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

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$1\leq n\leq 5e5$

操作数m$1\leq m\leq 5e5$

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

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，所以我们就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l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$w+dis[u]==dis[v]$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$long\; long$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
```

```cpp
#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 = 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;
}
```

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Zars19f开IL,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…