=0,f=1;char c=getchar();
    while(c<'0' || c>'9'){if(c=='-')f=-1;c=getchar();}
    while(c>='0'&&c<='9'){x=x*10+c-'0';c=getchar();}
    return x*f;
}
int dcmp(double x){return x<-eps?-1:(x>eps);}
struct Point
{
    double x,y;
    Point(double x=0,double y=0):x(x),y(y){}
    Point operator + (Point a){return Point(x+a.x,y+a.y);}
    Point operator - (Point a){return Point(x-a.x,y-a.y);}
    Point operator * (double a){return Point(a*x,a*y);}
    bool operator == (Point a){return dcmp(x-a.x)==0&&dcmp(y-a.y)==0;}
}d1[N],d2[N],p;
int top1,top2;
typedef Point Vector;
double sqr(double x){return x*x;}
double Cross(Vector v1,Vector v2){return v1.x*v2.y-v1.y*v2.x;}
double Dot(Vector v1,Vector v2){return v1.x*v2.x+v1.y*v2.y;}
double dissqr(Point p1,Point p2){return sqr(p1.x-p2.x)+sqr(p1.y-p2.y);}
bool cmp(Point p1, Point p2)
{
    int u=dcmp(Cross(p1-p,p2-p));
    return u>0 || (u==0&&dcmp(dissqr(p1,p)-dissqr(p2,p))<0);
}
int Graham(Point *d,int n)
{
    int k=0;
    for(int i=0;i<n;i++)
    {
        int u=dcmp(d[i].x-d[k].x);
        int v=dcmp(d[i].y-d[k].y);
        if(v<0 || (v==0&&u<0))k=i;
    }
    p=d[k];
    sort(d,d+n,cmp);
    n=unique(d,d+n)-d;
    if(n<=1)return n;
    if(dcmp(Cross(d[n-1]-d[0],d[1]-d[0]))==0){d[1]=d[n-1];return 2;}
    d[n++]=d[0];
    int m=1;
    for(int i=2;i<n;i++)
    {
        while(m>0&&dcmp(Cross(d[m]-d[m-1],d[i]-d[m-1]))<=0)m--;
        d[++m]=d[i];
    }
    return m;
}
```

```

}
bool SegInter(Point p1,Point p2,Point p3,Point p4)
{
    if(max(p1.x,p2.x)<min(p3.x,p4.x))return 0;
    if(max(p3.x,p4.x)<min(p1.x,p2.x))return 0;
    if(max(p1.y,p2.y)<min(p3.y,p4.y))return 0;
    if(max(p3.y,p4.y)<min(p1.y,p2.y))return 0;
    int u=dcmp(Cross(p2-p1,p3-p1))*dcmp(Cross(p2-p1,p4-p1));
    int v=dcmp(Cross(p4-p3,p1-p3))*dcmp(Cross(p4-p3,p2-p3));
    return u<=0&&v<=0;
}
double TriS(Point p1,Point p2,Point p3){return fabs(Cross(p2-p1,p3-p1)/2);}
bool InTri(Point p1,Point p2,Point p3,Point p0)
{
    double
s1=TriS(p1,p2,p0),s2=TriS(p1,p3,p0),s3=TriS(p2,p3,p0),s=TriS(p1,p2,p3);
    return dcmp(s-s1-s2-s3)==0;
}
int main()
{
    int T=read();
    while(T--)
    {
        top1=top2=0;
        int n=read();
        for(int i=1;i<=n;i++)
        {
            int x1=read(),x2=read(),y=read();
            if(y<0)d1[top1++]=Point(x1,x2);
            else d2[top2++]=Point(x1,x2);
        }
        top1=Graham(d1,top1);
        top2=Graham(d2,top2);
        bool f=1;
        for(int i=0;i<top1&&f;i++)
        for(int j=0;j<top2&&f;j++)
        if(SegInter(d1[i],d1[(i+1)%top1],d2[j],d2[(j+1)%top2]))f=0;
        for(int i=0;i<top1&&f;i++)
        for(int j=1;j<top2-1&&f;j++)
            if(InTri(d2[0],d2[j],d2[j+1],d1[i]))f=0;
        if(f)puts("Successful!");
        else puts("Infinite loop!");
    }
    return 0;
}

```

给出一个集合，集合中元素是 (\mathbf{x}_i, y_i) 其中 \mathbf{x}_i 是一个 d 维向量，本题中 $d=2$ $\mathbf{f}(\mathbf{x})=\text{sign}(\sum_{i=1}^d w_i x_i + b)=\text{sign}(\mathbf{w}^T \cdot \mathbf{x} + b)$

问有没有一个向量 \mathbf{w} 可以使得所有 $f(\mathbf{x}_i) = y_i$

题解：这题读题好难。。其实是计算几何，判断凸包是否相交 $x_1w_1 + x_2w_2 + b = 0$ 是二维平面上的
一条直线，一侧使得 $f(\mathbf{x}_i)$ 为 1 ，另一侧为 -1 。问题转化成有没有一条直线可以把两个
点集分开，于是又等价于对两个点集求凸包判断是否相交。判断凸包相交可以 $O(n^2)$ 做，先判两凸包
上的边两两是否线段相交，再看一凸包中的每一个顶点是否在另一凸包内。

replay

13:00-13:30

开场 _wzx27 开 E, infinity37 从前往后 Zars19 从后往前

_wzx27 提出最小割做法，三人讨论，确认可行，开始写题

infinity37 提出 B 像可持久化 trie _wzx27 提出像线性基 infinity37 和 Zars19 思考过后认为确实更像线性基

Zars19 提出 D 题做法

_wzx27 提交 E 错误 Zars19 开始写 D

经过两次修改 E 正确

13:30-14:00

infinity37 对 B 提出线性基前缀做法

Zars19 提交 D 正确

infinity37 开始写 B

_wzx27 和 Zars19 开 L, 认为 L 可贪心

infinity37 提交 B 错误

14:00-14:30

认识到没有注意多组数据的清空 infinity37 提交 B 正确

三人陆续读题，未果

14:30-15:30

一个小时内没有新的思路，于是结束了比赛，开始补题

总结

infinity37

本场比赛数学题居多，而队内三人都对数学题不太了解，因而做不出题目。另外，本场比赛中我发现了自己对后缀自动机的了解完全不够，只能写板子套题，而对后缀自动机的原理了解不够深刻，所以尽管看出F需要用后缀自动机解但是没能想明白应该如何解。因此我还需要对后缀自动机进行复习。

_wzx27

太菜了，要恶补数学（字符串的题也完全不会）。而且真的想5个小时感觉体力也不太够（不过男人不能说不行

Zars19

好...好难。只写出了一眼就想到的题。还需要多多做题，多多学习知识点。没有发现计算几何的题是道计算几何，在题目太长难读的时候要有抗压能力，要锻炼建模转换的思想，要有不屈的精神...

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

Permanent link:
https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:wangzai_milk:20200513%E6%AF%94%E8%B5%9B%E8%AE%B0%E5%BD%95&rev=1590298933

Last update: 2020/05/24 13:42