提高组高分试题 第一组 题解

2709 小w的铁路图

解

做法1

暴力枚举每条边(a_i,b_i)，将这条边从图上删去，后在图上dfs爆搜求a_i到b_i的最短路即可。期望得分10。

做法2

暴力枚举每条边(a_i,b_i)，将这条边从图上删去，后在图上bfs求a_i到b_i的最短路即可。时间复杂度$O(m^2)$，期望得分40。

做法3

因为图上没有重边，所以答案肯定不是1，即删去这条边(a_i,b_i)后a_i到b_i的最短路上除a_i外的经过的第一个点肯定不是b_i。我们需要求出以a_i为起点的，经过的第一个节点不是b_i的最短路，即次短路。

枚举图上的每个点S，计算以这个点为起点的所有边被删去后的最短路和次短路。从S开始向外搜索。维护数组$f[i][0/1]$，0表示最短路1表示次短路。对于每个还要状态记录$t[i][0/1]$表示这个状态下除了S外经过的第一个节点，使得次短路和最短路的除S外第一个经过的节点不同。注意从S点开始向外搜索的时候是不能回到S点的，这样会导致被删除掉的S出发的边重新被使用。

对于每条边(S,T)，$f[T][0]=1$，$f[T][1]$就是答案。

时间复杂度$O(n*m)$，期望得分100。

标准代码

C++11：

```
#include<bits/stdc++.h>
#define N 1010
#define M 100010
using namespace std;
int n,m,f[N][2],t[N][2],q[N*2],p[N*2],ans[N][N];
int A[M],B[M];
vector<int>to[N];
void work(int S)
{
  int l=1,r=1;
  for(int i=1;i<=n;i++)
    for(int j=0;j<=1;j++)f[i][j]=-1,t[i][j]=0;
    for (int i=0;i<to[S].size();i++)
    {
      q[r]=to[S][i];p[r]=0;
      f[q[r]][0]=1;
      t[q[r]][0]=to[S][i];
      r++;
    }
```

```
  while(l<r)
  {
    int x=q[l],y=p[l];l++;
    for(int i=0;i<to[x].size();i++)
    {
      if(to[x][i]==S)continue;
      if(f[to[x][i]][0]==-1)
      {
        f[to[x][i]][0]=f[x][y]+1;
        t[to[x][i]][0]=t[x][y];
        q[r]=to[x][i];p[r]=0;r++;
      }
      else
      {
        if(f[to[x][i]][1]!=-1||t[to[x][i]][0]==t[x][y])continue;
        f[to[x][i]][1]=f[x][y]+1;
        t[to[x][i]][1]=t[x][y];
        q[r]=to[x][i];p[r]=1;r++;
      }
    }
  }
}

int main()
{
  scanf("%d%d",&n,&m);
  for(int i=1;i<=m;i++)
  {
    scanf("%d%d",&A[i],&B[i]);
    to[A[i]].push_back(B[i]);
  }
  for(int i=1;i<=n;i++)
  {
    work(i);
    for(int j=1;j<=n;j++)ans[i][j]=f[j][1];
  }
  for(int i=1;i<=m;i++)
    printf("%d ",ans[A[i]][B[i]]);
  printf("\n");
  return 0;
}
```

2710 矩形的面积交

解

做法1

因为n个矩形两两不相交，所以询问矩形和它们的面积交等价于询问矩形和这n个矩形各自的面积交的和。

对于数据点1,2，$n,m \leq 5000$，对于每次询问暴力枚举n个矩形求面积交，加起来即可。

时间复杂度$O(n*m)$，期望得分20。

做法2

数据点3保证$W*L<=10^7$。可以开一个$W*L$的数组，对于每个矩形在数组的四个角落分别打标记。然后对于打标记的数组求前缀和。询问时，矩形的面积交等价于标记数组的二维前缀和，$ans(x1,y1,x2,y2)=f[x2][y2]-f[x1][y1]-f[x2][y1]+f[x1][y1]$。

时间复杂度$O(W*L)$，期望得分10。结合做法1期望得分20。

做法3

这个问题就是矩形加，矩形求和问题。可以使用树套树做法，比如线段树套线段树，或树状数组套主席树。对于n个矩形在对应区间做区间加，对于询问做区间求和即可。

时间复杂度$O(n*(logn)^2)$，期望得分60

做法4

将一个矩形$Q(x1,y1,x2,y2)$转化为两个前缀矩形$A(0,y1,x1,y2)$和$B(0,y1,x2,y2)$。矩形B和n个矩形的面积交减去矩形A和n个矩形的面积交就是矩形Q和n个矩形的面积交。

单独考虑一个矩形$F(x3,y3,x4,y4)$和矩形$A(0,y1,x1,y2)$的面积交。将矩形F也拆分成两个后缀矩形$F1(x3,y3,W,y4),F2(x4,y3,W,y4)$，$A$和$F$的面积交等价于$A$和$F1$的面积交$-A$和$F2$的面积交。

考虑A和$F1$的面积交，记$L=max(y1,y3)$，$R=min(y2,y4)$，相交的面积$S=(R-L)*(x1-x3)$。使用线段树辅助维护即可。

将所有拆分的矩形按x排序，对于修改的矩形在线段树上的对应区间$(y1+1,y2)$加上x，对于询问的矩形查询区间和sum，并统计区间被修改的长度和len，套用面积计算公式$S=len*x-sum$计算即可。

时间复杂度$O(nlogn)$，期望得分100。

标准代码

C++11：

```
#include<bits/stdc++.h>
#define ll long long
#define N 500010
using namespace std;
int n,m,W,L,tot,maxn;ll ans[N];
struct info{
  int x,l,r,inv,id;
  bool operator<(const info &p)const{return x<p.x;}
}s[N*4];
struct data{ll sum;int len;};
struct node{ll t1;int t2;data res;}t[N*4];

inline int get()
{
  char ch;int v;
  while(!isdigit(ch=getchar()));v=ch-48;
  while(isdigit(ch=getchar()))v=v*10+ch-48;
  return v;
}

class seg_tree
{
  void pushdown(int x,int l,int r)
  {
```

```
    int mid=l+r>>1,lc=x<<1,rc=lc+1;
    t[lc].t1+=t[x].t1;t[lc].t2+=t[x].t2;
    t[lc].res.sum+=(mid-l+1)*t[x].t1;
    t[lc].res.len+=(mid-l+1)*t[x].t2;
    t[rc].t1+=t[x].t1;t[rc].t2+=t[x].t2;
    t[rc].res.sum+=(r-mid)*t[x].t1;
    t[rc].res.len+=(r-mid)*t[x].t2;
    t[x].t1=0;t[x].t2=0;
  }
  data merge(data a,data b)
  {
    return (data){a.sum+b.sum,a.len+b.len};
  }
  public:
  void modify(int x,int l,int r,int ql,int qr,ll val,int inv)
  {
    if(ql<=l&&r<=qr)
    {
      t[x].t1+=val*inv;t[x].t2+=inv;
      t[x].res.sum+=val*inv*(r-l+1);
      t[x].res.len+=inv*(r-l+1);return;
    }
    int mid=l+r>>1,lc=x<<1,rc=lc+1;
    pushdown(x,l,r);
    if(ql<=mid)modify(lc,l,mid,ql,qr,val,inv);
    if(qr>mid)modify(rc,mid+1,r,ql,qr,val,inv);
    t[x].res=merge(t[lc].res,t[rc].res);
  }
  data qry(int x,int l,int r,int ql,int qr)
  {
    if(ql<=l&&r<=qr)return t[x].res;
    int mid=l+r>>1,lc=x<<1,rc=lc+1;
    pushdown(x,l,r);
    if(qr<=mid)return qry(lc,l,mid,ql,qr);
    if(ql>mid)return qry(rc,mid+1,r,ql,qr);
    return merge(qry(lc,l,mid,ql,qr),qry(rc,mid+1,r,ql,qr));
  }
}T;

int main()
{
  int X1,Y1,X2,Y2;
  scanf("%d%d%d%d",&n,&m,&W,&L);
  for(int i=1;i<=n;i++)
  {
    X1=get();Y1=get();X2=get();Y2=get();
    s[++tot]=(info){X1,Y1+1,Y2,1,0};
    s[++tot]=(info){X2,Y1+1,Y2,-1,0};
    maxn=max(maxn,Y2);
  }
  for(int i=1;i<=m;i++)
  {
    X1=get();Y1=get();X2=get();Y2=get();
    s[++tot]=(info){X1,Y1+1,Y2,-1,i};
    s[++tot]=(info){X2,Y1+1,Y2,1,i};
  }
  sort(s+1,s+tot+1);
  for(int i=1;i<=tot;i++)
```

```
  {
    if(!s[i].id)T.modify(1,0,maxn,s[i].l,s[i].r,s[i].x,s[i].inv);
    else
    {
      data tmp=T.qry(1,0,maxn,s[i].l,s[i].r);
      ans[s[i].id]+=s[i].inv*((ll)tmp.len*s[i].x-tmp.sum);
    }
  }
  for(int i=1;i<=m;i++)printf("%lld\n",ans[i]);
  return 0;
}
```

2711 重排题

解

一个数是11的倍数的充要条件是奇数位的和跟偶数位的和模11同余。所以，我们先对所有数字做一个背包，用$f[i][j]$表示选出i个数，并且这i个数的和模11余数为j的方案数。然后用逐位确定的方法：从高位到低位依次确定每一位最大可以是多少。对于每一位，先尝试能否填9，再尝试能否填8，再尝试能否填7……，以此类推。那么如何快速知道能否填这个数呢？一种简单的方法是对去掉这个数后剩下的数重新做一次背包，但是这样会很慢。那怎么办呢？我们只需要用反推的方法，把这个数从背包的可选数中去掉即可。

比方说，增加一个数的时候，代码如下：
for (int i=mx; i>=0; i--) for(int j=0; j<=10; j++) (f[i+1][(j+x)%11]+=f[i][j])%=P;
那么，去掉一个数的代码就是这样的：
for (int i=0; i<=mx; i++) for(int j=0; j<=10; j++) (f[i+1][(j+x)%11]+=P-f[i][j])%=P;

标准代码

C++11：

```
#include <bits/stdc++.h>
#define ft(i,a,b) for(int i=(a); i<=(b); ++i)
#define fd(i,a,b) for(int i=(a); i>=(b); --i)
#define fv(i,v) for(size_t i=0; i<(v).size(); ++i)
#define PB push_back
#define MP make_pair
#define F first
#define S second
using namespace std;

const int N=1050, P=998244353;

int f[N][11],mx;
int a[11];

void add(int x){
    fd(i,mx,0) ft(j,0,10) (f[i+1][(j+x)%11]+=f[i][j])%=P;
    a[x]++; mx++;
}

void del(int x){
```

```
    ft(i,0,mx) ft(j,0,10) (f[i+1][(j+x)%11]+=P-f[i][j])%=P;
    a[x]--; mx--;
}

char s[N];

int main(){
    scanf("%s",s+1);
    int n=strlen(s+1);
    f[0][0]=1;
    ft(i,1,n) add(s[i]-'0');

    int sum=0;
    ft(i,1,n) (sum+=s[i]-'0')%=11;
    int half=sum*6%11;
    if (f[n/2][half]==0){
        printf("-1\n");
        return 0;
    }

    int s0=half, s1=half;

    ft(i,1,n){
        //printf("%d %d %d\n",i,s0,s1);
        int aa,bb;
        if (i&1) { aa=n/2-i/2; bb=s0; }
            else { aa=(n+1)/2-(i+1)/2; bb=s1; }

        int x=9;
        while (true){
            if (!a[x]){
                x--; continue;
            }
            del(x);
            //if ((i&1) && f[n/2-i/2][s0]) break;
            //if (!(i&1) && f[(n+1)/2-(i+1)/2][s1]) break;
            if (f[aa][bb]) break;
            add(x);
            x--;
        }
        if (i&1) (s1+=(11-x))%=11;
            else (s0+=(11-x))%=11;

        //printf("%d %d %d\n",i,s0,s1);
        //printf("x==%d\n",x);
        putchar('0'+x);
    }
    putchar('\n');
    return 0;
}
```