

Contents

1 母版/基础/类/配置/黑科技

- 1.1 一般母版
- 1.2 高精度类
- 1.3 离散化
- 1.4 Linux 对拍
- 1.5 vimrc

2 数学

- 2.1 筛素数-欧拉筛法
- 2.2 高阶代数方程求根-求导

3 几何

- 3.1 最小圆覆盖-随机增量

4 博弈

5 DP

6 串

- 6.1 多模匹配-AC 自动机

7 图/树

- 7.1 单源最短路-Dijkstra
- 7.2 最短路-Floyd
- 7.3 单源最短路-SPFA
- 7.4 二分图最大匹配-匈牙利
- 7.5 有向图极大强连通分量-Tarjan 强连通
- 7.6 最大流-ISAP
- 7.7 最小生成树-Prim
- 7.8 最小生成树-Kruskal
- 7.9 树的直径-BFS
- 7.10 LCA-TarjanLCA

8 数据结构

- 8.1 并查集
- 8.2 区间和 _ 单点修改区间查询-树状数组
- 8.3 区间和 _ 区间修改单点查询-树状数组
- 8.4 区间和-线段树
- 8.5 区间第 k 大 _ 无修改-主席树
- 8.6 RMQ-ST

9 其它

1 母版/基础/类/配置/黑科技

1.1 一般母版

```
1 1 /*
2 2 Time:
3 3 Prob:
4 4 By RogerRo
5 5 */
6 #include<iostream>
7 #include<cstdio>
8 #include<cstdlib>
9 #include<cstring>
10 #include<vector>
11 #include<queue>
12 #include<set>
13 #include<map>
14 #include<cmath>
15 #include<algorithm>
16 #include<ctime>
17 #include<bitset>
18 #define ll long long
19 #define tr(i,l,r) for((i)=(l);(i)<=(r);++i)
20 #define rtr(i,r,l) for((i)=(r);(i)>=(l);--i)
21 #define oo 0x7F7F7F7F
22 using namespace std;
23 int read()
24 {
25     int x=0,f=1;
26     char ch=getchar();
27     while (ch<'0' || ch>'9') {if (ch=='-') f=-1; ch=getchar();}
28     while (ch>='0' && ch<='9') {x=x*10+ch-'0'; ch=getchar();}
29     return x*f;
30 }
31 void write(int x)
32 {
33     char a[20],s=0;
34     if (x==0){putchar('0'); return ;}
35     if (x<0) {putchar('-'); x=-x;}
36     while (x) {a[s++]=x%10+'0'; x=x/10;}
37     while (s-->0) putchar(a[s]);
38 }
39 void writeln(int x){write(x); putchar('\n');}
40 int main()
41 {
42
43     return 0;
44 }
```

1.2 高精度类

```
1 //要sqrt就一定要len和dcm是偶数
2 //不可以出现如big x=y;的东西，必须分开成big x;x=y;
```

```

3 #define len 3000
4 #define dcm 3000
5 void carry(int*x,int y){*(x-1)+=((*x+=y)+10000)/10-1000;*x=(*x+10000)%10;}
6 struct big
7 {
8     int _[len+2];
9
10    int& operator[](int x){return _[x];}
11    big(){memset(_,0,sizeof(int)*(len+2));}
12    big(char*x)
13    {
14        memset(_,0,sizeof(int)*(len+2));
15        char *y=x+strlen(x)-1,*z=strchr(x,'.').*i;
16        if(!z) z=y+1;
17        int t=dcm-(z-x);
18        tr(i,x,y) if(i!=z&&t>=1&&t<=len) _[++t]=*i-'0';
19    }
20
21    big& operator=(const big&x){memcpy(_,x._,sizeof(int)*(len+2));return *this;}
22    char* c_str()
23    {
24        char *s=new char[len]; int l,r,i=0,k;
25        tr(l,1,len) if(_[l]>0||l==dcm) break;
26        rtr(r,len,1) if(_[r]>0||r==dcm) break;
27        tr(k,l,r){if(k==dcm+1)s[i++]='.';s[i++]=_[k]+'0';}
28        s[i]=0; return s;
29    }
30
31    friend int comp(big x, big y) //0(len)
32    {
33        int i;
34        tr(i,1,len) if(x[i]!=y[i]) break;
35        return i>len?0:(x[i]>y[i]?1:-1);
36    }
37    friend big operator+(big x, big y) //0(len)
38    {
39        big z; int i;
40        rtr(i,len,1) carry(&z[i],x[i]+y[i]);
41        return z;
42    }
43    friend big operator-(big x, big y) //0(len)
44    {
45        big z; int i;
46        rtr(i,len,1) carry(&z[i],x[i]-y[i]);
47        return z;
48    }
49    friend big operator*(big x, big y) //0(len^2)
50    {
51        big z; int i,j;
52        rtr(i,len,1) rtr(j,min(dcm+len-i,len),max(dcm+1-i,1))
53            carry(&z[i+j-dcm],x[i]*y[j]);
54        return z;
55    }
56    friend big operator/(big x, big y) //0(len^2)
57    {

```

```

58        big z,t,tmp[10]; int i,j,k;
59        tr(k,1,9) tmp[k]=tmp[k-1]+y;
60        tr(j,1,len-dcm) t[j+dcm]=x[j];
61        j--;
62        tr(i,1,len)
63        {
64            tr(k,1,len-1) t[k]=t[k+1];
65            t[len]=++j<=len?x[j]:0;
66            tr(k,1,9) if(comp(tmp[k],t)>0) break;
67            z[i]=--k;
68            t=t-tmp[k];
69        }
70        return z;
71    }
72    friend int sqrt_deal(big&y,int a,int b,int l)
73    {
74        int t=a+y[b]%10-9;
75        if(2*b>l)t--=(y[2*b-l])/10;
76        if(b>=0&&!(a=sqrt_deal(y,t/10,b-1,l))) y[b]+=(t+999)%10-y[b]%10;
77        return a;
78    }
79    friend big sqrt(big x) //0(len^2)
80    {
81        int l,t=dcm/2; big y,z; y=x;
82        for(l=1;l<=len;l++)
83        {
84            y[++l]=10;
85            while(!sqrt_deal(y,0,l,l)) y[l]+=20;
86            z[++t]=y[l]/20; y[l]=10;
87        }
88        return z;
89    }
90    friend big floor(big x)
91    {
92        big z; z=x; int i;
93        tr(i,dcm+1,len) z[i]=0;
94        return z;
95    }
96    friend big ceil(big x){return comp(x,floor(x))==0?x:floor(x+big("1"));}
97 }

```

1.3 离散化

```

1 //dc[1,2,...]=[x1,x2,...]; rdc(x1,x2,...)=1,2,...
2 int n,a[maxn],dc[maxn];
3 int rdc(int x){return lower_bound(dc+1,dc+num+1,x)-dc;}
4 void init()
5 {
6     //...
7     memcpy(dc,a,(n+1)*sizeof(int));
8     sort(dc+1,dc+n+1);
9     num=unique(dc+1,dc+n+1)-(dc+1);
10 }

```

1.4 Linux 对拍

```

1 g++ $2 -o 1.out
2 g++ $3 -o 2.out
3 cnt=0;
4 while true; do
5 g++ $1 -o dm.out
6 ./dm.out>dm.txt
7 ./1.out<dm.txt>1.txt
8 ./2.out<dm.txt>2.txt
9 if diff 1.txt 2.txt; then let "cnt+=1"; echo ${cnt};
10 else exit 0;
11 fi
12 done

```

1.5 vimrc

```

1 " All system-wide defaults are set in $VIMRUNTIME/debian.vim and sourced by
2 " the call to :runtime you can find below. If you wish to change any of those
3 " settings, you should do it in this file (/etc/vim/vimrc), since debian.vim
4 " will be overwritten everytime an upgrade of the vim packages is performed.
5 " It is recommended to make changes after sourcing debian.vim since it alters
6 " the value of the 'compatible' option.
7
8 " This line should not be removed as it ensures that various options are
9 " properly set to work with the Vim-related packages available in Debian.
10 runtime! debian.vim
11
12 " Uncomment the next line to make Vim more Vi-compatible
13 " NOTE: debian.vim sets 'nocompatible'. Setting 'compatible' changes numerous
14 " options, so any other options should be set AFTER setting 'compatible'.
15 "set compatible
16
17 " Vim5 and later versions support syntax highlighting. Uncommenting the next
18 " line enables syntax highlighting by default.
19 if has("syntax")
20     syntax on
21 endif
22
23 " If using a dark background within the editing area and syntax highlighting
24 " turn on this option as well
25 "set background=dark
26
27 " Uncomment the following to have Vim jump to the last position when
28 " reopening a file
29 if has("autocmd")
30     au BufReadPost * if line("'\"") > 1 && line("'\"") <= line("$") | exe "
31         normal! g'\"" | endif
32 endif
33
34 " Uncomment the following to have Vim load indentation rules and plugins
35 " according to the detected filetype.
36 if has("autocmd")
37     filetype plugin indent on

```

```

37 endif
38
39 " The following are commented out as they cause vim to behave a lot
40 " differently from regular Vi. They are highly recommended though.
41 "set showcmd      " Show (partial) command in status line.
42 "set showmatch     " Show matching brackets.
43 "set ignorecase    " Do case insensitive matching
44 "set smartcase     " Do smart case matching
45 "set incsearch     " Incremental search
46 "set autowrite     " Automatically save before commands like :next and :make
47 "set hidden       " Hide buffers when they are abandoned
48 "set mouse=a      " Enable mouse usage (all modes)
49
50 " Source a global configuration file if available
51 if filereadable("/etc/vim/vimrc.local")
52     source /etc/vim/vimrc.local
53 endif
54
55 " FOR TRAINING
56 execute pathogen#infect()
57 filetype plugin indent on
58 map <F6> gg0/*<CR><TAB>Time:<CR>Prob:<CR>By RogerRo<CR><BACKSPACE><BACKSPACE>
59     >/<CR>#include<iostream><CR>#include<cstdio><CR>#include<cstring><CR>#
60     include<cstdlib><CR>#include<set><CR>#include<queue><CR>#include<cmath><CR>
61     >#include<vector><CR>#include<algorithm><CR>#include<map><CR>#define ll
62     long long<CR>#define oo 0x7F7F7F7F<CR>#define tr(i,l,r) for((i)=(l);(i)<=
63     (r);++i)<CR>using namespace std;<CR>int read()<CR>{<CR><TAB>int x=0,f=1;<CR>
64     >char ch=getchar();<CR>while (!(ch>='0' && ch<='9')) {if (ch=='-') f=-1;
65     ch=getchar();}<CR>while (ch>='0'&&ch<='9') {x=x*10+ch-'0'; ch=getchar();}<
66     CR>return x*f;<CR>}<CR><CR>int main()<CR>{<CR><CR><TAB>return 0;<CR>}<CR><
67     ESC>
68
69 " FOR CONTEST
70 colo torte
71 set nu
72 set ts=4
73 set sw=4
74 map <C-A> ggVG"+y
75 map <F2> :w<CR>
76 map <F3> :browse e<CR>
77 map <F4> :browse vsp<CR>
78 map <F5> :call Run()<CR>
79 func! Run()
80     exec "w"
81     exec "!g++ -Wall % -o %"
82     exec "!./%"
83 endfunc

```

2 数学

2.1 筛素数-欧拉筛法

 $O(N)$

```

1 int prime[maxm],a[n];
2 bool pprime[n];
3 void EulerPrime()
4 {
5     int i,j;
6     tr(i,2,n) pprime[i]=1;
7     tr(i,2,n)
8     {
9         if (pprime[i]) prime[++m]=i;
10        tr(j,1,m)
11        {
12            if (i*prime[j]>n) break;
13            pprime[i*prime[j]]=0;
14            if (i%prime[j]==0) break;
15        }
16    }
17 }

```

2.2 高阶代数方程求根-求导

$O(N^3 * S)$, S 取决于精度

```

1 //求导至最高次为t时, a[t][i]表x^i的系数, ans[t]记录根; oo依题而定
2 double a[maxn][maxn],ans[maxn][maxn];
3 int n,anss[maxn];
4 double get(int x,double y)
5 {
6     int i; double res=0;
7     rtr(i,x,0) res=res*y+a[x][i];
8     return res;
9 }
10 void dich(int x,double ll,double rr)
11 {
12     if (cmp(get(x,ll))==0){ans[x][++anss[x]]=ll;return;}
13     if (cmp(get(x,rr))==0){ans[x][++anss[x]]=rr;return;}
14     if (cmp(get(x,ll)*get(x,rr))>0) return;
15     double l=ll,r=rr,mid;
16     while (l+eps<r) //亦可改为循环一定次数
17     {
18         int tl=cmp(get(x,l)),tm=cmp(get(x,mid=(l+r)/2));
19         if (tl==0) break;
20         if (tl*tm>=0) l=mid; else r=mid;
21     }
22     ans[x][++anss[x]]=l;
23 }
24 void work()
25 {
26     int i,j; double l,r;
27     rtr(i,n-1,1) tr(j,0,i) a[i][j]=a[i+1][j+1]*(j+1);
28     tr(i,0,n-1)
29     {
30         l=-oo;
31         tr(j,1,anss[i]){dich(i+1,l,r=ans[i][j]); l=r;}
32         dich(i+1,l,oo);
33     }

```

```

34     tr(i,1,anss[n]) printf("%.10lf\n",ans[n][i]);
35 }

```

3 几何

3.1 最小圆覆盖-随机增量

$O(N)$

```

1 int n;
2 bool out(point a,circle b){return cmp(len(a-b.o)-b.r)==1;}
3 circle get(point a){return circle(a,0);}
4 circle get(point a,point b){return circle((a+b)/2,len(a-b)/2);}
5 circle get(point a,point b,point c){return outcircle(triangle(a,b,c));}
6 void work()
7 {
8     int i,j,k;
9     srand(time(0));
10    circle t;
11    random_shuffle(a+1,a+n+1);
12    for(i=2,t=get(a[1]);i<=n;i++) if (out(a[i],t))
13        for(j=1,t=get(a[i]);j<i;j++) if (out(a[j],t))
14            for(k=1,t=get(a[i],a[j]);k<j;k++) if (out(a[k],t))
15                t=get(a[i],a[j],a[k]);
16    printf("%.2lf %.2lf %.2lf\n",t.o.x,t.o.y,t.r);
17 }

```

4 博弈

5 DP

6 串

6.1 多模匹配-AC 自动机

求 n 个模式串中有多少个出现过, 模式串相同算作多个, $O(\sum P_i + T)$

```

1 //maxt=文本串长, maxp=模式串长, maxn=模式串数
2 struct ac{int s,to[26],fail;} a[maxn*maxp];
3 int m,n;
4 char ts[maxp],s[maxt];
5 queue<int> b;
6 void clear(int x)
7 {
8     a[x].s=a[x].fail=0;
9     memset(a[x].to,0,sizeof(a[x].to));
10 }
11 void ins(char *st)
12 {
13     int i,x=0,c,l=strlen(st);
14     tr(i,0,l-1)

```

```

15     {
16         if (!a[x].to[c==st[i]-'a']) {a[x].to[c]=++m; clear(m);}
17         x=a[x].to[c];
18     }
19     a[x].s++;
20 }
21 void build()
22 {
23     int i,h,t;
24     tr(i,0,25) if (t=a[0].to[i]) b.push(t);
25     while (b.size())
26     {
27         h=b.front(); b.pop();
28         tr(i,0,25)
29         if (t=a[h].to[i])
30         {
31             a[t].fail=a[a[h].fail].to[i];
32             b.push(t);
33         } else a[h].to[i]=a[a[h].fail].to[i];
34     }
35 }
36 int cnt(char *st)
37 {
38     int i,x=0,c,t,cnt=0,l=strlen(st);
39     tr(i,0,l-1)
40     {
41         c=st[i]-'a';
42         while (!a[x].to[c]&&x) x=a[x].fail;
43         x=a[x].to[c];
44         for(t=x;t&&a[t].s>-1;t=a[t].fail) {cnt+=a[t].s; a[t].s=-1;}
45     }
46     return cnt;
47 }
48 void work()
49 {
50     int i;
51     m=0; clear(0);
52     scanf("%d",&n);
53     tr(i,1,n)
54     {
55         scanf("%s",ts); ins(ts);
56     }
57     build();
58     scanf("%s",s); printf("%d\n",cnt(s));
59 }

```

7 图/树

7.1 单源最短路-Dijkstra

不加堆, $O(V^2 + E)$

```

1 struct edge{int pre,x,y,d;} a[maxm];
2 int n,m,ah[maxn],d[maxn];

```

```

3 bool p[maxn];
4 void update(int x)
5 {
6     int e;
7     p[x]=true;
8     for(e=ah[x];e>-1;e=a[e].pre)
9         if (!p[a[e].y]&&(!d[a[e].y]||a[e].d+d[x]<d[a[e].y]))
10             d[a[e].y]=a[e].d+d[x];
11 }
12 void dijkstra()
13 {
14     int i,j,t;
15     memset(p,0,sizeof(p));
16     update(1);
17     d[0]=oo;
18     tr(i,2,n)
19     {
20         t=0;
21         tr(j,1,n) if (!p[j]&&d[j]&&d[j]<d[t]) t=j;
22         update(t);
23     }
24     printf("%d\n",d[n]);
25 }

```

加堆, $O(E \log E + V)$

```

1 typedef pair<int,int> pa;
2 struct edge{int pre,x,y,d;} a[maxm];
3 int n,m,ah[maxn],ans[maxn];
4 priority_queue<pa,vector<pa>,greater<pa> >d;
5 bool p[maxn];
6 void dijkstra()
7 {
8     int v,s,e;
9     memset(p,0,sizeof(p));
10    d.push(make_pair(0,1));
11    while(!d.empty())
12    {
13        v=d.top().second;
14        s=d.top().first;
15        d.pop();
16        if (p[v]) continue;
17        p[v]=1;
18        ans[v]=s;
19        for(e=ah[v];e>-1;e=a[e].pre)
20            if (!p[a[e].y]) d.push(make_pair(s+a[e].d,a[e].y));
21    }
22    printf("%d\n",ans[n]);
23 }

```

7.2 最短路-Floyd

$O(V^3 + E)$

```

1 void floyd()

```

```

2 {
3     int i,j,k;
4     tr(k,1,n) tr(i,1,n)
5     if (a[i][k]) tr(j,1,n)
6     if (i!=j&&a[k][j]&&(!a[i][j]|| (a[i][j]&&a[i][k]+a[k][j]<a[i][j])))
7         a[i][j]=a[i][k]+a[k][j];
8 }

```

7.3 单源最短路-SPFA

不加优化, $O(VE + V^2) = O(kE)$

```

1 struct edge{int pre,x,y,d;} a[maxm];
2 int n,m,last[maxn],d[maxn],b[maxn];
3 bool p[maxn];
4 void spfa()
5 {
6     int h,t,e;
7     memset(d,0x7F,sizeof(d));
8     memset(p,0,sizeof(p));
9     b[0]=1; p[1]=1; d[1]=0;
10    h=n-1; t=0;
11    while (h!=t)
12    {
13        h=(h+1)%n;
14        for (e=last[b[h]];e>-1;e=a[e].pre)
15            if (d[a[e].x]+a[e].d<d[a[e].y])
16            {
17                d[a[e].y]=d[a[e].x]+a[e].d;
18                if (!p[a[e].y])
19                {
20                    t=(t+1)%n;
21                    b[t]=a[e].y;
22                    p[a[e].y]=1;
23                }
24            }
25        p[b[h]]=0;
26    }
27    printf("%d\n",d[n]);
28 }

```

SLF+LLL 优化, $O(VE + V^2) = O(kE)$

```

1 //a从1开始!
2 struct edge{int pre,x,y,d;} a[maxm];
3 int n,m,last[maxn],d[maxn],b[maxn];
4 bool p[maxn];
5 void spfa()
6 {
7     int e,h,t,sum,num;
8     memset(d,0x7F,sizeof(d));
9     memset(p,0,sizeof(p));
10    b[0]=1; p[1]=1; d[1]=0;
11    sum=0; num=1;
12    h=0; t=0;

```

```

13 while (num)
14 {
15     while (d[h]*num>sum)
16     {
17         t=(t+1)%n;
18         b[t]=b[h];
19         h=(h+1)%n;
20     }
21     e=last[b[h]];
22     p[b[h]]=0;
23     num--;
24     sum-=d[a[e].x];
25     h=(h+1)%n;
26     for (;a[e].x;e=a[e].pre)
27         if (d[a[e].x]+a[e].d<d[a[e].y])
28         {
29             if (p[a[e].y]) sum-=d[a[e].y];
30             d[a[e].y]=d[a[e].x]+a[e].d;
31             sum+=d[a[e].y];
32             if (!p[a[e].y])
33             {
34                 if (num && d[a[e].y]<d[b[h]])
35                 {
36                     h=(h+1)%n;
37                     b[h]=a[e].y;
38                 } else
39                 {
40                     t=(t+1)%n;
41                     b[t]=a[e].y;
42                 }
43                 p[a[e].y]=1;
44                 num++;
45             }
46         }
47     }
48     printf("%d\n",d[n]);
49 }

```

7.4 二分图最大匹配-匈牙利

$O(VE)$

```

1 struct edge{int x,y,pre;} a[maxm];
2 int nx,ny,m,last[maxn],my[maxn];
3 bool p[maxn];
4 int dfs(int x)
5 {
6     for (int e=last[x];e>-1;e=a[e].pre)
7         if (!p[a[e].y])
8         {
9             int y=a[e].y;
10            p[y]=1;
11            if (!my[y]||dfs(my[y])) return my[y]=x;
12        }
13    return 0;

```

```

14 }
15 void hungary()
16 {
17     int i,ans=0;
18     memset(my,0,sizeof(my));
19     tr(i,1,nx)
20     {
21         memset(p,0,sizeof(p));
22         if (dfs(i)) ans++;
23     }
24     printf("%d\n",ans);
25 }

```

7.5 有向图极大强连通分量-Tarjan 强连通

$O(V + E)$

```

1 //ds, ss, gs分别是dfn, sta, group计数器；group记所属分量号码，size记分量大小；
  insta记是否在栈中
2 struct edge{int x,y,pre;} a[maxm];
3 int n,m,ah[maxn],ds,dfn[maxn],low[maxn],ss,sta[maxn],gs,group[maxn],size[maxn]
  ];
4 bool insta[maxn];
5 void tarjan(int x)
6 {
7     int e,y,t;
8     dfn[x]=low[x]=++ds;
9     sta[++ss]=x; insta[x]=1;
10    for(e=ah[x];e>-1;e=a[e].pre)
11    {
12        if (!dfn[y=a[e].y]) tarjan(y);
13        if (insta[y]) low[x]=min(low[x],low[y]);
14    }
15    if (low[x]==dfn[x])
16        for(gs++,t=0;t!=x;t=sta[ss-]) {group[sta[ss]]=gs; size[gs]++;}
17 }
18 void work()
19 {
20     ds=ss=gs=0;
21     int i; tr(i,1,n) if (!dfn[i]) tarjan(i);
22 }

```

7.6 最大流-ISAP

简版（无 BFS，递归，gap，cur）， $O(V^2 * E)$

```

1 struct edge{int x,y,c,f,pre;} a[2*maxm];
2 int n,mm,m,last[maxn],d[maxn],gap[maxn],cur[maxn],ans;
3 void newedge(int x,int y,int c,int f)
4 {
5     m++;
6     a[m].x=x; a[m].y=y; a[m].c=c; a[m].f=f;
7     a[m].pre=last[x]; last[x]=m;
8 }

```

```

9 void init()
10 {
11     int i,x,y,c;
12     m=-1;
13     memset(last,-1,sizeof(last));
14     tr(i,1,mm)
15     {
16         x=read(); y=read(); c=read();
17         newedge(x,y,c,0);
18         newedge(y,x,c,c);
19     }
20     tr(i,1,n) cur[i]=last[i];
21     memset(d,0,sizeof(d));
22     memset(gap,0,sizeof(gap));
23     gap[0]=n;
24     ans=0;
25 }
26 int sap(int x,int flow)
27 {
28     int e,t;
29     if (x==n) return flow;
30     for (e=cur[x];e!=-1;e=a[e].pre)
31         if (a[e].f<a[e].c && d[a[e].y]+1==d[x])
32         {
33             cur[x]=e;
34             if (t=sap(a[e].y,min(flow,a[e].c-a[e].f)))
35             {
36                 a[e].f+=t; a[e^1].f-=t; return t;
37             }
38         }
39     if (--gap[d[x]]==0) d[n]=n;
40     d[x]=n;
41     for (e=last[x];e!=-1;e=a[e].pre)
42         if (a[e].f<a[e].c) d[x]=min(d[x],d[a[e].y]+1);
43     cur[x]=last[x];
44     ++gap[d[x]];
45     return 0;
46 }
47 int work()
48 {
49     while (d[n]<n) ans+=sap(1,oo);
50 }

```

完全版（有 BFS，非递归，gap，cur）， $O(V^2 * E)$

```

1 int n,mm,m,ans,last[maxn],cur[maxn],pre[maxn],d[maxn],gap[maxn],b[maxn];
2 bool p[maxn];
3 struct edge{int x,y,c,f,pre;} a[2*maxm];
4 void newedge(int x,int y,int c,int f)
5 {
6     m++;
7     a[m].x=x; a[m].y=y; a[m].c=c; a[m].f=f;
8     a[m].pre=last[x]; last[x]=m;
9 }
10 void init()
11 {

```

```

12 int i,x,y,c;
13 m=-1;
14 memset(last,-1,sizeof(last));
15 tr(i,1,mm)
16 {
17     x=read(); y=read(); c=read();
18     newedge(x,y,c,0);
19     newedge(y,x,c,c);
20 }
21 }
22 int aug()
23 {
24     int x,flow=a[cur[1]].c-a[cur[1]].f;
25     for (x=pre[n];x>1;x=pre[x]) flow=min(flow,a[cur[x]].c-a[cur[x]].f);
26     return flow;
27 }
28 void bfs()
29 {
30     int h,t,e;
31     memset(p,0,sizeof(p));
32     b[1]=n; p[n]=1;
33     h=0; t=1;
34     while (h<t)
35     {
36         h++;
37         for (e=last[b[h]];e!=-1;e=a[e].pre)
38             if (a[e].c==a[e].f && !p[a[e].y])
39             {
40                 b[++t]=a[e].y;
41                 p[a[e].y]=1;
42                 d[a[e].y]=d[a[e].x]+1;
43             }
44     }
45 }
46 void sap()
47 {
48     int x,e,flow;
49     memset(d,0,sizeof(d));
50     memset(gap,0,sizeof(gap));
51     bfs();
52     tr(x,1,n) gap[d[x]]++;
53     ans=0;
54     tr(x,1,n) cur[x]=last[x];
55     x=1; pre[1]=1;
56     while (d[1]<n)
57     {
58         for (e=cur[x];e!=-1;e=a[e].pre)
59             if (d[x]==d[a[e].y]+1 && a[e].f<a[e].c)
60             {
61                 cur[x]=e;
62                 pre[a[e].y]=x;
63                 x=a[e].y;
64                 break;
65             }
66     }
67     if (e==1)
68     {

```

```

68         if (!(--gap[d[x]])) return;
69         cur[x]=last[x];
70         d[x]=n;
71         for (e=last[x];e!=-1;e=a[e].pre)
72             if (a[e].f<a[e].c) d[x]=min(d[x],d[a[e].y]+1);
73         gap[d[x]]++;
74         x=pre[x];
75     }
76     if (x==n){
77         flow=aug();
78         for (x=pre[x];x>1;x=pre[x])
79         {
80             a[cur[x]].f+=flow; a[cur[x]^1].f-=flow;
81         }
82         a[cur[x]].f+=flow; a[cur[x]^1].f-=flow;
83         ans+=flow;
84         x=1;
85     }
86 }
87 }

```

7.7 最小生成树-Prim

不加堆, $O(V + E)$

```

1 struct edge{int x,y,d,pre;} a[maxm];
2 int n,m,ah[maxn],d[maxn];
3 bool p[maxn];
4 void prim()
5 {
6     int i,j,x,y,e,ans=0;
7     memset(d,0x7f,sizeof(d)); d[1]=0;
8     memset(p,0,sizeof(p));
9     tr(i,1,n)
10     {
11         x=0;
12         tr(j,1,n) if (!p[j]&&d[j]<d[x]) x=j;
13         ans+=d[x];
14         p[x]=1;
15         for (e=ah[x];e>-1;e=a[e].pre)
16             if (!p[y=a[e].y]) d[y]=min(d[y],a[e].d);
17     }
18     printf("%d\n",ans);
19 }

```

加堆, $O(V + E)$

```

1 struct edge{int x,y,d,pre;} a[maxm];
2 typedef pair<int,int> pa;
3 priority_queue<pa,vector<pa>,greater<pa> >d;
4 int n,m,ah[maxn];
5 bool p[maxn];
6 void prim()
7 {
8     int i,x,y,e,ans=0;

```



```

9   pa t;
10  while (!d.empty()) d.pop();
11  d.push(make_pair(0,1));
12  memset(p,0,sizeof(p));
13  tr(i,1,n)
14  {
15      while (!d.empty() && p[d.top().second]) d.pop();
16      t=d.top();
17      ans+=t.first;
18      p[x=t.second]=1;
19      for(e=ah[x]; e>-1; e=a[e].pre)
20          if (!p[y=a[e].y]) d.push(make_pair(a[e].d,y));
21  }
22  printf("%d\n",ans);
23 }

```

7.8 最小生成树-Kruskal

$O(E \log E + E)$

```

1  //a从1开始!
2  struct edge{int x,y,d;} a[maxm];
3  bool cmp(edge a,edge b){return a.d<b.d;}
4  int n,i,j,m,fa[maxn];
5  int gfa(int x){return x==fa[x]?x:fa[x]=gfa(fa[x]);}
6  void kruskal()
7  {
8      int ans,fx,fy;
9      sort(a+1,a+m+1,cmp);
10     tr(i,1,n) fa[i]=i;
11     ans=0;
12     tr(i,1,m)
13         if ((fx=gfa(a[i].x))!=(fy=gfa(a[i].y)))
14         {
15             fa[fx]=fy;
16             ans+=a[i].d;
17         }
18     printf("%d\n",ans);
19 }

```

7.9 树的直径-BFS

$O(N)$

```

1  struct edge{int x,y,d,pre;} a[2*maxn];
2  int n,m,ah[maxn],d0[maxn],d1[maxn],b[maxn];
3  bool p[maxn];
4  void bfs(int root,int *d)
5  {
6      int h,t,e,y;
7      memset(p,0,sizeof(p));
8      h=0; t=1;
9      b[1]=root;
10     p[root]=1;

```

```

11    while (h<t)
12    {
13        h++;
14        for (e=ah[b[h]]; e>-1; e=a[e].pre)
15            if (!p[y=a[e].y])
16            {
17                b[++t]=y;
18                p[y]=1;
19                d[y]=d[a[e].x]+a[x].d;
20            }
21    }
22 }
23 void work()
24 {
25     int i,s1,s2;
26     memset(d0,0,sizeof(d0));
27     memset(d1,0,sizeof(d1));
28     bfs(1,d0); s1=1; tr(i,1,n) if (d0[i]>d0[s1]) s1=i;
29     bfs(s1,d1); s2=1; tr(i,1,n) if (d1[i]>d1[s2]) s2=i;
30     printf("%d %d %d\n",s1,s2,d1[s2]);
31 }

```

7.10 LCA-TarjanLCA

$O(N + Q)$

```

1  struct query{int x,y,pre,lca;} b[2*maxq];
2  struct edge{int x,y,pre,d;} a[2*maxn];
3  int n,q,am,bm,ah[maxn],bh[maxn],fa[maxn],dep[maxn];
4  bool p[maxn];
5  int gfa(int x){return fa[x]==x?x:fa[x]=gfa(fa[x]);}
6  void tarjan(int x,int depth)
7  {
8      int tmp,y;
9      p[x]=1;
10     dep[x]=depth;
11     for(tmp=ah[x]; tmp>-1; tmp=a[tmp].pre)
12         if (!p[y=a[tmp].y])
13         {
14             tarjan(y,depth+a[tmp].d);
15             fa[y]=x;
16         }
17     for(tmp=bh[x]; tmp>-1; tmp=b[tmp].pre)
18         if (p[y=b[tmp].y]) b[tmp].lca=b[tmp^1].lca=gfa(y);
19 }
20 void work()
21 {
22     memset(dep,0,sizeof(dep));
23     memset(p,0,sizeof(p));
24     tarjan(1,0);
25     int i; tr(i,0,q-1) writeln(dep[b[2*i].x]+dep[b[2*i].y]-2*dep[b[2*i].lca]);
26 }

```

8 数据结构

8.1 并查集

```
1 int gfa(int x){return(fa[x]==x?x:fa[x]=gfa(fa[x]));}
```

8.2 区间和 __ 单点修改区间查询-树状数组

$O(N\log N + Q\log N)$

```
1 int n,a[maxn],f[maxn];
2 char tc;
3 void modify(int x,int y)
4 {
5     while (x<=n) {f[x]+=y; x+=x&-x;}
6 }
7 int sum(int x)
8 {
9     int res=0;
10    while (x) {res+=f[x]; x-=x&-x;}
11    return res;
12 }
13 void work()
14 {
15     int q,i,tx,ty;
16     n=read(); q=read();
17     memset(f,0,sizeof(f));
18     tr(i,1,n) modify(i,a[i]=read());
19     tr(i,1,q)
20     {
21         tc=getchar(); tx=read(); ty=read();
22         if (tc=='M') {modify(tx,ty-a[tx]); a[tx]=ty;}
23         else writeln(sum(ty)-sum(tx-1));
24     }
25 }
```

8.3 区间和 __ 区间修改单点查询-树状数组

$O(N\log N + Q\log N)$

```
1 int n,i,f[maxn];
2 void modify(int x,int y)
3 {
4     while (x) {f[x]+=y; x-=x&-x;}
5 }
6 int sum(int x)
7 {
8     int res=0;
9     while (x<=n) {res+=f[x]; x+=x&-x;}
10    return res;
11 }
12 void work()
13 {
```

```
14     int q,i;
15     n=read(); q=read();
16     memset(f,0,sizeof(f));
17     tr(i,1,q)
18     {
19         tc=getchar();
20         if (tc=='M') {modify(read()-1,-1); modify(read(),1);}
21         else writeln(sum(read()));
22     }
23 }
```

8.4 区间和-线段树

$O(N\log N + Q\log N)$

```
1 struct node{int s,tag;} a[4*maxn];
2 int n;
3 void update(int t,int l,int r)
4 {
5     if (l==r)
6     {
7         a[t<<1].tag+=a[t].tag;
8         a[t<<1|1].tag+=a[t].tag;
9     }
10    a[t].s+=(int)(r-l+1)*a[t].tag;
11    a[t].tag=0;
12 }
13 void add(int t,int l,int r,int x,int y,int z)
14 {
15     if (x<=l&&r<=y) {a[t].tag+=z; return ;}
16    a[t].s+=(int)(min(r,y)-max(l,x)+1)*z;
17    update(t,l,r);
18    int mid=(l+r)>>1;
19    if (x<=mid) add(t<<1,l,mid,x,y,z);
20    if (y>mid) add(t<<1|1,mid+1,r,x,y,z);
21 }
22 int sum(int t,int l,int r,int x,int y)
23 {
24     int res=0;
25     update(t,l,r);
26     if (x<=l&&r<=y) return a[t].s;
27     int mid=(l+r)>>1;
28     if (x<=mid) res+=sum(t<<1,l,mid,x,y);
29     if (y>mid) res+=sum(t<<1|1,mid+1,r,x,y);
30     return res;
31 }
32 void work()
33 {
34     int q,i,tx,ty; char tc;
35     n=read(); q=read();
36     tr(i,1,n) add(1,1,n,i,i,read());
37     tr(i,1,q)
38     {
39         tc=getchar(); tx=read(); ty=read();
40         if (tc=='A') add(1,1,n,tx,ty,read());
```

```

41     else writeln(sum(1,1,n,tx,ty));
42 }
43 }

```

8.5 区间第 k 大 __ 无修改-主席树

$O(N\log N + Q\log N)$

```

1 struct node{int l,r,size;} a[maxm];
2 int n,q,m,num,b[maxn],dc[maxn],root[maxn];
3 int rdc(int x){return lower_bound(dc+1,dc+num+1,x)-dc;}
4 void init()
5 {
6     int i;
7     n=read(); q=read();
8     tr(i,1,n) b[i]=read();
9     memcpy(dc,b,(n+1)*sizeof(int));
10    sort(dc+1,dc+n+1);
11    num=unique(dc+1,dc+n+1)-(dc+1);
12 }
13 int insert(int tx,int l,int r,int x)
14 {
15     int t,mid=(l+r)>>1;
16     a[t=++m]=a[tx]; a[t].size++;
17     if (l==r) return t;
18     if (x<=mid) a[t].l=insert(a[tx].l,l,mid,x);
19     else a[t].r=insert(a[tx].r,mid+1,r,x);
20     return t;
21 }
22 int kth(int tx,int ty,int l,int r,int k)
23 {
24     int ds,mid=(l+r)>>1;
25     if (l==r) return l;
26     if (k<=(ds=a[a[ty].l].size-a[a[tx].l].size))
27         return kth(a[tx].l,a[ty].l,l,mid,k);
28     else return kth(a[tx].r,a[ty].r,mid+1,r,k-ds);
29 }
30 void work()
31 {
32     int i,x,y,z;
33     tr(i,1,n) root[i]=insert(root[i-1],1,num,rdc(b[i]));
34     tr(i,1,q)
35     {
36         x=read(); y=read(); z=read();
37         writeln(dc[kth(root[x-1],root[y],1,num,z)]);
38     }
39 }

```

8.6 RMQ-ST

$O(N\log N) O(1)$

```

1 int n,q,mn[maxn][maxln];
2 void init()

```

```

3 {
4     int i;
5     n=read(); q=read();
6     tr(i,1,n) mn[i][0]=read();
7 }
8 void st()
9 {
10    int i,j,ln;
11    ln=log(n)/log(2);
12    tr(i,1,ln) tr(j,1,n-(1<<i)+1)
13        mn[j][i]=min(mn[j][i-1],mn[j+(1<<(i-1))][i-1]);
14 }
15 void work()
16 {
17     int i,x,y,t;
18     st();
19     tr(i,1,q)
20     {
21         x=read(); y=read();
22         t=log(y-x+1)/log(2);
23         writeln(min(mn[x][t],mn[y-(1<<t)+1][t]));
24     }
25 }

```

9 其它