省选级别试题 第四组题解

智商对决

题解

显然，若一场考试玩家得分高于NPC，则将其设定为u_i,否则设定为l_i

那么，令$goal = sigma(b_i * l_i)$，设定玩家在第i场考试得的前b_i分收益为l_i，后$X - b_i$分收益为r_i,要求总收益$\geq goal$

二分答案，考虑计算学习T小时可以获得的收益

注意到$u_i \geq l_i$，由调整法，可以证明最优的方案中有$[T/x]$场考试得了满分，另有一场考试得了$T\%X$分，因此枚举得了$T\%X$分的考试，按总收益选取最大的若干场满分考试

时间复杂度$O(NlogN + Nlog(nx))$

标准代码

c++ 11

```
#include<algorithm>
#include<cstdlib>
#include<cstring>
#include<cstdio>
using namespace std;
const int MAXN = 2e5 + 5;
typedef long long ll;
typedef long double ld;
typedef unsigned long long ull;
template <typename T> void chkmax(T &x, T y) {x = max(x, y); }
template <typename T> void chkmin(T &x, T y) {x = min(x, y); }
template <typename T> void read(T &x) {
    x = 0; int f = 1;
    char c = getchar();
    for (; !isdigit(c); c = getchar()) if (c == '-') f = -f;
    for (; isdigit(c); c = getchar()) x = x * 10 + c - '0';
    x *= f;
}
template <typename T> void write(T x) {
    if (x < 0) x = -x, putchar('-');
    if (x > 9) write(x / 10);
    putchar(x % 10 + '0');
}
template <typename T> void writeln(T x) {
    write(x);
    puts("");
}
int n, x, p[MAXN];
ll goal, b[MAXN], l[MAXN], r[MAXN], s[MAXN];
bool cmp(int x, int y) {
    return s[x] > s[y];
```

```
}
ll calc(int pos, ll t) {
    if (t <= b[pos]) return t * l[pos];
    else return b[pos] * l[pos] + (t - b[pos]) * r[pos];
}
ll check(ll t) {
    ll ans = 0, sum = 0;
    int lft = t % x;
    if (lft == 0) lft = x;
    int cnt = (t - lft) / x;
    //assert(cnt < n);
    for (int i = 1; i <= cnt; i++)
        sum += s[p[i]];
    for (int i = cnt + 1; i <= n; i++)
        chkmax(ans, sum + calc(p[i], lft));
    sum += s[p[cnt + 1]];
    for (int i = 1; i <= cnt; i++)
        chkmax(ans, sum - s[p[i]] + calc(p[i], lft));
    return ans;
}
int main() {
    read(n), read(x);
    for (int i = 1; i <= n; i++) {
        read(b[i]), read(l[i]), read(r[i]);
        s[i] = b[i] * l[i] + (x - b[i]) * r[i];
        p[i] = i, goal += b[i] * l[i];
    }
    sort(p + 1, p + n + 1, cmp);
    if (goal == 0) {
        writeln(0);
        return 0;
    }
    ll l = 1, r = 1ll * n * x;
    while (l < r) {
        ll mid = (l + r) / 2;
        if (check(mid) >= goal) r = mid;
        else l = mid + 1;
    }
    writeln(l);
    return 0;
}
```

是字符串还是图论

题解

设 $f[i][c]$ 表示从点i走到字母为c的点的最短距离，这个可以通过枚举c 然后 bfs 得到，时间复杂度 $O(n*sigma)$或 $O(n*sigma^2)$ 。对于两个点 i,j，它们的最短路有两种情况： 1.最短路上只经过相邻点的连边. 2.最短路上有经过同种字母之间的连边. 于是我们可以知道i和j的距离等于 $min(|i-j|,min(f[i][c]+1+f[j][c]))$ 。

再设$dist[c1][c2]$为字母为$c1$和$c2$的点的最小距离，显然$dist[s[i]][c] <= f[i][c] <= dist[s[i]][c]+1$，所以我们计算每个点的状态$mark[i]$，其中$mark[i][c]=f[i][c]-dist[s[i]][c]$，于是我们就可以从点和点的距离（通过$mark$）变成点和字母的距离啦，至于为什么不是字母和字母的距离，那是因为这样就不能排除情况1的最短路了。

到这里解决方法就差不多出来了，我们从1到n枚举点 i ，然后算出在 i 之前的哪些点与i距离属于情况1，这样的点一定是连续的，假设是now ~ i，然后我们统计1 ~ $now-1$中的两个值，其中$num[x][y]$=字母为 x 状态为 z（z包含y，即$y\&z=y$）的点的个数之和，$nums[x]$=字母为x的点的个数，时间复杂度$O(n*2^{sigma})$。

然后枚举点 i 到字母为 j 的点的距离dst（显然$dst=min($ $f[i][c]+1+dist[c][j]$+字母为 j 的点的$mark$值的第c位$)$）和个数cnt，时间复杂度$O(n*sigma^2)$，令$mindst=min($ $f[i][c]+1+dist[c][j]$ $)$，mrk表示状态（其中若$f[i][c]+1+dist[c][j]=mindst$则$mrk[c]=1$否则$mrk[c]=0$），若$num[j][mrk]>0$那么$dst=mindst+1, cnt=num[j][mrk]$，否则$dst=mindst, cnt=nums[j]$。然后用$dst$和$cnt$更新答案。 最后不要忘了还要判断一下 i 到now的距离。

标准代码

c++ 11

```
#include<cstdio>
#include<cstring>
#include<iostream>
using namespace std;
const int N=100100;
int n,f[N][16],dis[20][20];
char s[N];
int q[N],d[N];
int mark[N];
int c[N][1<<8];
inline void bfs(int c){
    int l=0,r=0;
    for(int i=1;i<=n;i++)if(s[i]-'a'==c){
        q[r++]=i,d[i]=0;
    }else   d[i]=-1;
    bool vis[16]={0};
    vis[c]=true;
    while(l<r){
        int now=q[l++];
        if(!vis[s[now]-'a']){
            vis[s[now]-'a']=true;
            for(int i=1;i<=n;i++)
                if(s[i]==s[now]&&d[i]==-1){
                    d[i]=d[now]+1;
                    q[r++]=i;
                }
        }
        if(now>1&&d[now-1]==-1) q[r++]=now-1,d[now-1]=d[now]+1;
        if(now<n&&d[now+1]==-1) q[r++]=now+1,d[now+1]=d[now]+1;
    }
    for(int i=1;i<=n;i++)
        if(d[i]!=-1)f[i][c]=d[i];
}
int main(){
   // freopen("1.out","w",stdout);
    scanf("%d",&n);
```

```
    scanf("%s",s+1);
    memset(f,0x3f,sizeof(f));
    for(int i=0;i<8;i++)
        bfs(i);
    memset(dis,0x3f,sizeof(dis));
    for(int i=1;i<=n;i++)
        for(int j=0;j<8;j++)
            dis[s[i]-'a'][j]=min(dis[s[i]-'a'][j],f[i][j]);
    for(int i=1;i<=n;i++)   for(int j=0;j<8;j++)
        if(f[i][j]>dis[s[i]-'a'][j])    mark[i]|=1<<j;
   // for(int i=1;i<=n;i++)
   //     printf("mark[%d]=%d\n",i,mark[i]);
    int ans=0;
    long long cnt=0;
    for(int i=1;i<=n;i++){
        for(int j=max(i-15,1);j<i;j++){
            int now=i-j;
            for(int k=0;k<8;k++)
                now=min(now,f[j][k]+1+f[i][k]);
            if(now==ans)    cnt++;
            if(now>ans)     ans=now,cnt=1;
        }
        int t=i-16;
        if(t>=1)    c[s[t]-'a'][mark[t]]++;
        for(int j=0;j<8;j++)    for(int k=0;k<256;k++)
            if(c[j][k]){
                int now=0x7fffffff;
                for(int l=0;l<8;l++){
                    now=min(now,dis[j][l]+1+f[i][l]+((k&(1<<l))>>l));
                   //printf("%d\n",(k&(1<<l))>>l);
                }
                if(now==ans)    cnt+=c[j][k];
                if(now>ans) ans=now,cnt=c[j][k];
            }
    }
    printf("%d %lld\n",ans,cnt);
}
```

夸父逐日

题解

拆点。每个点i拆成d个点$(i,1),(i,2)\dots(i,d)$，其中(i,j)代表星期j的第i个点

若$i\to j$有边，则由(i,t)向$(j,t\%d+1)$连边

缩点，在DAG上跑动态规划

标准代码

c++ 11

```
#include<bits/stdc++.h>
```

```
#define WALK(v,p) for(int i=hd[v][p],to;i;i=e[i].nxt)
using namespace std;
const int maxn=5e6+10;
int n,m,d,id[1<<17][51],o[maxn],k[maxn],vis[maxn],h[1<<17],t,tt;
int s[maxn],vl[maxn],f[maxn],a[1<<17][51],hd[2][maxn],w[maxn];
struct edge{int v,nxt;}e[maxn*2];
void add(int p,int u,int v){e[++t]={v,hd[p][u]},hd[p][u]=t;}
void dfs(int p){
    vis[p]=1;
    WALK(1,p)if(!vis[to=e[i].v])dfs(to);
    s[++t]=p;
}
void dfs1(int p){
    vis[p]=t;w[tt++]=p;
    if(!h[o[p]]&&a[o[p]][k[p]])++vl[t],h[o[p]]=1;
    WALK(0,p)if(!vis[to=e[i].v])dfs1(to);
}
int main(){
    scanf("%d%d%d",&n,&m,&d);
    for(int i=1;i<=n;i++)for(int j=0;j<d;j++)id[i][j]=++t,o[t]=i,k[t]=j;
    t=0;
    for(int i=1;i<=m;i++){
        int x,y;scanf("%d%d",&x,&y);
        for(int j=0;j<d;j++)
            add(0,id[x][j],id[y][(j+1)%d]),
            add(1,id[y][(j+1)%d],id[x][j]);
    }
    for(int i=1;i<=n;i++){
        getchar();
        for(int j=0;j<d;j++)
            a[i][j]=getchar()=='1';
    }
    t=0;
    for(int i=1;i<=n*d;i++)if(!vis[i])dfs(i);
    memset(vis,0,sizeof vis);t=0;
    for(int i=n*d;i;i--)if(!vis[s[i]]){
        tt=0;++t,dfs1(s[i]);f[t]=vl[t];
        for(int j=0;j<tt;j++){
            h[o[w[j]]]=0;

WALK(0,w[j])if(vis[to=e[i].v]!=t)f[t]=max(f[t],f[vis[to]]+vl[t]);
        }
    }
    printf("%d\n",f[vis[id[1][0]]]);
    return 0;
}
```