

hust-yc-template

yincircle

2019 年 9 月 26 日

目录

1 字符串	1	4 图论	20
1.1 最大最小表示法	1	4.1 k 短路	20
1.2 KMP	1	4.2 最大流	21
1.2.1 前缀函数 (每一个前缀的最长 border)	1	4.3 最小费用流	22
1.2.2 Z 函数 (每一个后缀和该字符串的 LCP 长度)	1	4.4 点分治	22
1.3 manacher	1	5 计算几何	23
1.4 AC 自动机	2	6 杂项	33
1.5 SAM	3	6.1 define	33
1.6 PAM	5	6.2 数位 dp	33
1.7 DA+LCA	6	6.3 fastio	33
1.8 HASH	6	6.4 date	35
2 数据结构	8	6.5 常用概念	35
2.1 01Trie	8	6.6 欧拉路径	35
2.2 Trie	8	6.7 映射	35
2.3 线段树	9	6.8 反演	35
2.4 主席树	9	6.9 弦图	36
2.5 HLD	10	6.10 五边形数	36
3 数学	12	6.11 pick 定理	36
3.1 矩阵	12	6.12 重心	36
3.2 快速乘	12	6.13 曼哈顿距离与切比雪夫距离	36
3.3 快速幂	13	6.14 第二类 Bernoulli number	36
3.4 筛	13	6.15 Fibonacci 数	36
3.4.1 线性筛素数	13	6.16 Catalan 数	36
3.4.2 线性筛欧拉函数	13	6.17 Lucas 定理	36
3.4.3 线性筛莫比乌斯函数	13	6.18 扩展 Lucas 定理	37
3.4.4 杜教筛	14	6.19 BEST theorem	37
3.5 素数测试	15	6.20 欧拉示性数定理	37
3.5.1 素数判断	15	6.21 最值反演 (MinMax 容斥)	37
3.5.2 Pollard-Rho	15	6.22 Polya 定理	37
3.6 BM 线性递推	16	6.23 Stirling 数	37
3.7 扩展欧几里德	16	6.24 常用排列组合公式	37
3.8 线性基	17	6.25 三角公式	37
3.9 exBSGS	18	6.26 积分表	38
3.10 中国剩余定理	18		
3.11 类欧几里德	19		
3.12 逆元	19		
3.13 组合数	19		
3.14 公式	19		
3.14.1 数论公式	19		
3.14.2 一些数论函数求和的例子	19		
3.14.3 莫比乌斯反演	20		
3.14.4 低阶等幂求和	20		
3.14.5 一些组合公式	20		
3.15 Theorem	20		

1 字符串

1.1 最大最小表示法

```

1  int get_posmin(char *s){
2      int len=strlen(s);
3      int i=0,j=1,k=0;
4      while(i<len&&j<len&&k<len){
5          int t=s[(i+k)%len]-s[(j+k)%len];
6          if(t==0)k++;
7          else {
8              if(t>0)
9                  i+=k+1;
10                 //max: j+=k+1;
11             else
12                 j+=k+1;
13                 //max: i+=k+1;
14             if(i==j)j++;
15             k=0;
16         }
17     }
18     return min(i,j);
19 }
```

1.2 KMP

1.2.1 前缀函数（每一个前缀的最长 border）

```

1  /*
2  length of min loop
3  if(len%(len-nex[len])==0) res=len/(len-nex[len]);
4  else res=1;
5  s->s[0..n-1]
6  */
7  char s[maxn];
8  int nex[maxn];
9  void get_next(char *s,int *nex,int len){
10     int i,j;
11     i=0;
12     j=nex[0]=-1;
13     while(i<len)
14         if(j==-1||s[i]==s[j]) nex[++i]=++j;
15         else j=nex[j];
16 }
17 int KMP(char *a,char *b,int lena,int lenb){
18     int i,j;
```

```

19     get_next(b,nex,lenb);
20     i=j=0;
21     while(i<lena){
22         if(j==-1||a[i]==b[j]){i++;j++;}
23         else j=nex[j];
24         if(j==lenb) break;//successful match
25     }
26     return j==-1?0:j;
27 }
```

1.2.2 Z 函数（每一个后缀和该字符串的 LCP 长度）

```

1  void get_z(int a[], char s[], int n) {
2      int l = 0, r = 0; a[0] = n;
3      for(int i=1;i<n;i++) {
4          a[i] = i > r ? 0 : min(r - i + 1, a[i - 1]);
5          while (i + a[i] < n && s[a[i]] == s[i + a[i]
6              ]) ++a[i];
7          if (i + a[i] - 1 > r) { l = i; r = i + a[i] -
8              1; }
9      }
```

1.3 manacher

```

1  int ma[maxn<<1|1];
2  char s[maxn];
3  void manacher(char s[]){
4      int n = strlen(s);
5      int id=0, ub=0;
6      for(int i=0; i<2*n-1; i++) {
7          int p=i/2, q=(i+1)/2;
8          int l = q<ub?min(ub-q, ma[id-i]):0;
9          while(p-l>=0 && q+l<n && s[p-l]==s[q+l]) l++;
10         if(ub < q+l) {
11             ub = q+l;
12             id = i*2;
13         }
14         ma[i] = l;
15     }
16     for(int i=0; i<2*n-1; i++)
17         ma[i] = ma[i]*2-(!i&1);
18 }
```

1.4 AC 自动机

```

1 // HDU 6138
2 //给定若干字典串。
3 // query:strx stry 求最长的p,p为strx、stry子串，且p为
  某字典串的前缀
4 struct Aho_Corasick_Automaton{
5     //basic
6     int nxt[maxn*10][26],fail[maxn*10];
7     int root,tot;
8     //special
9     int flag[maxn*10];
10    int len[maxn*10];
11    void clear(){
12        memset(nxt[0],0,sizeof nxt[0]);
13        root = tot=0;
14    }
15    int newnode(){
16        tot++;
17        memset(nxt[tot],0,sizeof nxt[tot]);
18        flag[tot] = len[tot]=0;
19        return tot;
20    }
21    void insert(char *s ){
22        int now = root;
23        while (*s){
24            int id = *s-'a';
25            if(!nxt[now][id])nxt[now][id] = newnode()
                ;
26            len[nxt[now][id]] = len[now]+1;
27            now = nxt[now][id];
28        }
29    }
30    void insert(string str){
31        int now = root;
32        for (int i=0;i<str.size();i++){
33            int id = str[i]-'a';
34            if(!nxt[now][id])nxt[now][id] = newnode()
                ;
35            len[nxt[now][id]] = len[now]+1;
36            now = nxt[now][id];
37        }
38    }
39    void build(){
40        fail[root] = root;
41        queue<int>Q;Q.push(root);
42        while (!Q.empty()){

```

```

43            int head = Q.front();Q.pop();
44            for (int i=0;i<26;i++){
45                if(!nxt[head][i])continue;
46                int temp = nxt[head][i];
47                fail[temp] = fail[head];
48                while (fail[temp]&&!nxt[fail[temp]][i]
                    ){
49                    fail[temp] = fail[fail[temp]];
50                }
51                if(head&&nxt[fail[temp]][i])fail[temp]
                    = nxt[fail[temp]][i];
52                Q.push(temp);
53            }
54        }
55    }
56    void search(string str,int QID);
57    int query(string str,int QID);
58 }acam;
59 void Aho_Corasick_Automaton::search(string str,int
    QID) {
60     int now = root;
61     for (int i=0;i<str.size();i++){
62         int id = str[i]-'a';
63         now = nxt[now][id];int temp = now;
64         while (temp!=root&&flag[temp]!=QID){
65             flag[temp] = QID;
66             temp = fail[temp];
67         }
68     }
69 }
70 int Aho_Corasick_Automaton::query(string str, int
    QID) {
71     int ans =0;int now = root;
72     for (int i=0;i<str.size();i++){
73         int id = str[i]-'a';
74         now = nxt[now][id];
75         int temp = now;
76         while (temp!=root){
77             if(flag[temp]==QID){
78                 ans = max(ans,len[temp]);
79                 break;
80             }
81             temp = fail[temp];
82         }
83     }
84     return ans;
85 }
86 string a[maxn];

```

```

87     int m,n,qid;
88     int main(){
89         int T;cin>>T;
90         while (T--){
91             acam.clear();cin>>n;
92             for (int i=1;i<=n;i++){
93                 cin>>a[i];
94                 acam.insert(a[i]);
95             }
96             acam.build();cin>>m;
97             for (int i=1;i<=m;i++){
98                 int x,y;cin>>x>>y;
99                 qid++;
100                acam.search(a[x],qid);
101                int ans = acam.query(a[y],qid);
102                cout<<ans<<endl;
103            }
104        }
105        return 0;
106    }

```

1.5 SAM

```

1     struct SAM{
2         int last,cnt,nxt[maxn*2][27],fa[maxn*2],l[maxn*2];
3         void init(){
4             last = cnt=1;
5             memset(nxt[1],0,sizeof nxt[1]);
6             fa[1]=0;ans=0;l[1]=0;
7         }
8         int inline newnode(){
9             ++cnt;
10            memset(nxt[cnt],0,sizeof nxt[cnt]);
11            fa[cnt]=l[cnt]=0;
12            return cnt;
13        }
14        void add(int c){
15            int p = last;
16            int np = newnode();
17            last = np;
18            l[np] = l[p]+1;
19            while (p&&!nxt[p][c]){
20                nxt[p][c]=np;
21                p = fa[p];
22            }
23            if (!p){

```

```

24                fa[np]=1;
25            }else{
26                int q = nxt[p][c];
27                if (l[q]==l[p]+1){
28                    fa[np] = q;
29                }else{
30                    int nq = newnode();
31                    memcpy(nxt[nq],nxt[q],sizeof nxt[q]);
32                    fa[nq] = fa[q];
33                    l[nq] = l[p]+1;
34                    fa[np]=fa[q]=nq;
35                    while (nxt[p][c]==q){
36                        nxt[p][c]=nq;
37                        p=fa[p];
38                    }
39                }
40            }
41            ans+=l[last]-l[fa[last]];
42        }
43    }sam;
44
45    //SPOJ substring
46    // calc ans_i=长度=i的所有子串，出现次数最多的一种出现了多少次。
47    #include<bits/stdc++.h>
48    #define RIGHT
49    //RIGHT: parent树的dfs序上主席树，求每个点的Right集合
50    using namespace std;
51    const int maxn = 25e4+100;
52    struct Node{int L,R,val;}Tree[maxn*40];
53    #ifdef RIGHT
54    struct Chairman_Tree{
55        int cnt = 0;
56        int root[maxn*2];
57        void init(){
58            memset(root,0,sizeof root);
59            cnt =0;
60        }
61        /* 建T0空树 */
62        int buildT0(int l, int r){
63            int k = cnt++;
64            Tree[k].val =0;
65            if (l==r) return k;
66            int mid = l+r >>1;
67            Tree[k].L = buildT0(l, mid);Tree[k].R =
68                buildT0(mid + 1, r);
69            return k;

```

```

70  /* 上一个版本节点P, 【ppos】 +=del 返回新版本节点*/ 112
71  int update (int P,int l,int r,int ppos,int del){ 113
72      assert(cnt < maxn*50); 114
73      int k = cnt++; 115
74      Tree[k].val = Tree[P].val +del; 116
75      if (l==r) return k; 117
76      int mid = l+r >>1; 118
77      if (ppos<=mid){ 119
78          Tree[k].L = update(Tree[P].L,l,mid,ppos, 120
79                          del); 121
79          Tree[k].R = Tree[P].R; 122
80      }else{ 123
81          Tree[k].L = Tree[P].L; 124
82          Tree[k].R = update(Tree[P].R,mid+1,r,ppos 125
83                          ,del); 126
84      } 127
85      return k; 128
86  } 129
87  int query(int PL,int PR,int l,int r,int L,int R) 130
88  { 131
89      if (l>R || L>r)return 0; 132
90      if (L <= l && r <= R)return Tree[PR].val - 133
91          Tree[PL].val; 134
92      int mid = l + r >> 1; 135
93      return query(Tree[PL].L,Tree[PR].L,l,mid,L,R) 136
94          + query(Tree[PL].R,Tree[PR].R,mid+1,r,L, 137
95          R); 138
96  } 139
97  }tree; 140
98  #endif 141
99  char s[maxn];int n,ans[maxn]; 142
100  /*注意需要按l将节点基数排序来拓扑更新parent树*/ 143
101  struct Suffix_Automaton{ 144
102      //basic 145
103      int nxt[maxn*2][26],fa[maxn*2],l[maxn*2]; 146
104      int last,cnt; 147
105      //extension 148
106      int cntA[maxn*2],A[maxn*2];/*辅助拓扑更新*/ 149
107      int num[maxn*2];/*每个节点代表的所有串的出现次数*/ 150
108      #ifdef RIGHT 151
109      vector<int> E[maxn*2]; 152
110      int dfs1[maxn*2],dfsr[maxn*2],dfn; 153
111      int pos[maxn*2]; 154
112      int end_pos[maxn*2];//1基 155
113      #endif 156
114      Suffix_Automaton(){ clear(); } 157
115      void clear(){ 158
116          last =cnt=1; 159
117          fa[1]=l[1]=0; 160
118          memset(nxt[1],0,sizeof nxt[1]); 161
119      } 162
120      void init(char *s){ 163
121          while (*s){ 164
122              add(*s-'a');s++; 165
123          } 166
124      } 167
125      void add(int c){ 168
126          int p = last; 169
127          int np = ++cnt; 170
128          memset(nxt[cnt],0,sizeof nxt[cnt]); 171
129          l[np] = l[p]+1;last = np; 172
130          while (p&&!nxt[p][c])nxt[p][c] = np,p = fa[p] 173
131              l; 174
132          if (!p)fa[np]=1; 175
133          else{ 176
134              int q = nxt[p][c]; 177
135              if (l[q]==l[p]+1)fa[np] =q; 178
136              else{ 179
137                  int nq = ++ cnt; 180
138                  l[nq] = l[p]+1; 181
139                  memcpy(nxt[nq],nxt[q],sizeof (nxt[q])) 182
140                      ; 183
141                  fa[nq] =fa[q];fa[np] = fa[q] =nq; 184
142                  while (nxt[p][c]==q)nxt[p][c] =nq,p = 185
143                      fa[p]; 186
144              } 187
145          } 188
146      } 189
147      void build(){ 190
148          memset(cntA,0,sizeof cntA); 191
149          memset(num,0,sizeof num); 192
150          for (int i=1;i<=cnt;i++)cntA[l[i]]++; 193
151          for (int i=1;i<=cnt;i++)cntA[i]+=cntA[i-1]; 194
152          for (int i=cnt;i>=1;i--)A[cntA[l[i]]--] =i; 195
153          /*更行主串节点*/ 196
154          int temp=1; 197
155          for (int i=0;i<n;i++){ 198
156              num[temp = nxt[temp][s[i]-'a']] =1; 199
157          } 200
158          /*拓扑更新*/ 201
159          for (int i=cnt;i>=1;i--){ 202
160              //basic 203
161              int x = A[i]; 204
162              num[fa[x]]+=num[x]; 205
163              //special 206
164              ans[l[x]] = max(ans[l[x]],num[x]); 207
165          } 208
166      } 209
167  } 210

```

```

157     }
158     //special
159     for (int i=l[last];i>1;i--){
160         ans[i-1] = max(ans[i-1],ans[i]);
161     }
162 }
163
164 #ifdef RIGHT
165 int get_right_between(int u,int l,int r){
166     return tree.query(tree.root[dfsl[u] - 1],tree
167         .root[dfsr[u]],1,:n,l,r);
168 }
169 void dfs(int u){
170     dfsl[u] = ++ dfn;
171     pos[dfn] = u;
172     for (int v : E[u]){
173         dfs(v);
174     }
175     dfsr[u] = dfn;
176 }
177 void extract_right(){
178     int temp = 1;
179     for (int i=0;i<n;i++){
180         temp = nxt[temp][s[i] - 'a'];
181         end_pos[temp] = i+1;
182     }
183     for (int i=2;i<=cnt;i++){
184         E[fa[i]].push_back(i);
185     }
186     dfn = 0;
187     dfs(1);
188     tree.root[0] = tree.buildT0(1,n);
189     for (int i=1;i<=cnt;i++){
190         int u = pos[i];
191         if (end_pos[u]){
192             int idx = end_pos[u];
193             tree.root[i] = tree.update(tree.root[i
194                 -1],1,n,idx,1);
195         }else{
196             tree.root[i] = tree.root[i-1];
197         }
198     }
199 }
200 #endif
201 void debug(){
202     for (int i=cnt;i>=1;i--){
203         printf("num[%d]=%d l[%d]=%d fa[%d]=%d\n",
204             i,num[i],i,l[i],i,fa[i]);

```

```

202     }
203 }
204 }sam;
205 int main(){
206     scanf("%s",s);
207     /* calc n must before sam.init()*/
208     n = strlen(s);
209     sam.init(s);
210     sam.build();
211     for (int i=1;i<=n;i++){
212         printf("%d\n",ans[i]);
213     }
214     return 0;
215 }

```

1.6 PAM

```

1 struct Palindromic_AutoMaton{
2     //basic
3     int s[maxn],now;
4     int nxt[maxn][26],fail[maxn],l[maxn],last,tot;
5     //extension
6     int num[maxn];
7     void clear(){
8         //1节点: 奇数长度root 0节点: 偶数长度root
9         s[0]=l[1]=-1;
10        fail[0]=tot=now=1;
11        last=l[0]=0;
12        memset(nxt[0],0,sizeof(nxt[0]));
13        memset(nxt[1],0,sizeof(nxt[1]));
14    }
15    Palindromic_AutoMaton(){clear();}
16    int newnode(int x){
17        tot++;
18        memset(nxt[tot],0,sizeof(nxt[tot]));
19        fail[tot]=num[tot]=0;
20        l[tot]=x;
21        return tot;
22    }
23    int get_fail(int x){
24        while(s[now-l[x]-2]!=s[now-1])x=fail[x];
25        return x;
26    }
27    void add(int ch){
28        s[now++]=ch;
29        int cur=get_fail(last);
30        if(!nxt[cur][ch]){

```

```

31         int tt=newnode(1[cur]+2);
32         fail[tt]=nxt[get_fail(fail[cur])][ch];
33         nxt[cur][ch]=tt;
34     }
35     last=nxt[cur][ch];num[last]++;
36 }
37 void build(){
38     for(int i=tot;i>=2;i--){
39         num[fail[i]]+=num[i];
40     }
41     num[0]=num[1]=0;
42 }
43 void init(char* ss){
44     while(*ss){
45         add(*ss-'a');ss++;
46     }
47 }
48 void init(string str){
49     for(int i=0;i<(int)str.size();i++){
50         add(str[i]-'a');
51     }
52 }
53 }pam;

```

1.7 DA+LCA

```

1  int t1[maxn],t2[maxn],c[maxn];
2
3  bool cmp(int *r,int a,int b,int l){
4      return r[a]==r[b]&&r[a+l]==r[b+l];
5  }
6
7  void da(int str[],int sa[],int rank[],int height[],
8      int n,int m){
9      n++;
10     int i,j,p,*x=t1,*y=t2;
11     for(i=0;i<m;i++)c[i]=0;
12     for(i=0;i<n;i++)c[x[i]=str[i]]++;
13     for(i=1;i<m;i++)c[i]+=c[i-1];
14     for(i=n-1;i>=0;i--)sa[--c[x[i]]]=i;
15     for(j=1;j<=n;j<=1){
16         p=0;
17         for(i=n-j;i<n;i++)y[p++]=i;
18         for(i=0;i<n;i++)if(sa[i]>=j)y[p++]=sa[i]-j;
19         for(i=0;i<m;i++)c[i]=0;
20         for(i=0;i<n;i++)c[x[y[i]]]++;
21         for(i=1;i<m;i++)c[i]+=c[i-1];

```

```

21         for(i=n-1;i>=0;i--)sa[--c[x[y[i]]]]=y[i];
22         swap(x,y);
23         p=1;
24         x[sa[0]]=0;
25         for(i=1;i<n;i++)
26             x[sa[i]]=cmp(y,sa[i-1],sa[i],j)?p-1:p++;
27         if(p>=n)break;
28         m=p;
29     }
30     int k=0;
31     n--;
32     for(i=0;i<=n;i++)rank[sa[i]]=i;
33     for(i=0;i<n;i++){
34         if(k)k--;
35         j=sa[rank[i]-1];
36         while(str[i+k]==str[j+k])k++;
37         height[rank[i]]=k;
38     }
39 }
40
41 int rnk[maxn],height[maxn],r[maxn],sa[maxn];
42 int rmq[maxn];
43
44 int n,minnum[maxn][20];
45 void RMQ(){
46     int i,j;
47     int m=(int)(log(n*1.0)/log(2.0));
48     for(i=1;i<=n;i++)
49         minnum[i][0]=height[i];
50     for(j=1;j<=m;j++)
51         for(i=1;i+(1<<j)-1<=n;i++)
52             minnum[i][j]=min(minnum[i][j-1],minnum[i
53                 +(1<<(j-1))][j-1]);
54 }
55 int askrmq(int a,int b){
56     int k=(int)(log(b-a+1.0)/log(2.0));
57     return min(minnum[a][k],minnum[b-(1<<k)+1][k]);
58 }
59 int lcp(int a,int b){
60     a=rnk[a],b=rnk[b];
61     if(a>b)
62         swap(a,b);
63     return askrmq(a+1,b);

```

1.8 HASH


```

1  #include<bits/stdc++.h>
2  using namespace std;
3  typedef unsigned long long ULL;
4  const int maxn = 305*305;
5  /* 字符集大小 */
6  const int sigma = maxn;
7  /* hash次数 */
8  const int HASH_CNT = 2;
9  int n;
10 int s[maxn];
11 /* char* 1-bas
12 * sum[i] = s[i]+s[i-1]*Seed+s[i-2]*Seed^2+...+s[1]*
    Seed^(i-1)*/
13 ULL Prime_Pool[] = {1998585857ul,2333333333ul};
14 ULL Seed_Pool[]={911,146527,19260817,91815541};
15 ULL Mod_Pool
    []={29123,998244353,1000000009,4294967291ull};
16 struct Hash_1D{
17     ULL Seed,Mod;
18     ULL bas[maxn];ULL sum[maxn];
19     int perm[sigma];
20     void init(int seedIndex,int modIndex){
21         Seed = Seed_Pool[seedIndex];
22         Mod = Mod_Pool[modIndex];
23         bas[0]=1;
24         for (int i=1;i<=n;i++){
25             bas[i] = bas[i-1]*Seed%Mod;
26         }
27         for (int i=1;i<=n;i++){
28             sum[i] = (sum[i-1]*Seed%Mod+s[i])%Mod;
29         }
30     }
31     /*random_shuffle 离散化id, 防止kill_hash*/
32     void indexInit(int seedIndex,int modIndex){
33         for (int i=1;i<=n;i++){
34             perm[i]=i;
35         }
36         random_shuffle(perm+1,perm+1+sigma);
37         Seed = Seed_Pool[seedIndex];
38         Mod = Mod_Pool[modIndex];
39         bas[0]=1;
40         for (int i=1;i<=n;i++){
41             bas[i] = bas[i-1]*Seed%Mod;
42         }
43         for (int i=1;i<=n;i++){
44             sum[i] = (sum[i-1]*Seed%Mod+perm[s[i]])%
                Mod;
45         }
46     }
47     ULL getHash(int l,int r){
48         return (sum[r]-sum[l-1]*bas[r-l+1]%Mod+Mod)%
                Mod;
49     }
50 }hasher[HASH_CNT];
51 map<pair<pair<ULL,ULL>,int>,int>veid;int vecnt;
52 map<string,int>id;int idcnt;
53 vector<int> pos[maxn];
54 string a[maxn];
55 int sumL[maxn];
56 int main(){
57     cin>>n;
58     for (int i=1;i<=n;i++){
59         cin>>a[i];
60         if (!id[a[i]])id[a[i]] = ++idcnt;
61         s[i] = id[a[i]];
62         sumL[i] = sumL[i-1]+a[i].size();
63     }
64     for (int i=0;i<HASH_CNT;i++){
65         hasher[i].indexInit(i,i);
66     }
67     int ans = sumL[n]+n-1;
68     for (int i=1;i<=n;i++){
69         for (int j=1;j<=n;j++){
70             ULL hash1 = hasher[0].getHash(i,j);
71             ULL hash2 = hasher[1].getHash(i,j);
72             int len = j-i+1;
73             pair<pair<ULL,ULL>,int> x = {{hash1,hash2
                },len};
74             if (veid[x]==0)veid[x] = ++vecnt;
75             pos[veid[x]].push_back(i);
76         }
77     }
78     int maxDelta =0;
79     for (auto x:veid){
80         int len = x.first.second;
81         int i = x.second;
82         sort(pos[i].begin(),pos[i].end());
83         int num =0;
84         for (int j=0,last = -maxn;j<pos[i].size();j
            ++){
85             if (pos[i][j]>=last+len){
86                 last = pos[i][j];
87                 num++;
88             }
89         }
90         if (num==1)continue;

```

```

91     int cost1 = sumL[pos[i][0]+len-1]-sumL[pos[i]
        ][0]-1]+len-1;
92     int cost2 = len;
93     int tempDelta = (cost1-cost2)*num;
94     maxDelta = max(maxDelta,tempDelta);
95 }
96 cout<<ans-maxDelta<<endl;
97 return 0;
98 }

```

2 数据结构

2.1 01Trie

```

1  /*
2  数组大小(x+1)*MAX:插入的值的最大值<2^x<MAX
3  Trie.Insert(1,x,v);
4  Trie.Delete(1,x,v);
5  Trie.query(1,x,v);
6  Trie.clear(1,x);
7  */
8  struct Trie
9  {
10     int cnt[32*MAX],val[32*MAX];
11     void Insert(int x,int pos,int v)
12     {
13         if(pos<0)
14         {
15             cnt[x]++;
16             val[x]=v;
17             return;
18         }
19         Insert((x<<1)|((v>>pos)&1),pos-1,v);
20         cnt[x]=cnt[x<<1]+cnt[x<<1|1];
21     }
22     void Delete(int x,int pos,int v)
23     {
24         if(pos<0)
25         {
26             cnt[x]--;
27             return;
28         }
29         Delete((x<<1)|((v>>pos)&1),pos-1,v);
30         cnt[x]=cnt[x<<1]+cnt[x<<1|1];
31     }
32     void clear(int x,int pos)
33     {

```

```

34     cnt[x]=0;
35     val[x]=0;
36     if(pos<0) return;
37     clear(x<<1,pos-1);
38     clear(x<<1|1,pos-1);
39 }
40 int query(int x,int pos,int v)//查询与v异或的最大
    值 并返回
41 {
42     if(pos<0) return val[x];
43     int temp=(v>>pos)&1;
44     temp|=x<<1;
45     if(cnt[temp^1]) return query(temp^1,pos-1,v);
46     return query(temp,pos-1,v);
47 }
48 }tr;

```

2.2 Trie

```

1  struct Trie
2  {
3      #define type int
4      struct trie
5      {
6          int v;
7          trie *next[26];
8          trie()
9          {
10              v=0;
11              for(int i=0;i<26;i++) next[i]=NULL;
12          }
13      }*root;
14      void insert(trie *p,char *s)
15      {
16          int i=0,t;
17          while(s[i])
18          {
19              t=s[i]-'a';
20              if(p->next[t]==NULL) p->next[t]=new trie;
21              p=p->next[t];
22              p->v++;//按情况改
23              i++;
24          }
25      }
26      int find(trie *p,char *s)
27      {
28          int i=0,t;

```

```

29     while(s[i])
30     {
31         t=s[i]-'a';
32         if(p->next[t]==NULL) return 0;
33         p=p->next[t];
34         i++;
35     }
36     return p->v;//按情况改
37 }
38 //删除前缀为s的字符串
39 void del(char *s)
40 {
41     int i=0,t,temp;
42     trie *p,*pre;
43     pre=p=root;
44     while(s[i])
45     {
46         t=s[i]-'a';
47         if(p->next[t]==NULL) return;
48         if(!s[i+1])
49         {
50             temp=p->next[t]->v;
51             p->next[t]=NULL;
52             break;
53         }
54         pre=p;
55         p=p->next[t];
56         i++;
57     }
58     i=0;
59     p=root;
60     while(s[i])
61     {
62         t=s[i]-'a';
63         if(p->next[t]==NULL) return;
64         p=p->next[t];
65         p->v-=temp;
66         i++;
67     }
68 }
69 #undef type
70 }tr;

```

2.3 线段树

```

1 struct Seg_Tree{
2     int val[maxn<<2],lazy[maxn<<2];

```

```

3 void init(){
4     memset(val,0,sizeof(val));
5     memset(lazy,0,sizeof(val));
6 }
7 inline void up(int rt){
8     val[rt]=val[rt<<1]+val[rt<<1|1];
9 }
10 inline void down(int rt,int l,int r){
11     int mid=l+r>>1;
12     if(lazy[rt]){
13         lazy[rt<<1]+=lazy[rt];
14         lazy[rt<<1|1]+=lazy[rt];
15         val[rt<<1]+=lazy[rt]*(mid-l+1);
16         val[rt<<1|1]+=lazy[rt]*(r-mid);
17         lazy[rt]=0;
18     }
19 }
20 void build(int rt,int l,int r){
21     if(l==r){
22         val[rt]=a[l];
23         return ;
24     }
25     int mid=l+r>>1;
26     build(rt<<1,l,mid);
27     build(rt<<1|1,mid+1,r);
28     up(rt);
29 }
30 void update(int rt,int l,int r,int L,int R,int
    del){
31     if(l>R||r<L)return ;
32     if(L<=l&&r<=R){
33         val[rt]+=del*(r-l+1);
34         lazy[rt]+=del;
35         return ;
36     }
37     int mid=l+r>>1;
38     down(rt,l,r);
39     update(rt<<1,l,mid,L,R,del);
40     update(rt<<1|1,mid+1,r,L,R,del);
41     up(rt);
42 }
43 void add(int rt,int l,int r,int L,int del){
44     if(l==r){
45         val[rt]+=del;
46         return ;
47     }
48     down(rt,l,r);
49     int mid=l+r>>1;

```

```

50     if(L<=mid)add(rt<<1,l,mid,L,del);
51     else add(rt<<1|1,mid+1,r,L,del);
52     up(rt);
53 }
54 int query_sum(int rt,int l,int r,int L,int R){
55     if(L<=l&&r<=R)return val[rt];
56     int mid=l+r>>1;
57     down(rt,l,r);
58     int res=0;
59     if(L<=mid)res+=query_sum(rt<<1,l,mid,L,R);
60     if(R>mid)res+=query_sum(rt<<1|1,mid+1,r,L,R);
61     return res;
62 }
63 }seg;

```

2.4 主席树

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  const int maxn=1e5+100;
4  int a[maxn];int rk[maxn];int pos[maxn];
5  int root[maxn];int cnt,m,n,T;
6  struct Chairman_Tree{
7      struct Node{int L,R,val;}tree[maxn*500];
8      void init(){
9          memset(root,0,sizeof root);
10         cnt =0;
11     }
12     /* 建T0空树 */
13     int buildT0(int l, int r){
14         int k = cnt++;
15         tree[k].val =0;
16         if (l==r) return k;
17         int mid = l+r >>1;
18         tree[k].L = buildT0(l, mid);tree[k].R =
19             buildT0(mid + 1, r);
20         return k;
21     }
22     /* 上一个版本节点P, 【ppos】 +=del 返回新版本节点*/
23     int update (int P,int l,int r,int ppos,int del){
24         int k = cnt++;
25         tree[k].val = tree[P].val +del;
26         if (l==r) return k;
27         int mid = l+r >>1;
28         if (ppos<=mid){
29             tree[k].L = update(tree[P].L,l,mid,ppos,
30                 del);

```

```

29         tree[k].R = tree[P].R;
30     }else{
31         tree[k].L = tree[P].L;
32         tree[k].R = update(tree[P].R,mid+1,r,ppos
33             ,del);
34     }
35     return k;
36 }
37 int query_kth(int lt,int rt,int l,int r,int k){
38     if (l==r) return a[rk[l]];
39     int mid = l+r >>1;
40     if (tree[tree[rt].L].val-tree[tree[lt].L].val
41         >=k) return query_kth(tree[lt].L,tree[rt
42             ].L,l,mid,k);
43     else return query_kth(tree[lt].R,tree[rt].R,
44         mid+1,r,k+tree[tree[lt].L].val-tree[tree[
45             rt].L].val);
46 }
47 }tree;
48 bool cmp(int x,int y){return a[x]<a[y];}
49 int main() {
50     scanf("%d", &T);
51     while (T--) {
52         scanf("%d%d",&n,&m);
53         for (int i=1;i<=n;i++){
54             scanf("%d",&a[i]);
55             rk[i]=i;
56         }
57         tree.init();
58         sort(rk+1,rk+1+n,cmp);
59         for (int i1=1;i1<=n;i1++){
60             pos[rk[i1]] =i1;
61         }
62         root[0] = tree.buildT0(1, n);
63         for (int i1=1;i1<=n;i1++){
64             root[i1] = tree.update(root[i1-1],1,n,pos
65                 [i1],1);
66         }
67         while (m--){
68             int l,r,k;scanf("%d%d%d",&l,&r,&k);
69             printf("%d\n",tree.query_kth(root[l-1],
70                 root[r],1,n,k));
71         }
72     }
73     return 0;
74 }

```

2.5 HLD

```

1  /*
2  size[] 数组，以x为根的子树节点个数
3  top[] 数组，当前节点的所在链的顶端节点
4  son[] 数组，重儿子
5  deep[] 数组，当前节点的深度
6  fa[] 数组，当前节点的父亲
7  idx[] 数组，树中每个节点剖分后的新编号
8  rnk[] 数组，idx的逆，表示线段上中当前位置表示哪个节点
9  */
10 struct HLD
11 {
12     #define type int
13     struct edge{int a,b;type v;edge(int _a,int _b,
14         type _v=0):a(_a),b(_b),v(_v){}};
15     struct node{int to;type w;node(){}node(int _to,
16         type _w):to(_to),w(_w){}};
17     vector<int> mp[MAX];
18     vector<edge> e;
19     int deep[MAX],fa[MAX],size[MAX],son[MAX];
20     int rnk[MAX],top[MAX],idx[MAX],tot;
21     int n,rt;
22     void init(int _n)
23     {
24         n=_n;
25         for(int i=0;i<=n;i++) mp[i].clear();
26         e.clear();
27         e.pb(edge(0,0));
28     }
29     void add_edge(int a,int b,type v=0)
30     {
31         e.pb(edge(a,b,v));
32         mp[a].pb(b);
33         mp[b].pb(a);
34     }
35     void dfs1(int x,int pre,int h)
36     {
37         int i,to;
38         deep[x]=h;
39         fa[x]=pre;
40         size[x]=1;
41         for(i=0;i<sz(mp[x]);i++)
42         {
43             to=mp[x][i];
44             if(to==pre) continue;
45             dfs1(to,x,h+1);

```

```

46             size[x]+=size[to];
47             if(son[x]==-1||size[to]>size[son[x]]) son
48                 [x]=to;
49         }
50     }
51     void dfs2(int x,int tp)
52     {
53         int i,to;
54         top[x]=tp;
55         idx[x]=++tot;
56         rnk[idx[x]]=x;
57         if(son[x]==-1) return;
58         dfs2(son[x],tp);
59         for(i=0;i<sz(mp[x]);i++)
60         {
61             to=mp[x][i];
62             if(to!=son[x]&&to!=fa[x]) dfs2(to,to);
63         }
64     }
65     void work(int _rt)
66     {
67         int i;
68         rt=_rt;
69         mem(son,-1);
70         tot=0;
71         dfs1(rt,0,0);
72         dfs2(rt,rt);
73     }
74     int LCA(int x,int y)
75     {
76         while(top[x]!=top[y])
77         {
78             if(deep[top[x]]<deep[top[y]]) swap(x,y);
79             x=fa[top[x]];
80         }
81         if(deep[x]>deep[y]) swap(x,y);
82         return x;
83     }
84     //node
85     void init_node()
86     {
87         build(n);
88     }
89     void modify_node(int x,int y,type val)
90     {
91         while(top[x]!=top[y])
92         {
93             if(deep[top[x]]<deep[top[y]]) swap(x,y);

```

```

91         update(idx[top[x]],idx[x],val);
92         x=fa[top[x]];
93     }
94     if(deep[x]>deep[y]) swap(x,y);
95     update(idx[x],idx[y],val);
96 }
97 type query_node(int x,int y)
98 {
99     type res=0;
100     while(top[x]!=top[y])
101     {
102         if(deep[top[x]]<deep[top[y]]) swap(x,y);
103         res+=query(idx[top[x]],idx[x]);
104         x=fa[top[x]];
105     }
106     if(deep[x]>deep[y]) swap(x,y);
107     res+=query(idx[x],idx[y]);
108     return res;
109 }
110 //path
111 void init_path()
112 {
113     v[idx[rt]]=0;
114     for(int i=1;i<n;i++)
115     {
116         if(deep[e[i].a]<deep[e[i].b]) swap(e[i].a
117             ,e[i].b);
118         v[idx[e[i].a]]=e[i].v;
119     }
120     build(n);
121 }
122 void modify_edge(int id,type val)
123 {
124     if(deep[e[id].a]>deep[e[id].b]) update(idx[e[
125         id].a],idx[e[id].a],val);
126     else update(idx[e[id].b],idx[e[id].b],val);
127 }
128 void modify_path(int x,int y,type val)
129 {
130     while(top[x]!=top[y])
131     {
132         if(deep[top[x]]<deep[top[y]]) swap(x,y);
133         update(idx[top[x]],idx[x],val);
134         x=fa[top[x]];
135     }
136     if(deep[x]>deep[y]) swap(x,y);
137     if(x!=y) update(idx[x]+1,idx[y],val);
138 }

```

```

137 type query_path(int x,int y)
138 {
139     type res=0;
140     while(top[x]!=top[y])
141     {
142         if(deep[top[x]]<deep[top[y]]) swap(x,y);
143         res+=query(idx[top[x]],idx[x]);
144         x=fa[top[x]];
145     }
146     if(deep[x]>deep[y]) swap(x,y);
147     if(x!=y) res+=query(idx[x]+1,idx[y]);
148     return res;
149 }
150 #undef type
151 }hld;
152 //*****attention!*****//
153 //hld.init(n)
154 //hld.add_edge(): undirected edge.
155 //*****//

```

3 数学

3.1 矩阵

```

1 struct Mat {
2     static const LL M = 2;
3     LL v[M][M];
4     Mat() { memset(v, 0, sizeof v); }
5     void eye() { FOR (i, 0, M) v[i][i] = 1; }
6     LL* operator [] (LL x) { return v[x]; }
7     const LL* operator [] (LL x) const { return v[x]
8         ]; }
9     Mat operator * (const Mat& B) {
10         const Mat& A = *this;
11         Mat ret;
12         FOR (k, 0, M)
13             FOR (i, 0, M) if (A[i][k])
14                 ret[i][j] = (ret[i][j] + A[i][k] * B[k][j]) %
15                     MOD;
16         return ret;
17     }
18     Mat pow(LL n) const {
19         Mat A = *this, ret; ret.eye();
20         for (; n; n >>= 1, A = A * A)
21             if (n & 1) ret = ret * A;
22         return ret;
23     }
24 }

```

```

22     }
23     Mat operator + (const Mat& B) {
24         const Mat& A = *this;
25         Mat ret;
26         FOR (i, 0, M)
27             FOR (j, 0, M)
28                 ret[i][j] = (A[i][j] + B[i][j]) % MOD;
29         return ret;
30     }
31     void prt() const {
32         FOR (i, 0, M)
33             FOR (j, 0, M)
34                 printf("%lld%c", (*this)[i][j], j == M - 1 ?
35                     '\n' : ' ');
36     };

```

3.2 快速乘

```

1  LL mul(LL a, LL b, LL m) {
2      LL ret = 0;
3      while (b) {
4          if (b & 1) {
5              ret += a;
6              if (ret >= m) ret -= m;
7          }
8          a += a;
9          if (a >= m) a -= m;
10         b >>= 1;
11     }
12     return ret;
13 }

```

3.3 快速幂

如果模数是素数，则可在函数体内加上 n

```

1  LL qpow(LL a, LL b, LL Mod){
2      LL ret=1;
3      while(b){
4          if(b&1)ret=(ret*a)%Mod;
5          a=(a*a)%Mod;
6          b>>=1;
7      }
8      return ret;
9  }

```

3.4 筛

3.4.1 线性筛素数

```

1  const LL p_max=1e6;
2  LL prime[p_max+100],p_sz;//用vector存素数能优化时间
3  void get_prime(){
4      bool vis[p_max+100];
5      for(int i=2;i<=p_max;i++){
6          if(!vis[i])prime[p_sz++]=i;
7          for(int j=0;j<p_sz&&prime[j]*i<=p_max;j++){
8              vis[prime[j]*i]=1;
9              if(i%prime[j]==0)break;
10         }
11     }
12 }

```

3.4.2 线性筛欧拉函数

```

1  const LL p_max=1e6;
2  LL phi[p_max+100],prime[p_max+100],p_sz=0;
3  bool vis[p_max+100];
4  void get_phitable(){
5      phi[1]=1;
6      for(int i=2;i<=p_max;i++){
7          if(!vis[i]){
8              prime[p_sz++]=i;
9              phi[i]=i-1;
10         }
11         LL d;
12         for(int j=0;j<p_sz&&(d=prime[j]*i)<=p_max;j++){
13             ++d;
14             vis[d]=1;
15             if(i%prime[j]==0){
16                 phi[d]=phi[i]*prime[j];
17                 break;
18             }
19             else phi[d]=phi[i]*(prime[j]-1);
20         }
21     }

```

3.4.3 线性筛莫比乌斯函数

```

1  const LL p_max=1e6;

```

```

2    LL mu[p_max+100], prime[p_max+100], p_sz;
3    bool vis[p_max+100];
4    void get_mutable(){
5        mu[1]=1;
6        for(int i=2; i<=p_max; i++){
7            if(!vis[i]){
8                prime[p_sz++]=i;
9                mu[i]=-1;
10           }
11       LL d;
12       for(int j=0; j<p_sz&&(d=prime[j]*i)<=p_max; j++){
13           vis[d]=1;
14           if(i%prime[j]==0){
15               mu[d]=0;
16               break;
17           }
18           else mu[d]=-mu[i];
19       }
20   }
21 }

```

3.4.4 杜教筛

求 $S(n) = \sum_{i=1}^n f(i)$, 其中 f 是一个积性函数。

构造一个积性函数 g , 那么由 $(f * g)(n) = \sum_{d|n} f(d)g(\frac{n}{d})$, 得到 $f(n) = (f * g)(n) - \sum_{d|n, d < n} f(d)g(\frac{n}{d})$ 。

$$g(1)S(n) = \sum_{i=1}^n (f * g)(i) - \sum_{i=1}^n \sum_{d|i, d < i} f(d)g(\frac{n}{d}) \quad (1)$$

$$\stackrel{t=\frac{i}{d}}{=} \sum_{i=1}^n (f * g)(i) - \sum_{t=2}^n g(t)S(\lfloor \frac{n}{t} \rfloor) \quad (2)$$

当然, 要能够由此计算 $S(n)$, 会对 f, g 提出一些要求:

$f * g$ 要能够快速求前缀和。

g 要能够快速求分段和 (前缀和)。

对于正常的积性函数 $g(1) = 1$, 所以不会有什么问题。

在预处理 $S(n)$ 前 $n^{\frac{2}{3}}$ 项的情况下复杂度是 $O(n^{\frac{2}{3}})$ 。

杜教筛筛欧拉函数

```

1    #include<iostream>
2    #include<unordered_map>
3    #include<vector>
4    using namespace std;
5    typedef long long ll;
6    const int maxn=5e6;
7    const int mod=1e9+7;
8    int Madd(int a,int b){
9        return a+b<mod?a+b:a+b-mod;
10   }

```

```

11   int Msub(int a,int b){
12       return a<b?a-b:a-b+mod;
13   }
14   int Mmul(int a,int b){
15       return (ll)a*b%mod;
16   }
17   vector<int>p;
18   int vis[maxn+10];
19   int phi[maxn+10];
20   unordered_map<ll,ll>map;
21   ll qphi(ll n){
22       if(n<maxn)return phi[n];
23       auto it=map.find(n);
24       if(it!=map.end())
25           return it->second;
26       ll res=n&1?Mmul((n+1)/2%mod,n%mod):Mmul(n/2%mod,
27           (n+1)%mod);
28       for(ll i=2,last;i<=n;i=last+1){
29           last=n/(n/i);
30           res=Msub(res,Mmul((last-i+1)%mod,qphi(n/i)));
31       }
32       map.emplace(n,res);
33       return res;
34   }
35   void init(){
36       phi[1]=1;
37       for(int i=2;i<maxn;i++){
38           if(!vis[i]){
39               p.push_back(i);
40               phi[i]=i-1;
41           }
42           for(int j=0;j<(int)p.size()&&i*p[j]<maxn;j++){
43               vis[i*p[j]]=1;
44               if(i%p[j]){
45                   phi[i*p[j]]=phi[i]*(p[j]-1);
46               }
47               else{
48                   phi[i*p[j]]=phi[i]*p[j];
49                   break;
50               }
51           }
52       }
53   }
54   int main(){
55       ios_base::sync_with_stdio(false);
56       cin.tie(0);
57       init();

```



```

58     long long a;
59     cin>>a;
60     cout<<qphi(a)<<endl;
61     return 0;
62 }
杜教筛莫比乌斯函数
1  #include<iostream>
2  #include<unordered_map>
3  #include<vector>
4  using namespace std;
5  typedef long long ll;
6  const int maxn=5e6;
7
8  vector<int>p;
9  int vis[maxn+10];
10 int mu[maxn+10];
11 unordered_map<ll,ll>map;
12 ll qmu(ll n){
13     if(n<maxn)return mu[n];
14     auto it=map.find(n);
15     if(it!=map.end())
16         return it->second;
17     ll res=1;
18     for(ll i=2,last;i<=n;i=last+1){
19         last=n/(n/i);
20         res-=(ll)(last-i+1)*qmu(n/i);
21     }
22     map.emplace(n,res);
23     return res;
24 }
25 void init(){
26     mu[1]=1;
27     for(int i=2;i<maxn;i++){
28         if(!vis[i]){
29             p.push_back(i);
30             mu[i]=-1;
31         }
32         for(int j=0;j<(int)p.size()&&i*p[j]<maxn;j++){
33             vis[i*p[j]]=1;
34             if(i%p[j])
35                 mu[i*p[j]]=-mu[i];
36             else
37                 break;
38         }
39     }
40     for(int i=2;i<maxn;i++)
41         mu[i]+=mu[i-1];
42 }

```

```

43 int main(){
44     ios_base::sync_with_stdio(false);
45     cin.tie(0);
46     init();
47     long long a,b;
48     cin>>a>>b;
49     cout<<qmu(b)-qmu(a-1)<<endl;
50     return 0;
51 }

```

3.5 素数测试

3.5.1 素数判断

前置：快速乘、快速幂 int 范围内只需检查 2, 7, 61 long long 范围 2, 325, 9375, 28178, 450775, 9780504, 1795265022 3E15 内 2, 2570940, 880937, 610386380, 4130785767 4E13 内 2, 2570940, 211991001, 3749873356

```

1 bool checkQ(LL a, LL n) {
2     if (n == 2 || a >= n) return 1;
3     if (n == 1 || !(n & 1)) return 0;
4     LL d = n - 1;
5     while (!(d & 1)) d >>= 1;
6     LL t = bin(a, d, n); // 不一定需要快速乘
7     while (d != n - 1 && t != 1 && t != n - 1) {
8         t = mul(t, t, n);
9         d <<= 1;
10    }
11    return t == n - 1 || d & 1;
12 }
13 bool primeQ(LL n) {
14     static vector<LL> t = {2, 325, 9375, 28178,
15         450775, 9780504, 1795265022};
16     if (n <= 1) return false;
17     for (LL k: t) if (!checkQ(k, n)) return false;
18     return true;
19 }

```

3.5.2 Pollard-Rho

```

1 mt19937 mt(time(0));
2 LL pollard_rho(LL n, LL c) {
3     LL x = uniform_int_distribution<LL>(1, n - 1)(mt
4         ), y = x;
5     auto f = [&](LL v) { LL t = mul(v, v, n) + c;
6         return t < n ? t : t - n; };
7     while (1) {

```

```

6      x = f(x); y = f(f(y));
7      if (x == y) return n;
8      LL d = gcd(abs(x - y), n);
9      if (d != 1) return d;
10   }
11 }
12
13 LL fac[100], fcnt;
14 void get_fac(LL n, LL cc = 19260817) {
15     if (n == 4) { fac[fcnt++] = 2; fac[fcnt++] = 2;
16         return; }
17     if (primeQ(n)) { fac[fcnt++] = n; return; }
18     LL p = n;
19     while (p == n) p = pollard_rho(n, --cc);
20     get_fac(p); get_fac(n / p);
21 }

```

3.6 BM 线性递推

```

1  typedef vector<int> VI;
2  namespace linear_seq
3  {
4      #define rep(i,a,n) for (int i=a;i<n;i++)
5      #define SZ(x) ((int)(x).size())
6      const ll mod=1e9+7;
7      ll powmod(ll a,ll b){ll res=1;a%=mod; assert(b
8          >=0); for(;b;b>>=1){if(b&1)res=res*a%mod;a=a
9          *a%mod;}return res;}
10     const int N=10010;
11     ll res[N],base[N],_c[N],_md[N];
12     vector<int> Md;
13     void mul(ll *a,ll *b,int k)
14     {
15         rep(i,0,k+k) _c[i]=0;
16         rep(i,0,k) if (a[i]) rep(j,0,k) _c[i+j]=(
17             _c[i+j]+a[i]*b[j])%mod;
18         for (int i=k+k-1;i>=k;i-- if (_c[i])
19             rep(j,0,SZ(Md)) _c[i-k+Md[j]]=( _c[i-k+Md[
20                 j]]+_c[i]*_md[Md[j]])%mod;
21         rep(i,0,k) a[i]=_c[i];
22     }
23     int solve(ll n,VI a,VI b){
24         ll ans=0,pnt=0;
25         int k=SZ(a);
26         assert(SZ(a)==SZ(b));
27         rep(i,0,k) _md[k-1-i]=-a[i];_md[k]=1;
28         Md.clear();

```

```

25     rep(i,0,k) if (_md[i]!=0) Md.push_back(i)
26     ;
27     rep(i,0,k) res[i]=base[i]=0;
28     res[0]=1;
29     while ((1ll<<pnt)<=n) pnt++;
30     for (int p=pnt;p>=0;p--) {
31         mul(res,res,k);
32         if ((n>>p)&1) {
33             for (int i=k-1;i>=0;i--) res[i+1]=
34                 res[i];res[0]=0;
35             rep(j,0,SZ(Md)) res[Md[j]]=(res[Md[
36                 j]]-res[k]*_md[Md[j]])%mod;
37         }
38     }
39     rep(i,0,k) ans=(ans+res[i]*b[i])%mod;
40     if (ans<0) ans+=mod;
41     return ans;
42 }
43
44 VI BM(VI s){
45     VI C(1,1),B(1,1);
46     int L=0,m=1,b=1;
47     rep(n,0,SZ(s)){
48         ll d=0;
49         rep(i,0,L+1) d=(d+(1ll)C[i]*s[n-i])%mod
50         ;
51         if(d==0) ++m;
52         else if(2*L<=n){
53             VI T=C;
54             ll c=mod-d*powmod(b,mod-2)%mod;//
55             Åa0^a
56             while (SZ(C)<SZ(B)+m) C.pb(0);
57             rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i
58                 ])%mod;
59             L=n+1-L; B=T; b=d; m=1;
60         } else {
61             ll c=mod-d*powmod(b,mod-2)%mod;//
62             Åa0^a
63             while (SZ(C)<SZ(B)+m) C.pb(0);
64             rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i
65                 ])%mod;
66             ++m;
67         }
68     }
69     return C;
70 }
71
72 int gao(VI a,ll n)
73 {
74     VI c=BM(a);

```

```

65         c.erase(c.begin());
66         rep(i,0,SZ(c)) c[i]=(mod-c[i])%mod;
67         return solve(n,c,VI(a.begin(),a.begin()+
        SZ(c)));
68     }
69     };//linear_seq::gao(VI{ },n-1)

```

```

38         t=x;
39         x%=b;
40         if(x<=0) x+=b;//or x<0
41         ll k=(t-x)/b;
42         y+=k*a;
43     }
44     return g;
45 }

```

3.7 扩展欧几里德

```

1  /*
2  解 $xa+yb=gcd(a,b)$ 
3  返回值为 $gcd(a,b)$ 
4  其中一组解为 $x\ y$ 
5  通解:
6   $x1=x+b/gcd(a,b)*t$ 
7   $y1=y-a/gcd(a,b)*t$ 
8  ( $t$ 为任意整数)
9  */
10 ll exgcd(ll a,ll b,ll &x,ll &y)
11 {
12     if(b==0)
13     {
14         x=1;
15         y=0;
16         return a;
17     }
18     ll g,t;
19     g=exgcd(b,a%b,x,y);
20     t=x;
21     x=y;
22     y=t-a/b*y;
23     return g;
24 }
25 //xa+yb=c 有解条件  $c\%gcd(a,b)==0$ 
26 ll linear_equation(ll a,ll b,ll c,ll &x,ll &y)
27 {
28     ll g,t;
29     g=exgcd(a,b,x,y);
30     if(!c) x=y=0;
31     else if((!a&&!b&&c)||c%g) return -1;//no
        solution
32     else if(!a&&b) x=1,y=c/b;
33     else if(a&&!b) x=c/a,y=-c/a;
34     else
35     {
36         a/=g,b/=g,c/=g;
37         x*=c,y*=c;

```

3.8 线性基

```

1  struct Base
2  {
3      #define type ll
4      #define mx 60
5      type d[mx+3];
6      int p[mx+3],cnt;
7      void init()
8      {
9          memset(d,0,sizeof(d));
10         cnt=0;
11     }
12     bool insert(type x,int pos=0)
13     {
14         int i;
15         for(i=mx;~i;i--)
16         {
17             if(!(x&(1LL<<i))) continue;
18             if(!d[i])
19             {
20                 cnt++;
21                 d[i]=x;
22                 p[i]=pos;
23                 break;
24             }
25             if(p[i]<pos)
26             {
27                 swap(d[i],x);
28                 swap(p[i],pos);
29             }
30             x^=d[i];
31         }
32         return x>0;
33     }
34     type query_max(int pos=-1)
35     {
36         int i;

```

```

37     type res=0;
38     for(i=mx;~i;i--)
39     {
40         if(p[i]>=pos)
41         {
42             if((res^d[i])>res) res^=d[i];
43         }
44     }
45     return res;
46 }
47 type query_min(int pos=-1)
48 {
49     for(int i=0;i<=mx;i++)
50     {
51         if(d[i]&& p[i]>=pos) return d[i];
52     }
53     return 0;
54 }
55 void merge(Base x)
56 {
57     if(cnt<x.cnt)
58     {
59         swap(cnt,x.cnt);
60         swap(d,x.d);
61         swap(p,x.p);
62     }
63     for(int i=mx;~i;i--)
64     {
65         if(x.d[i]) insert(x.d[i]);
66     }
67 }
68 //kth min
69 //first use rebuild()
70 type tp[mx+3];
71 void rebuild()
72 {
73     int i,j;
74     cnt=0;
75     for(i=mx;~i;i--)
76     {
77         for(j=i-1;~j;j--)
78         {
79             if(d[i]&(1LL<<j)) d[i]^=d[j];
80         }
81     }
82     for(i=0;i<=mx;i++)
83     {
84         if(d[i]) tp[cnt++]=d[i];

```

```

85     }
86 }
87 type kth(type k)
88 {
89     type res=0;
90     if(k>=(1LL<<cnt)) return -1;
91     for(int i=mx;~i;i--)
92     {
93         if(k&(1LL<<i)) res^=tp[i];
94     }
95     return res;
96 }
97 };

```

3.9 exBSGS

```

1 //a^x_i=b (mod c)
2 ll exBSGS(ll a,ll b,ll c)
3 {
4     ll i,g,d,num,now,sq,t,x,y;
5     if(c==1) return b?-1:(a!=1);
6     if(b==1) return a?0:-1;
7     if(a%c==0) return b?-1:1;
8     num=0;
9     d=1;
10    while((g=__gcd(a,c))>1)
11    {
12        if(b%g) return -1;
13        num++;
14        b/=g;
15        c/=g;
16        d=(d*a/g)%c;
17        if(d==b) return num;
18    }
19    mp.clear();
20    sq=ceil(sqrt(c));
21    t=1;
22    for(i=0;i<sq;i++)
23    {
24        if(!mp.count(t)) mp[t]=i;
25        else mp[t]=min(mp[t],i);
26        t=t*a%c;
27    }
28    for(i=0;i<sq;i++)
29    {
30        exgcd(d,c,x,y);
31        x=(x*b%c+c)%c;

```

```

32         if(mp.count(x)) return i*sq+mp[x]+num;
33         d=d*t%c;
34     }
35     return -1;
36 }

```

3.10 中国剩余定理

```

1 //m是除数 r是余数 p是除数的LCM(也就是答案的循环节)
2 int CRT(int *m,int *r,int n)
3 {
4     int p=m[0],res=r[0],x,y,g;
5     for(int i=1;i<n;i++)
6     {
7         g=exgcd(p,m[i],x,y);
8         if((r[i]-res)%g) return -1;//无解
9         x=(r[i]-res)/g*x%(m[i]/g);
10        res+=x*p;
11        p=p/g*m[i];
12        res%=p;
13    }
14    return res>0?res:res+p;
15 }

```

3.11 类欧几里德

$$m = \lfloor \frac{an+b}{c} \rfloor.$$

$f(a, b, c, n) = \sum_{i=0}^n \lfloor \frac{ai+b}{c} \rfloor$: 当 $a \geq c$ or $b \geq c$ 时, $f(a, b, c, n) = (\frac{a}{c})n(n+1)/2 + (\frac{b}{c})(n+1) + f(a \bmod c, b \bmod c, c, n)$; 否则 $f(a, b, c, n) = nm - f(c, c-b-1, a, m-1)$ 。

$g(a, b, c, n) = \sum_{i=0}^n i \lfloor \frac{ai+b}{c} \rfloor$: 当 $a \geq c$ or $b \geq c$ 时, $g(a, b, c, n) = (\frac{a}{c})n(n+1)(2n+1)/6 + (\frac{b}{c})n(n+1)/2 + g(a \bmod c, b \bmod c, c, n)$; 否则 $g(a, b, c, n) = \frac{1}{2}(n(n+1)m - f(c, c-b-1, a, m-1) - h(c, c-b-1, a, m-1))$ 。

$h(a, b, c, n) = \sum_{i=0}^n \lfloor \frac{ai+b}{c} \rfloor^2$: 当 $a \geq c$ or $b \geq c$ 时, $h(a, b, c, n) = (\frac{a}{c})^2 n(n+1)(2n+1)/6 + (\frac{b}{c})^2 (n+1) + (\frac{a}{c})(\frac{b}{c})n(n+1) + h(a \bmod c, b \bmod c, c, n) + 2(\frac{b}{c})g(a \bmod c, b \bmod c, c, n)$; 否则 $h(a, b, c, n) = nm(m+1) - 2g(c, c-b-1, a, m-1) - 2f(c, c-b-1, a, m-1) - f(a, b, c, n)$ 。

3.12 逆元

预处理 1 n 的逆元

```

1 LL inv[N] = {-1, 1};
2 void inv_init(LL n, LL p) {
3     inv[1] = 1;
4     for(int i=2;i<=n;i++)

```

```

5     inv[i] = (p - p / i) * inv[p % i] % p;
6 }

```

预处理阶乘及其逆元

```

1 LL invf[M], fac[M] = {1};
2 void fac_inv_init(LL n, LL p) {
3     for(int i = 2; i <= n; i++)
4         fac[i] = i * fac[i - 1] % p;
5     invf[n - 1] = qpow(fac[n - 1], p - 2, p);
6     for(int i = n-1; i >= 0; i--)
7         invf[i] = invf[i + 1] * (i + 1) % p;
8 }

```

3.13 组合数

如果数较小, 模较大时使用逆元前置模板: 逆元-预处理阶乘及其逆元

```

1 inline LL C(LL n, LL m) { // n >= m >= 0
2     return n < m || m < 0 ? 0 : fac[n] * invf[m] %
3         MOD * invf[n - m] % MOD;
4 }

```

如果模数较小, 数字较大, 使用 Lucas 定理

前置模板可选 1: 求组合数 (如果使用阶乘逆元, 需

```

1 fac_inv_init(MOD, MOD)

```

前置模板可选 2: 模数不固定下使用, 无法单独使用。

```

1 LL C(LL n, LL m) { // m >= n >= 0
2     if (m - n < n) n = m - n;
3     if (n < 0) return 0;
4     LL ret = 1;
5     for(int i=1;i<=n;i++)
6         ret = ret * (m - n + i) % MOD * qpow(i, MOD -
7             2, MOD) % MOD;
8     return ret;
9 }
10 LL Lucas(LL n, LL m) { // m >= n >= 0
11     return m ? C(n % MOD, m % MOD) * Lucas(n / MOD,
12         m / MOD) % MOD : 1;
13 }

```

3.14 公式

3.14.1 数论公式

当 $x \geq \phi(p)$ 时有 $a^x \equiv a^{x \bmod \phi(p) + \phi(p)} \pmod{p}$

4 图论

4.1 k 短路

POJ 2449

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  const int maxN=10000;
4  const int INF=0x3f3f3f3f;
5  typedef pair<int,int>P;
6  int n,m,s,t,k;
7  int dist[maxN],tdist[maxN],cnt[maxN];
8  bool f[maxN];
9  vector<P>Adj[maxN];
10 vector<P>Rev[maxN];
11 struct edge{
12     int to,len;
13     edge(){}
14     edge(int t,int l):to(t),len(l){}
15 };
16 priority_queue<edge> q;
17 bool operator< (const edge &a,const edge &b){
18     return (a.len+dist[a.to])>(b.len+dist[b.to]);
19 }
20 void dijkstra(){
21     memset(dist,0,sizeof(dist));
22     fill(tdist,tdist+maxN,INF);
23     tdist[t]=0;
24     while(!q.empty())q.pop();
25     q.push(edge(t,0));
26     while(!q.empty()){
27         int x=q.top().to;
28         int d=q.top().len;
29         q.pop();
30         if(tdist[x]<d)continue;
31         for(int i=0;i<(int)Rev[x].size();i++){
32             int y=Rev[x][i].first;
33             int len=Rev[x][i].second;
34             if(d+len<tdist[y]){
35                 tdist[y]=d+len;
36                 q.push(edge(y,tdist[y]));
37             }
38         }
39     }
40     for(int i=1;i<=n;i++){
41         dist[i]=tdist[i];
42     }
43     int aStar(){

```

$$\mu^2(n) = \sum_{d^2|n} \mu(d)$$

$$\sum_{d|n} \varphi(d) = n$$

$$\sum_{d|n} 2^{\omega(d)} = \sigma_0(n^2), \text{ 其中 } \omega \text{ 是不同素因子个数}$$

$$\sum_{d|n} \mu^2(d) = 2^{\omega(d)}$$

3.14.2 一些数论函数求和的例子

$$\sum_{i=1}^n i[gcd(i,n)=1] = \frac{n\varphi(n)+[n=1]}{2}$$

$$\sum_{i=1}^n \sum_{j=1}^m [gcd(i,j)=x] = \sum_d \mu(d) \lfloor \frac{n}{dx} \rfloor \lfloor \frac{m}{dx} \rfloor$$

$$\sum_{i=1}^n \sum_{j=1}^m gcd(i,j) = \sum_{i=1}^n \sum_{j=1}^m \sum_{d|gcd(i,j)} \varphi(d) = \sum_d \varphi(d) \lfloor \frac{n}{d} \rfloor \lfloor \frac{m}{d} \rfloor$$

$$S(n) = \sum_{i=1}^n \mu(i) = 1 - \sum_{i=1}^n \sum_{d|i, d<i} \mu(d) \stackrel{t=\frac{i}{d}}{=} 1 - \sum_{t=2}^n S(\lfloor \frac{n}{t} \rfloor)$$

$$\text{利用 } [n=1] = \sum_{d|n} \mu(d)$$

$$S(n) = \sum_{i=1}^n \varphi(i) = \sum_{i=1}^n i - \sum_{i=1}^n \sum_{d|i, d<i} \varphi(i) \stackrel{t=\frac{i}{d}}{=} \frac{i(i+1)}{2} - \sum_{t=2}^n S(\frac{n}{t})$$

$$\text{利用 } n = \sum_{d|n} \varphi(d)$$

$$\sum_{i=1}^n \mu^2(i) = \sum_{i=1}^n \sum_{d^2|i} \mu(d) = \sum_{d=1}^{\lfloor \sqrt{n} \rfloor} \mu(d) \lfloor \frac{n}{d^2} \rfloor$$

$$\sum_{i=1}^n \sum_{j=1}^m gcd^2(i,j) = \sum_d d^2 \sum_t \mu(t) \lfloor \frac{n}{dt} \rfloor^2$$

$$\stackrel{x=dt}{=} \sum_x \lfloor \frac{n}{x} \rfloor^2 \sum_{d|x} d^2 \mu(\frac{x}{d})$$

$$\sum_{i=1}^n \varphi(i) = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n [i \perp j] - 1 = \frac{1}{2} \sum_{i=1}^n \mu(i) \cdot \lfloor \frac{n}{i} \rfloor^2 - 1$$

3.14.3 莫比乌斯反演

$$g(n) = \sum_{d|n} f(d) \Leftrightarrow f(n) = \sum_{d|n} \mu(d) g(\frac{n}{d})$$

$$f(n) = \sum_{n|d} g(d) \Leftrightarrow g(n) = \sum_{n|d} \mu(\frac{d}{n}) f(d)$$

3.14.4 低阶等幂求和

$$\sum_{i=1}^n i^1 = \frac{n(n+1)}{2} = \frac{1}{2}n^2 + \frac{1}{2}n$$

$$\sum_{i=1}^n i^2 = \frac{n(n+1)(2n+1)}{6} = \frac{1}{3}n^3 + \frac{1}{2}n^2 + \frac{1}{6}n$$

$$\sum_{i=1}^n i^3 = \left[\frac{n(n+1)}{2} \right]^2 = \frac{1}{4}n^4 + \frac{1}{2}n^3 + \frac{1}{4}n^2$$

$$\sum_{i=1}^n i^4 = \frac{n(n+1)(2n+1)(3n^2+3n-1)}{30} = \frac{1}{5}n^5 + \frac{1}{2}n^4 + \frac{1}{3}n^3 - \frac{1}{30}n$$

$$\sum_{i=1}^n i^5 = \frac{n^2(n+1)^2(2n^2+2n-1)}{12} = \frac{1}{6}n^6 + \frac{1}{2}n^5 + \frac{5}{12}n^4 - \frac{1}{12}n^2$$

3.14.5 一些组合公式

$$\text{错排公式: } D_1 = 0, D_2 = 1, D_n = (n-1)(D_{n-1} + D_{n-2}) = n!(\frac{1}{2!} - \frac{1}{3!} + \dots + (-1)^n \frac{1}{n!}) = \lfloor \frac{n!}{e} + 0.5 \rfloor$$

卡特兰数 (n 对括号合法方案数, n 个结点二叉树个数, $n \times n$ 方格中对角线下方的单调路径数, 凸 $n+2$ 边形的三角形划分数, n 个元素的合法出栈序列数): $C_n = \frac{1}{n+1} \binom{2n}{n} = \frac{(2n)!}{(n+1)!n!}$

3.15 Theorem

费马平方和定理: 奇素数能表示为两个平方数之和的充分必要条件是素数被 4 除余 1。

```

44     if(dist[s]==INF)return -1;
45     while(!q.empty())q.pop();
46     q.push(edge(s,0));
47     memset(cnt,0,sizeof(cnt));
48     while(!q.empty()){
49         int x=q.top().to;
50         int d=q.top().len;
51         q.pop();
52         cnt[x]++;
53         if(cnt[t]==k)return d;
54         if(cnt[x]>k)continue;
55         for(int i=0;i<(int)Adj[x].size();i++){
56             int y=Adj[x][i].first;
57             int len=Adj[x][i].second;
58             q.push(edge(y,d+len));
59         }
60     }
61     return -1;
62 }
63 int main(){
64     scanf("%d%d",&n,&m);
65     for(int i=0;i<m;i++){
66         int st,ed,l;
67         scanf("%d%d%d",&st,&ed,&l);
68         Adj[st].push_back(make_pair(ed,l));
69         Rev[ed].push_back(make_pair(st,l));
70     }
71     scanf("%d%d%d",&s,&t,&k);
72     if(s==t)k++;
73     dijkstra();
74     printf("%d\n",aStar());
75     return 0;
76 }

```

4.2 最大流

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  const int maxn = 11000;
5  const int maxm = 110000;
6  const int INF = 0x3f3f3f3f;
7  struct Max_Flow{
8      int first[maxn],nxt[maxm*2],des[maxm*2],c[maxm
          *2],tot;
9      int dep[maxn];int ss,tt;
10     Max_Flow(){ clear(); }

```

```

11     void clear(){
12         memset(first,-1,sizeof first);tot =-1;
13     }
14     inline void addEdge(int u,int v,int w){
15         tot++;
16         des[tot] = v;c[tot] =w;
17         nxt[tot] = first[u];first[u] = tot;
18     }
19     bool bfs(){
20         memset(dep,-1,sizeof dep);
21         dep[ss] =0;
22         queue<int> Q;Q.push(ss);
23         while (!Q.empty()){
24             int q = Q.front();Q.pop();
25             for (int t = first[q];t!=-1;t= nxt[t]){
26                 int v = des[t],cx = c[t];
27                 if (dep[v]==-1&&cx){
28                     dep[v] = dep[q]+1;
29                     Q.push(v);
30                 }
31             }
32         }
33         return dep[tt]!=-1;
34     }
35     int dfs(int node,int now){
36         if (node==tt)return now;
37         int res =0;
38         for (int t = first[node];t!=-1&&res<now;t=nxt
            [t]){
39             int v = des[t],cx = c[t];
40             if (dep[v]==dep[node]+1&&cx){
41                 int x = min(cx,now-res);
42                 x = dfs(v,x);
43                 res+=x;c[t]-=x;c[t^1]+=x;
44             }
45         }
46         if (!res) dep[node] = -2;
47         return res;
48     }
49     // tuple<from,to,flow>
50     void init(vector<tuple<int,int,int> > Edge){
51         for (auto tp : Edge){
52             int u,v,w;tie(u,v,w) = tp;
53             addEdge(u,v,w);addEdge(v,u,0);
54         }
55     }
56     // s->t max_flow
57     ll max_flow(int s,int t){

```

```

58     ss = s; tt = t;
59     ll res = 0, del = 0;
60     while (bfs()){while (del = dfs(ss, INF)){res
        += del;}}
61     return res;
62 }
63 }net;
64 int n, m, s, t;
65 vector<tuple<int, int, int> > E;
66 int main(){
67     scanf("%d%d%d", &n, &m, &s, &t);
68     for (int i=0; i<m; i++){
69         int u, v, w;
70         scanf("%d%d%d", &u, &v, &w);
71         E.push_back(make_tuple(u, v, w));
72     }
73     net.init(E);
74     printf("%lld\n", net.max_flow(s, t));
75     return 0;
76 }

```

4.3 最小费用流

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int maxn = 2000+50;
4  const int maxm = 20000+50;
5  const int INF = 0x3f3f3f3f;
6  int m, n, ss, tt, dis[maxn], pre[maxn];
7  int first[maxn], from[maxm*2], des[maxm*2], nxt[maxm
    *2], cost[maxm*2], flow[maxm*2], tot;
8  bool in[maxn];
9  inline void addE(int x, int y, int f, int c){
10     tot++;
11     from[tot] = x; des[tot] = y;
12     flow[tot] = f; cost[tot] = c;
13     nxt[tot] = first[x]; first[x] = tot;
14 }
15 inline void addEdge(int x, int y, int f, int c){
16     addE(x, y, f, c); addE(y, x, 0, -c);
17 }
18 void input(){
19     scanf("%d", &n, &m);
20     tot = -1;
21     memset(first, -1, sizeof first);
22     for (int i=0; i<m; i++){
23         int u, v, c;

```

```

24     scanf("%d%d", &u, &v, &c);
25     addEdge(u, v, 1, c); addEdge(v, u, 1, c);
26 }
27 addEdge(0, 1, 2, 0);
28 }
29 bool spfa(){
30     memset(in, 0, sizeof in);
31     memset(dis, INF, sizeof dis);
32     memset(pre, -1, sizeof pre);
33     dis[ss] = 0; in[ss] = 1;
34     queue<int> Q; Q.push(ss);
35     while (!Q.empty()){
36         int q = Q.front();
37         Q.pop(); in[q] = 0;
38         for (int t = first[q]; t != -1; t = nxt[t]){
39             int v = des[t], len = cost[t], cx = flow[t];
40             if (cx && dis[v] > dis[q] + len){
41                 dis[v] = dis[q] + len;
42                 pre[v] = t;
43                 if (!in[v]){
44                     Q.push(v); in[v] = 1;
45                 }
46             }
47         }
48     }
49     return pre[tt] != -1;
50 }
51 void solve(){
52     ss = 0; tt = n;
53     int totflow = 0, totcost = 0, nowflow = 0, nowcost = 0;
54     while (spfa()){
55         nowcost = 0; nowflow = INF;
56         int now = pre[tt];
57         while (now != -1){
58             nowflow = min(nowflow, flow[now]);
59             now = pre[from[now]];
60         }
61         now = pre[tt];
62         while (now != -1){
63             flow[now] -= nowflow;
64             flow[now^1] += nowflow;
65             nowcost += cost[now];
66             now = pre[from[now]];
67         }
68         nowcost += nowflow;
69         totflow += nowflow;
70         totcost += nowcost;
71     }

```



```

72     cout<<totcost<<endl;
73 }
74 int main(){
75     input();
76     solve();
77     return 0;
78 }

```

4.4 点分治

```

1  //求树上长度小于等于k的有向路径数
2  #include<stdio.h>
3  #include<algorithm>
4  #include<cstring>
5  using namespace std;
6  const int MAX = 1e4+100;
7  const int INF = 0x3f3f3f3f;
8  int first [MAX*2]; int des[MAX*2];
9  int len[MAX*2]; int nxt[MAX*2];
10 int n,k,tot; int a[MAX]; int sum[MAX];
11 int dp[MAX]; int dis[MAX]; int num,ans;
12 bool vis[MAX]; int Sum,Min,Minid;
13 void init(){
14     memset(first,0,sizeof first);
15     tot =0; ans =0;
16     memset(vis,0,sizeof vis);
17 }
18 inline void add(int x,int y,int z){
19     tot++;
20     des[tot]= y; len[tot] =z;
21     nxt[tot] = first[x]; first[x] = tot;
22 }
23 void input(){
24     for (int i=1;i<n;i++){
25         int u,v,w;
26         scanf("%d%d%d",&u,&v,&w);
27         add(u,v,w); add(v,u,w);
28     }
29 }
30 void dfs1(int node,int father){
31     sum[node] = 1; dp[node] = 0;
32     for (int t = first[node];t;t = nxt[t]){
33         int v = des[t];
34         if (v == father||vis[v]){ continue; }
35         continue;
36     }
37     dfs1(v,node);

```

```

38     sum[node] += sum[v];
39     dp[node] = max(dp[node],sum[v]);
40 }
41 }
42 void dfs2(int node,int father){
43     int temp = max(dp[node],Sum-sum[node]);
44     if (temp<Min){
45         Min = temp; Minid = node;
46     }
47     for (int t = first[node];t;t = nxt[t]){
48         int v = des[t];
49         if (v==father||vis[v]){ continue; }
50         dfs2(v,node);
51     }
52 }
53 int getRoot(int u){
54     dfs1(u,0); Sum = sum[u];
55     Min = INF; Minid = -1;
56     dfs2(u,0);
57     return Minid;
58 }
59 void getDist(int node,int father,int dist){
60     dis[num++] = dist;
61     for (int t = first[node];t;t = nxt[t]){
62         int v =des[t];
63         if (v == father||vis[v]){ continue; }
64         getDist(v,node,dist+len[t]);
65     }
66 }
67 int calc (int u,int val){
68     num=0; int res =0;
69     getDist(u,0,0);
70     sort(dis,dis+num);
71     int i=0;int j=num-1;
72     while (i<j){
73         if (dis[i]+dis[j]+2*val<=k){
74             res+=j-i;
75             i++;
76         }else{ j--; }
77     }
78     return res;
79 }
80 void solve(int u){
81     int root = getRoot(u);
82     ans +=calc(root,0); vis[root] = true;
83     for (int t = first[root];t;t = nxt[t]){
84         int v = des[t];
85         if (vis[v]){

```

```

86         continue;
87     }
88     ans-=calc(v,len[t]);
89     solve(v);
90 }
91 }
92 int main(){
93     while (scanf("%d%d",&n,&k)!=EOF&&n&&k){
94         init();
95         input();
96         solve(1);
97         printf("%d\n",ans);
98     }
99     return 0;
100 }

```

5 计算几何

```

1
2 #include <iostream>
3 #include <cstdio>
4 #include <cmath>
5 #include <algorithm>
6
7
8 using namespace std;
9 const double PI = acos(-1.0);
10 const double eps = 1e-10;
11
12 /*****常用函数*****/
13 //判断ta与tb的大小关系
14 int sgn( double ta, double tb)
15 {
16     if(fabs(ta-tb)<eps)return 0;
17     if(ta<tb) return -1;
18     return 1;
19 }
20
21 //点
22 class Point
23 {
24     public:
25
26     double x, y;
27
28     Point(){

```

```

29     Point( double tx, double ty){ x = tx, y = ty;}
30
31     bool operator < (const Point &_se) const
32     {
33         return x<_se.x || (x==_se.x && y<_se.y);
34     }
35     friend Point operator + (const Point &_st,const
36         Point &_se)
37     {
38         return Point(_st.x + _se.x, _st.y + _se.y);
39     }
40     friend Point operator - (const Point &_st,const
41         Point &_se)
42     {
43         return Point(_st.x - _se.x, _st.y - _se.y);
44     }
45     double operator ^(const Point &b)const
46     {
47         return x*b.y - y*b.x;
48     }
49     //点位置相同(double类型)
50     bool operator == (const Point &_off)const
51     {
52         return sgn(x, _off.x) == 0 && sgn(y, _off.y)
53             == 0;
54     }
55 };
56
57 /*****常用函数*****/
58 //点乘
59 double dot(const Point &po,const Point &ps,const
60     Point &pe)
61 {
62     return (ps.x - po.x) * (pe.x - po.x) + (ps.y -
63         po.y) * (pe.y - po.y);
64 }
65 //叉乘
66 double xmult(const Point &po,const Point &ps,const
67     Point &pe)
68 {
69     return (ps.x - po.x) * (pe.y - po.y) - (pe.x -
70         po.x) * (ps.y - po.y);
71 }
72 //两点间距离的平方
73 double getdis2(const Point &st,const Point &se)
74 {
75     return (st.x - se.x) * (st.x - se.x) + (st.y -

```

```

        se.y) * (st.y - se.y);
70     }
71     //两点间距离
72     double getdis(const Point &st,const Point &se)
73     {
74         return sqrt((st.x - se.x) * (st.x - se.x) + (st.
            y - se.y) * (st.y - se.y));
75     }
76
77     //两点表示的向量
78     class Line
79     {
80     public:
81
82         Point s, e;//两点表示, 起点[s], 终点[e]
83         double a, b, c;//一般式,ax+by+c=0
84         double angle;//向量的角度, [-pi,pi]
85
86         Line(){}
87         Line( Point ts, Point te):s(ts),e(te){}//
            get_angle();}
88         Line(double _a,double _b,double _c):a(_a),b(_b),
            c(_c){}
89
90         //排序用
91         bool operator < (const Line &ta)const
92         {
93             if(angle!=ta.angle) return angle<ta.angle;
94             return ((s - ta.s)^(ta.e - ta.s)) < 0;
95         }
96         //向量与向量的叉乘
97         friend double operator / ( const Line &_st,
            const Line &_se)
98         {
99             return (_st.e.x - _st.s.x) * (_se.e.y - _se.s
                .y) - (_st.e.y - _st.s.y) * (_se.e.x -
                    _se.s.x);
100         }
101         //向量间的点乘
102         friend double operator *( const Line &_st, const
            Line &_se)
103         {
104             return (_st.e.x - _st.s.x) * (_se.e.x - _se.s
                .x) - (_st.e.y - _st.s.y) * (_se.e.y -
                    _se.s.y);
105         }
106         //从两点表示转换为一般表示
107         //a=y2-y1,b=x1-x2,c=x2*y1-x1*y2
108
109         bool pton()
110         {
111             a = e.y - s.y;
112             b = s.x - e.x;
113             c = e.x * s.y - e.y * s.x;
114             return true;
115         }
116         //半平面交用
117         //点在向量左边(右边的小于号改成大于号即可,在对应直
            线上则加上=号)
118         friend bool operator < (const Point &_Off, const
            Line &_Ori)
119         {
120             return (_Ori.e.y - _Ori.s.y) * (_Off.x - _Ori
                .s.x)
121                 < (_Off.y - _Ori.s.y) * (_Ori.e.x - _Ori.s.x)
122                 ;
123         }
124         //求直线或向量的角度
125         double get_angle( bool isVector = true)
126         {
127             angle = atan2( e.y - s.y, e.x - s.x);
128             if(!isVector && angle < 0)
129                 angle += PI;
130             return angle;
131         }
132         //点在线段或直线上 1:点在直线上 2:点在s,e所在矩形内
133         bool has(const Point &_Off, bool isSegment =
            false) const
134         {
135             bool ff = sgn( xmult( s, e, _Off), 0) == 0;
136             if( !isSegment) return ff;
137             return ff
138                 && sgn(_Off.x - min(s.x, e.x), 0) >= 0 && sgn
                    (_Off.x - max(s.x, e.x), 0) <= 0
139                 && sgn(_Off.y - min(s.y, e.y), 0) >= 0 && sgn
                    (_Off.y - max(s.y, e.y), 0) <= 0;
140         }
141         //点到直线/线段的距离
142         double dis(const Point &_Off, bool isSegment =
            false)
143         {
144             ///化为一般式
145             pton();
146             //到直线垂足的距离
147             double td = (a * _Off.x + b * _Off.y + c) /

```

```

    sqrt(a * a + b * b);
148 //如果是线段判断垂足
149 if(isSegment)
150 {
151     double xp = (b * b * _Off.x - a * b *
        _Off.y - a * c) / (a * a + b * b);
152     double yp = (-a * b * _Off.x + a * a *
        _Off.y - b * c) / (a * a + b * b);
153     double xb = max(s.x, e.x);
154     double yb = max(s.y, e.y);
155     double xs = s.x + e.x - xb;
156     double ys = s.y + e.y - yb;
157     if(xp > xb + eps || xp < xs - eps || yp >
        yb + eps || yp < ys - eps)
158         td = min( getdis(_Off,s), getdis(_Off,e))
        ;
159 }
160 return fabs(td);
161 }
162
163 //关于直线对称的点
164 Point mirror(const Point &_Off)
165 {
166     ///注意先转为一般式
167     Point ret;
168     double d = a * a + b * b;
169     ret.x = (b * b * _Off.x - a * a * _Off.x - 2
        * a * b * _Off.y - 2 * a * c) / d;
170     ret.y = (a * a * _Off.y - b * b * _Off.y - 2
        * a * b * _Off.x - 2 * b * c) / d;
171     return ret;
172 }
173 //计算两点的中垂线
174 static Line ppline(const Point &_a,const Point &
    _b)
175 {
176     Line ret;
177     ret.s.x = (_a.x + _b.x) / 2;
178     ret.s.y = (_a.y + _b.y) / 2;
179     //一般式
180     ret.a = _b.x - _a.x;
181     ret.b = _b.y - _a.y;
182     ret.c = (_a.y - _b.y) * ret.s.y + (_a.x - _b.
        x) * ret.s.x;
183     //两点式
184     if(fabs(ret.a) > eps)
185     {
186         ret.e.y = 0.0;
187         ret.e.x = - ret.c / ret.a;
188         if(ret.e == ret. s)
189         {
190             ret.e.y = 1e10;
191             ret.e.x = - (ret.c - ret.b * ret.e.y)
                / ret.a;
192         }
193     }
194     else
195     {
196         ret.e.x = 0.0;
197         ret.e.y = - ret.c / ret.b;
198         if(ret.e == ret. s)
199         {
200             ret.e.x = 1e10;
201             ret.e.y = - (ret.c - ret.a * ret.e.x)
                / ret.b;
202         }
203     }
204     return ret;
205 }
206
207 //-----直线和直线（向量）-----
208 //向量向左边平移t的距离
209 Line& moveLine( double t)
210 {
211     Point of;
212     of = Point( -( e.y - s.y), e.x - s.x);
213     double dis = sqrt( of.x * of.x + of.y * of.y)
        ;
214     of.x= of.x * t / dis, of.y = of.y * t / dis;
215     s = s + of, e = e + of;
216     return *this;
217 }
218 //直线重合
219 static bool equal(const Line &_st,const Line &
    _se)
220 {
221     return _st.has( _se.e) && _se.has( _st.s);
222 }
223 //直线平行
224 static bool parallel(const Line &_st,const Line
    &_se)
225 {
226     return sgn( _st / _se, 0) == 0;
227 }
228 //两直线（线段）交点
229 //返回-1代表平行，0代表重合，1代表相交

```

```

230 static bool crossLPt(const Line &_st,const Line
      &_se, Point &ret)
231 {
232     if(parallel(_st,_se))
233     {
234         if(Line::equal(_st,_se)) return 0;
235         return -1;
236     }
237     ret = _st.s;
238     double t = ( Line(_st.s,_se.s) / _se) / ( _st
      / _se);
239     ret.x += (_st.e.x - _st.s.x) * t;
240     ret.y += (_st.e.y - _st.s.y) * t;
241     return 1;
242 }
243 //-----线段和直线（向量）-----
244 //直线和线段相交
245 //参数：直线[_st],线段[_se]
246 friend bool crossSL( Line &_st, Line &_se)
247 {
248     return sgn( xmult( _st.s, _se.s, _st.e) *
      xmult( _st.s, _st.e, _se.e), 0) >= 0;
249 }
250
251 //判断线段是否相交(注意添加eps)
252 static bool isCrossSS( const Line &_st, const
      Line &_se)
253 {
254     //1.快速排斥试验判断以两条线段为对角线的两个矩
      形是否相交
255     //2.跨立试验（等于0时端点重合）
256     return
257     max(_st.s.x, _st.e.x) >= min(_se.s.x, _se.e.x
      ) &&
258     max(_se.s.x, _se.e.x) >= min(_st.s.x, _st.e.x
      ) &&
259     max(_st.s.y, _st.e.y) >= min(_se.s.y, _se.e.y
      ) &&
260     max(_se.s.y, _se.e.y) >= min(_st.s.y, _st.e.y
      ) &&
261     sgn( xmult( _se.s, _st.s, _se.e) * xmult( _se
      .s, _se.e, _st.s), 0) >= 0 &&
262     sgn( xmult( _st.s, _se.s, _st.e) * xmult( _st
      .s, _st.e, _se.s), 0) >= 0;
263 }
264 };
265
266 //寻找凸包的graham 扫描法所需的排序函数
267 Point gsort;
268 bool gcmp( const Point &ta, const Point &tb)/// 选取
      与最后一条确定边夹角最小的点，即余弦值最大者
269 {
270     double tmp = xmult( gsort, ta, tb);
271     if( fabs( tmp) < eps)
272     return getdis( gsort, ta) < getdis( gsort, tb);
273     else if( tmp > 0)
274     return 1;
275     return 0;
276 }
277
278 class Polygon
279 {
280 public:
281     const static int maxpn = 5e4+7;
282     Point pt[maxpn];///点（顺时针或逆时针）
283     Line dq[maxpn]; //求半平面交打开注释
284     int n;///点的个数
285
286     //求多边形面积，多边形内点必须顺时针或逆时针
287     double area()
288     {
289         double ans = 0.0;
290         for(int i = 0; i < n; i++)
291         {
292             int nt = (i + 1) % n;
293             ans += pt[i].x * pt[nt].y - pt[nt].x * pt[i].
                y;
294         }
295         return fabs( ans / 2.0);
296     }
297
298     //求多边形重心，多边形内点必须顺时针或逆时针
299     Point gravity()
300     {
301         Point ans;
302         ans.x = ans.y = 0.0;
303         double area = 0.0;
304         for(int i = 0; i < n; i++)
305         {
306             int nt = (i + 1) % n;
307             double tp = pt[i].x * pt[nt].y - pt[nt].x *
                pt[i].y;
308             area += tp;
309             ans.x += tp * (pt[i].x + pt[nt].x);
310             ans.y += tp * (pt[i].y + pt[nt].y);
311         }

```

```

312     ans.x /= 3 * area;
313     ans.y /= 3 * area;
314     return ans;
315 }
316 //判断点是否在任意多边形内[射线法], O(n)
317 bool ahas( Point &_Off)
318 {
319     int ret = 0;
320     double infv = 1e20; //坐标系最大范围
321     Line l = Line( _Off, Point( -infv ,_Off.y));
322     for(int i = 0; i < n; i ++)
323     {
324         Line ln = Line( pt[i], pt[(i + 1) % n]);
325         if(fabs(ln.s.y - ln.e.y) > eps)
326         {
327             Point tp = (ln.s.y > ln.e.y)? ln.s: ln.e;
328             if( ( fabs( tp.y - _Off.y) < eps && tp.x
329                 < _Off.x + eps) || Line::isCrossSS(
330                     ln, l))
331             {
332                 ret++;
333             }
334             else if( Line::isCrossSS( ln, l))
335                 ret++;
336         }
337     }
338     return ret&1;
339 }
340 //判断任意点是否在凸包内, O(logn)
341 bool bhas( Point &p)
342 {
343     if( n < 3)
344         return false;
345     if( xmult( pt[0], p, pt[1]) > eps)
346         return false;
347     if( xmult( pt[0], p, pt[n-1]) < -eps)
348         return false;
349     int l = 2, r = n-1;
350     int line = -1;
351     while( l <= r)
352     {
353         int mid = ( l + r) >> 1;
354         if( xmult( pt[0], p, pt[mid]) >= 0)
355             line = mid, r = mid - 1;
356         else l = mid + 1;
357     }
358     return xmult( pt[line-1], p, pt[line]) <= eps;
359 }
360 //凸多边形被直线分割
361 Polygon split( Line &_Off)
362 {
363     //注意确保多边形能被分割
364     Polygon ret;
365     Point spt[2];
366     double tp = 0.0, np;
367     bool flag = true;
368     int i, pn = 0, spn = 0;
369     for(i = 0; i < n; i ++)
370     {
371         if(flag)
372             pt[pn++] = pt[i];
373         else
374             ret.pt[ret.n++] = pt[i];
375         np = xmult( _Off.s, _Off.e, pt[(i + 1) % n]);
376         if(tp * np < -eps)
377         {
378             flag = !flag;
379             Line::crossLPt( _Off, Line(pt[i], pt[(i +
380                 1) % n]), spt[spn++]);
381         }
382         tp = (fabs(np) > eps)?np: tp;
383     }
384     ret.pt[ret.n++] = spt[0];
385     ret.pt[ret.n++] = spt[1];
386     n = pn;
387     return ret;
388 }
389 /** 卷包裹法求点集凸包, _p为输入点集, _n为点的数量 **/
390 void ConvexClosure( Point _p[], int _n)
391 {
392     sort( _p, _p + _n);
393     n = 0;
394     for(int i = 0; i < _n; i++)
395     {
396         while( n > 1 && sgn( xmult( pt[n-2], pt[n-1],
397             _p[i]), 0) <= 0)
398             n--;
399         pt[n++] = _p[i];
400     }
401     int _key = n;
402     for(int i = _n - 2; i >= 0; i--)
403     {

```

```

404         while( n > _key && sgn( xmult( pt[n-2], pt[n-1], _p[i]), 0) <= 0)
405             n--;
406         pt[n++] = _p[i];
407     }
408     if(n>1) n--; //除去重复的点, 该点已是凸包凸包起点
409 }
410 /***** 寻找凸包的graham 扫描法*****/
411 /***** _p为输入的点集, _n为点的数量*****/
412
413 void graham( Point _p[], int _n)
414 {
415     int cur=0;
416     for(int i = 1; i < _n; i++)
417         if( sgn( _p[cur].y, _p[i].y) > 0 || ( sgn( _p[
418             cur].y, _p[i].y) == 0 && sgn( _p[cur].x, _p[
419                 i].x) > 0) )
420             cur = i;
421     swap( _p[cur], _p[0]);
422     n = 0, gsort = pt[n++] = _p[0];
423     if( _n <= 1) return;
424     sort( _p + 1, _p+_n, gcmp);
425     pt[n++] = _p[1];
426     for(int i = 2; i < _n; i++)
427     {
428         while(n>1 && sgn( xmult( pt[n-2], pt[n-1], _p
429             [i]), 0) <= 0) // 当凸包退化成直线时需特别
430             注意n
431         n--;
432         pt[n++] = _p[i];
433     }
434 }
435 //凸包旋转卡壳(注意点必须顺时针或逆时针排列)
436 //返回值凸包直径的平方(最远两点距离的平方)
437 pair<Point,Point> rotating_calipers()
438 {
439     int i = 1 % n;
440     double ret = 0.0;
441     pt[n] = pt[0];
442     pair<Point,Point> ans=make_pair(pt[0],pt[0]);
443     for(int j = 0; j < n; j++)
444     {
445         while( fabs( xmult( pt[i+1], pt[j], pt[j +
446             1])) > fabs( xmult( pt[i], pt[j], pt[j +
447                 1])) + eps)
448             i = (i + 1) % n;
449         //pt[i]和pt[j],pt[i + 1]和pt[j + 1]可能是对踵
450         点
451         if(ret < getdis2(pt[i],pt[j])) ret = getdis2(
452             pt[i],pt[j]), ans = make_pair(pt[i],pt[j
453             ]);
454         if(ret < getdis2(pt[i+1],pt[j+1])) ret =
455             getdis(pt[i+1],pt[j+1]), ans = make_pair(
456                 pt[i+1],pt[j+1]);
457     }
458     return ans;
459 }
460 //凸包旋转卡壳(注意点必须逆时针排列)
461 //返回值两凸包的最短距离
462 double rotating_calipers( Polygon &_Off)
463 {
464     int i = 0;
465     double ret = 1e10; //inf
466     pt[n] = pt[0];
467     _Off.pt[_Off.n] = _Off.pt[0];
468     //注意凸包必须逆时针排列且pt[0]是左下角点的位置
469     while( _Off.pt[i + 1].y > _Off.pt[i].y)
470         i = (i + 1) % _Off.n;
471     for(int j = 0; j < n; j++)
472     {
473         double tp;
474         //逆时针时为 >, 顺时针则相反
475         while((tp = xmult(_Off.pt[i + 1], pt[j], pt[j +
476             1]) - xmult(_Off.pt[i], pt[j], pt[j +
477                 1])) > eps)
478             i = (i + 1) % _Off.n;
479         //((pt[i],pt[i+1])和(_Off.pt[j],_Off.pt[j +
480             1])可能是最近线段
481         ret = min(ret, Line(pt[j], pt[j + 1]).dis(
482             _Off.pt[i], true));
483         ret = min(ret, Line(_Off.pt[i], _Off.pt[i +
484             1]).dis(pt[j + 1], true));
485         if(tp > -eps) //如果不考虑TLE问题最好不要加这个
486             判断
487         {
488             ret = min(ret, Line(pt[j], pt[j + 1]).dis
489                 (_Off.pt[i + 1], true));
490             ret = min(ret, Line(_Off.pt[i], _Off.pt[i
491                 + 1]).dis(pt[j], true));
492         }
493     }
494     return ret;
495 }
496 //-----半平面交-----

```

```

480 //复杂度:O(nlog2(n))
481 //获取半平面交的多边形 (多边形的核)
482 //参数: 向量集合[l], 向量数量[ln];(半平面方向在向量左
    边)
483 //函数运行后如果n[即返回多边形的点数量]为0则不存在半平
    面交的多边形 (不存在区域或区域面积无穷大)
484 int judge( Line &_lx, Line &_ly, Line &_lz)
485 {
486     Point tmp;
487     Line::crossLPt(_lx,_ly,tmp);
488     return sgn(xmult(_lz.s,tmp,_lz.e),0);
489 }
490 int halfPanelCross(Line L[], int ln)
491 {
492     int i, tn, bot, top;
493     for(int i = 0; i < ln; i++)
494         L[i].get_angle();
495     sort(L, L + ln);
496     //平面在向量左边的筛选
497     for(i = tn = 1; i < ln; i++)
498         if(fabs(L[i].angle - L[i - 1].angle) > eps)
499             L[tn++] = L[i];
500     ln = tn, n = 0, bot = 0, top = 1;
501     dq[0] = L[0], dq[1] = L[1];
502     for(i = 2; i < ln; i++)
503     {
504         while(bot < top && judge(dq[top],dq[top-1],L
            [i]) > 0)
505             top--;
506         while(bot < top && judge(dq[bot],dq[bot+1],L
            [i]) > 0)
507             bot++;
508         dq[++top] = L[i];
509     }
510     while(bot < top && judge(dq[top],dq[top-1],dq[
        bot]) > 0)
511         top--;
512     while(bot < top && judge(dq[bot],dq[bot+1],dq[
        top]) > 0)
513         bot++;
514     //若半平面交退化为点或线
515     // if(top <= bot + 1)
516     // return 0;
517     dq[++top] = dq[bot];
518     for(i = bot; i < top; i++)
519         Line::crossLPt(dq[i],dq[i + 1],pt[n++]);
520     return n;
521 }

522 };
523
524
525 class Circle
526 {
527 public:
528     Point c;//圆心
529     double r;//半径
530     double db, de;//圆弧度数起点, 圆弧度数终点(逆时针0
        -360)
531
532     //-----圆-----
533
534     //判断圆在多边形内
535     bool inside( Polygon &_Off)
536     {
537         if(_Off.ahas(c) == false)
538             return false;
539         for(int i = 0; i < _Off.n; i++)
540         {
541             Line l = Line(_Off.pt[i], _Off.pt[(i + 1) %
                _Off.n]);
542             if(l.dis(c, true) < r - eps)
543                 return false;
544         }
545         return true;
546     }
547
548     //判断多边形在圆内 (线段和折线类似)
549     bool has( Polygon &_Off)
550     {
551         for(int i = 0; i < _Off.n; i++)
552             if( getdis2(_Off.pt[i],c) > r * r - eps)
553                 return false;
554             return true;
555     }
556
557     //-----圆弧-----
558     //圆被其他圆截得的圆弧, 参数: 圆[_Off]
559     Circle operator-(Circle &_Off) const
560     {
561         //注意圆必须相交, 圆心不能重合
562         double d2 = getdis2(c,_Off.c);
563         double d = getdis(c,_Off.c);
564         double ans = acos((d2 + r * r - _Off.r * _Off.r)
            / (2 * d * r));
565         Point py = _Off.c - c;
566         double oans = atan2(py.y, py.x);

```



```

567     Circle res;
568     res.c = c;
569     res.r = r;
570     res.db = oans + ans;
571     res.de = oans - ans + 2 * PI;
572     return res;
573 }
574 //圆被其他圆截得的圆弧, 参数: 圆[_Off]
575 Circle operator+(Circle &_Off) const
576 {
577     //注意圆必须相交, 圆心不能重合
578     double d2 = getdis2(c, _Off.c);
579     double d = getdis(c, _Off.c);
580     double ans = acos((d2 + r * r - _Off.r * _Off.r)
581         / (2 * d * r));
582     Point py = _Off.c - c;
583     double oans = atan2(py.y, py.x);
584     Circle res;
585     res.c = c;
586     res.r = r;
587     res.db = oans - ans;
588     res.de = oans + ans;
589     return res;
590 }
591 //过圆外一点的两条切线
592 //参数: 点[_Off](必须在圆外), 返回: 两条切线(切线的s点
593 //      为_Off, e点为切点)
594 pair<Line, Line> tangent( Point &_Off)
595 {
596     double d = getdis(c, _Off);
597     //计算角度偏移的方式
598     double angp = acos(r / d), angc = atan2(_Off.y -
599         c.y, _Off.x - c.x);
600     Point pl = Point(c.x + r * cos(angc + angp), c.y
601         + r * sin(angc + angp)),
602     pr = Point(c.x + r * cos(angc - angp), c.y + r *
603         sin(angc - angp));
604     return make_pair(Line(_Off, pl), Line(_Off, pr))
605         ;
606 }
607 //计算直线和圆的两个交点
608 //参数: 直线[_Off](两点式), 返回两个交点, 注意直线必须
609 //      和圆有两个交点
610 pair<Point, Point> cross(Line _Off)
611 {
612     _Off.pton();
613     //到直线垂足的距离
614     double td = fabs(_Off.a * c.x + _Off.b * c.y +
615         _Off.c) / sqrt(_Off.a * _Off.a + _Off.b *
616         _Off.b);
617     //计算垂足坐标
618     double xp = (_Off.b * _Off.b * c.x - _Off.a *
619         _Off.b * c.y - _Off.a * _Off.c) / (_Off.a *
620         _Off.a + _Off.b * _Off.b);
621     double yp = (- _Off.a * _Off.b * c.x + _Off.a *
622         _Off.a * c.y - _Off.b * _Off.c) / (_Off.a *
623         _Off.a + _Off.b * _Off.b);
624     double angc = atan2(yp - c.y, xp - c.x);
625     double angp = acos(td / r);
626     return make_pair(Point(c.x + r * cos(angc + angp
627         ), c.y + r * sin(angc + angp)),
628         Point(c.x + r * cos(angc - angp), c.y + r * sin(
629         angc - angp)));
630 }
631 };
632
633 class triangle
634 {
635 public:
636     Point a, b, c; //顶点
637     triangle(){}
638     triangle(Point a, Point b, Point c): a(a), b(b), c(c
639         ){}
640     //计算三角形面积
641     double area()
642     {
643         return fabs( xmult(a, b, c)) / 2.0;
644     }
645     //计算三角形外心
646     //返回: 外接圆圆心
647     Point circumcenter()
648     {
649         double pa = a.x * a.x + a.y * a.y;
650         double pb = b.x * b.x + b.y * b.y;
651         double pc = c.x * c.x + c.y * c.y;
652         double ta = pa * ( b.y - c.y) - pb * ( a.y - c.y
653             ) + pc * ( a.y - b.y);
654         double tb = -pa * ( b.x - c.x) + pb * ( a.x - c.
655             x) - pc * ( a.x - b.x);

```

```

645     double tc = a.x * ( b.y - c.y) - b.x * ( a.y - c
646         .y) + c.x * ( a.y - b.y);
647     return Point( ta / 2.0 / tc, tb / 2.0 / tc);
648 }
649 //计算三角形内心
650 //返回： 内接圆圆心
651 Point incenter()
652 {
653     Line u, v;
654     double m, n;
655     u.s = a;
656     m = atan2(b.y - a.y, b.x - a.x);
657     n = atan2(c.y - a.y, c.x - a.x);
658     u.e.x = u.s.x + cos((m + n) / 2);
659     u.e.y = u.s.y + sin((m + n) / 2);
660     v.s = b;
661     m = atan2(a.y - b.y, a.x - b.x);
662     n = atan2(c.y - b.y, c.x - b.x);
663     v.e.x = v.s.x + cos((m + n) / 2);
664     v.e.y = v.s.y + sin((m + n) / 2);
665     Point ret;
666     Line::crossLPt(u,v,ret);
667     return ret;
668 }
669 //计算三角形垂心
670 //返回： 高的交点
671 Point perpencenter()
672 {
673     Line u,v;
674     u.s = c;
675     u.e.x = u.s.x - a.y + b.y;
676     u.e.y = u.s.y + a.x - b.x;
677     v.s = b;
678     v.e.x = v.s.x - a.y + c.y;
679     v.e.y = v.s.y + a.x - c.x;
680     Point ret;
681     Line::crossLPt(u,v,ret);
682     return ret;
683 }
684 //计算三角形重心
685 //返回： 重心
686 //到三角形三顶点距离的平方和最小的点
687 //三角形内到三边距离之积最大的点
688 Point barycenter()
689 {
690
691
692     Line u,v;
693     u.s.x = (a.x + b.x) / 2;
694     u.s.y = (a.y + b.y) / 2;
695     u.e = c;
696     v.s.x = (a.x + c.x) / 2;
697     v.s.y = (a.y + c.y) / 2;
698     v.e = b;
699     Point ret;
700     Line::crossLPt(u,v,ret);
701     return ret;
702 }
703 //计算三角形费马点
704 //返回： 到三角形三顶点距离之和最小的点
705 Point fermentPoint()
706 {
707     Point u, v;
708     double step = fabs(a.x) + fabs(a.y) + fabs(b.x)
709         + fabs(b.y) + fabs(c.x) + fabs(c.y);
710     int i, j, k;
711     u.x = (a.x + b.x + c.x) / 3;
712     u.y = (a.y + b.y + c.y) / 3;
713     while (step > eps)
714     {
715         for (k = 0; k < 10; step /= 2, k++)
716         {
717             for (i = -1; i <= 1; i++)
718             {
719                 for (j = -1; j <= 1; j++)
720                 {
721                     v.x = u.x + step * i;
722                     v.y = u.y + step * j;
723                     if (getdis(u,a) + getdis(u,b) +
724                         getdis(u,c) > getdis(v,a) +
725                         getdis(v,b) + getdis(v,c))
726                         u = v;
727                 }
728             }
729         }
730     }
731     return u;
732 };
733 int main(void)
734 {
735
736     return 0;

```

737 }

6 杂项

6.1 define

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  using LL = long long;
4  #define FOR(i, x, y) for (decay<decltype(y)>::type i = (x), _##i = (y); i < _##i; ++i)
5  #define FORD(i, x, y) for (decay<decltype(x)>::type i = (x), _##i = (y); i > _##i; --i)

```

6.2 数位 dp

```

1  LL dfs(LL base, LL pos, LL len, LL s, bool limit) {
2      if (pos == -1) return s ? base : 1;
3      if (!limit && dp[base][pos][len][s] != -1) return dp[base][pos][len][s];
4      LL ret = 0;
5      LL ed = limit ? a[pos] : base - 1;
6      for(int i = 0; i < ed + 1; i++) {
7          tmp[pos] = i;
8          if (len == pos)
9              ret += dfs(base, pos - 1, len - (i == 0), s, limit && i == a[pos]);
10         else if (s && pos < (len + 1) / 2)
11             ret += dfs(base, pos - 1, len, tmp[len - pos] == i, limit && i == a[pos]);
12         else
13             ret += dfs(base, pos - 1, len, s, limit && i == a[pos]);
14     }
15     if (!limit) dp[base][pos][len][s] = ret;
16     return ret;
17 }
18
19 LL solve(LL x, LL base) {
20     LL sz = 0;
21     while (x) {
22         a[sz++] = x % base;
23         x /= base;
24     }
25     return dfs(base, sz - 1, sz - 1, 1, true);
26 }

```

6.3 fastio

```

1  namespace fastIO{
2      #define BUF_SIZE 100000
3      #define OUT_SIZE 100000

```

```

4 //fread->read
5 bool IOerror=0;
6 //inline char nc(){char ch=getchar();if(ch==-1)IOerror=1;return ch;}
7 inline char nc(){
8     static char buf[BUF_SIZE],*p1=buf+BUF_SIZE,*pend=buf+BUF_SIZE;
9     if(p1==pend){
10         p1=buf;pend=buf+fread(buf,1,BUF_SIZE,stdin);
11         if(pend==p1){IOerror=1;return -1;}
12     }
13     return *p1++;
14 }
15 inline bool blank(char ch){return ch==' '||ch=='\n' ||ch=='\r' ||ch=='\t';}
16 template<class T> inline bool read(T &x){
17     bool sign=0;char ch=nc();x=0;
18     for(;blank(ch);ch=nc());
19     if(IOerror)return false;
20     if(ch=='-')sign=1,ch=nc();
21     for(;ch>='0'&&ch<='9';ch=nc())x=x*10+ch-'0';
22     if(sign)x=-x;
23     return true;
24 }
25 inline bool read(double &x){
26     bool sign=0;char ch=nc();x=0;
27     for(;blank(ch);ch=nc());
28     if(IOerror)return false;
29     if(ch=='-')sign=1,ch=nc();
30     for(;ch>='0'&&ch<='9';ch=nc())x=x*10+ch-'0';
31     if(ch=='.'){
32         double tmp=1; ch=nc();
33         for(;ch>='0'&&ch<='9';ch=nc())tmp/=10.0,x+=tmp*(ch-'0');
34     }
35     if(sign)x=-x;
36     return true;
37 }
38 inline bool read(char *s){
39     char ch=nc();
40     for(;blank(ch);ch=nc());
41     if(IOerror)return false;
42     for(;!blank(ch)&&!IOerror;ch=nc())*s++=ch;
43     *s=0;
44     return true;
45 }
46 inline bool read(char &c){
47     for(c=nc();blank(c);c=nc());
48     if(IOerror){c=-1;return false;}
49     return true;
50 }
51 template<class T,class... U>bool read(T& h,U&... t){return read(h)&&read(t...);}

```

```

52     #undef OUT_SIZE
53     #undef BUF_SIZE
54 };
55 using namespace fastIO;

```

6.4 date

```

1  /*
2  zeller返回星期几%7
3  */
4  int zeller(int y,int m,int d) {
5      if (m<=2) y--,m+=12; int c=y/100; y%=100;
6      int w=((c>>2)-(c<<1)+y+(y>>2)+(13*(m+1)/5)+d-1)%7;
7      if (w<0) w+=7; return(w);
8  }
9  /*
10  用于计算天数
11  */
12  int getId(int y, int m, int d) {
13      if (m < 3) {y --; m += 12;}
14      return 365 * y + y / 4 - y / 100 + y / 400 + (153 * m + 2) / 5 + d;
15  }

```

6.5 常用概念

6.6 欧拉路径

欧拉回路：每条边恰走一次的回路

欧拉通路：每条边恰走一次的路径

欧拉图：存在欧拉回路的图

半欧拉图：存在欧拉通路的图

有向欧拉图：每个点入度 = 出度

无向欧拉图：每个点度数为偶数

有向半欧拉图：一个点入度 = 出度 + 1，一个点入度 = 出度 - 1，其他点入度 = 出度

无向半欧拉图：两个点度数为奇数，其他点度数为偶数

6.7 映射

[injective] or [one-to-one] 函数值不重复

[surjective] or [onto] 值域都被取到

[bijective] or [one-to-one correspondence] 一一对应

6.8 反演

反演中心 O ，反演半径 r ，点 p 的反演点 p' 满足 $|OP||OP'| = r^2$

不经过反演中心的直线，反形为经过反演中心的圆

不经过反演中心的圆，反形为圆，反演中心为这两个互为反形的圆的位似中心

6.9 弦图

设 $next(v)$ 表示 $N(v)$ 中最前的点. 令 $w*$ 表示所有满足 $A \in B$ 的 w 中最后的一个点, 判断 $v \cup N(v)$ 是否为极大团, 只需判断是否存在一个 $w \in w*$, 满足 $Next(w) = v$ 且 $|N(v)| + 1 \leq |N(w)|$ 即可.

6.10 五边形数

$$\prod_{n=1}^{\infty} (1 - x^n) = \sum_{n=0}^{\infty} (-1)^n (1 - x^{2n+1}) x^{n(3n+1)/2}$$

6.11 pick 定理

整多边形面积 $A =$ 内部格点数 $i +$ 边上格点数 $\frac{b}{2} - 1$

6.12 重心

半径为 r , 圆心角为 θ 的扇形重心与圆心的距离为 $\frac{4r \sin(\theta/2)}{3\theta}$

半径为 r , 圆心角为 θ 的圆弧重心与圆心的距离为 $\frac{4r \sin^3(\theta/2)}{3(\theta - \sin(\theta))}$

6.13 曼哈顿距离与切比雪夫距离

曼哈顿距离:

$$dis = |x1 - x2| + |y1 - y2|$$

切比雪夫距离:

$$dis = \max(|x1 - x2|, |y1 - y2|)$$

manhattan to chebyshev

$$(x, y) \rightarrow (x + y, x - y)$$

chebyshev to manhattan

$$(x, y) \rightarrow (\frac{x+y}{2}, \frac{x-y}{2})$$

6.14 第二类 Bernoulli number

$$B_m = 1 - \sum_{k=0}^{m-1} \binom{m}{k} \frac{B_k}{m-k+1}$$

$$S_m(n) = \sum_{k=1}^n k^m = \frac{1}{m+1} \sum_{k=0}^m \binom{m+1}{k} B_k n^{m+1-k}$$

6.15 Fibonacci 数

$$F_n = \frac{\varphi^n - (-\varphi)^{-n}}{\sqrt{5}}, \varphi = \frac{1+\sqrt{5}}{2}$$

$$F_n = \lfloor \frac{\varphi^n}{\sqrt{5}} + \frac{1}{2} \rfloor$$

6.16 Catalan 数

$$C_{n+1} = \frac{2(2n+1)}{n+2} C_n$$

$$C_n = \frac{1}{n+1} \binom{2n}{n} = \frac{(2n)!}{(n+1)!n!}$$

前 20 项: 1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012, 742900, 2674440, 9694845, 35357670, 129644790, 477638700, 1767263190

所有的奇卡特兰数 C_n 都满足 $n = 2^k - 1$ 。所有其他的卡特兰数都是偶数

6.17 Lucas 定理

$$C(n, m) \bmod p = C(n \bmod p, m \bmod p) * C(n/p, m/p), p \text{ 是质数}$$

6.18 扩展 Lucas 定理

若 p 不是质数，将 p 分解质因数后分别求解，再用中国剩余定理合并

6.19 BEST theorem

有向图中欧拉回路的数量 $ec(G) = t_w(G) \prod_{v \in V} (\deg(v) - 1)!$.

其中 $\deg(v)$ 表示 v 的入度， $tw(G)$ 表示以 w 为根的外向树的数量，且在连通欧拉图中以任一点为根的外向树数量相同

若需要定起点，则答案乘上 $\deg(s)$ ，表示对每一条欧拉回路， s 出现了 $\deg(s)$ 次，选取一个点切开得到一条从 s 出发的欧拉回路

6.20 欧拉示性数定理

对平面图 $V - E + F = 2$

6.21 最值反演 (MinMax 容斥)

$$\max\{S\} = \sum_{T \subseteq S} (-1)^{|T|-1} \min\{T\}$$

扩展到期望

$$E[\max\{S\}] = \sum_{T \subseteq S} (-1)^{|T|-1} E[\min\{T\}]$$

6.22 Polya 定理

设对 n 个对象用 m 种颜色: b_1, b_2, \dots, b_m 着色。

设 $m^{c(p_i)} = (b_1 + b_2 + \dots + b_m)^{c_1(p_i)} (b_1^2 + b_2^2 + \dots + b_m^2)^{c_2(p_i)} \dots (b_1^n + b_2^n + \dots + b_m^n)^{c_n(p_i)}$ ，其中 $c_j(p_i)$ 表示置换群中第 i 个置换循环长度为 j 的个数。

设 $S_k = (b_1^k + b_2^k + \dots + b_m^k), k = 1, 2, \dots, n$ ，则波利亚计数定理的母函数形式为： $P(G) = \frac{1}{|G|} \sum_{j=1}^g \Pi_{k=1}^n S_k^{c_k(p_j)}$

6.23 Stirling 数

第一类: n 个元素的项目分作 k 个环排列的方法数目

$$s(n, k) = (-1)^{n+k} |s(n, k)|$$

$$|s(n, 0)| = 0$$

$$|s(1, 1)| = 1$$

$$|s(n, k)| = |s(n-1, k-1)| + (n-1) * |s(n-1, k)|$$

第二类: n 个元素的集定义 k 个等价类的方法数

$$S(n, 1) = S(n, n) = 1$$

$$S(n, k) = S(n-1, k-1) + k * S(n-1, k)$$

6.24 常用排列组合公式

$\sum_{i=1}^n x_i = k, x_i \geq 0$ 的解数为 $C(n+k-1, n-1)$

$x_1 \geq 0, x_i \leq x_{i+1}, x_n \leq k-1$ 的解数等价于在 $[0, k-1]$ 共 k 个数中可重复的取 n 个数的组合数，为 $C(n+k-1, n)$

6.25 三角公式

$$\sin(a \pm b) = \sin a \cos b \pm \cos a \sin b$$

$$\cos(a \pm b) = \cos a \cos b \mp \sin a \sin b$$

$$\tan(a \pm b) = \frac{\tan(a) \pm \tan(b)}{1 \mp \tan(a) \tan(b)}$$

$$\tan(a) \pm \tan(b) = \frac{\sin(a \pm b)}{\cos(a) \cos(b)}$$

$$\sin(a) + \sin(b) = 2 \sin\left(\frac{a+b}{2}\right) \cos\left(\frac{a-b}{2}\right)$$

$$\begin{aligned}
\sin(a) - \sin(b) &= 2 \cos\left(\frac{a+b}{2}\right) \sin\left(\frac{a-b}{2}\right) \\
\cos(a) + \cos(b) &= 2 \cos\left(\frac{a+b}{2}\right) \cos\left(\frac{a-b}{2}\right) \\
\cos(a) - \cos(b) &= -2 \sin\left(\frac{a+b}{2}\right) \sin\left(\frac{a-b}{2}\right) \\
\sin(na) &= n \cos^{n-1} a \sin a - \binom{n}{3} \cos^{n-3} a \sin^3 a + \binom{n}{5} \cos^{n-5} a \sin^5 a - \dots \\
\cos(na) &= \cos^n a - \binom{n}{2} \cos^{n-2} a \sin^2 a + \binom{n}{4} \cos^{n-4} a \sin^4 a - \dots
\end{aligned}$$

6.26 积分表

$$\begin{aligned}
&== \text{含有 } ax+b \text{ 的积分} == \\
&\int (ax+b)^n dx = \frac{(ax+b)^{n+1}}{a(n+1)} + C \\
&\int \frac{1}{ax+b} dx = \frac{1}{a} \ln|ax+b| + C \\
&\int \frac{x}{ax+b} dx = \frac{1}{a^2} (ax+b - b \ln|ax+b|) + C \\
&\int \frac{x^2}{ax+b} dx = \frac{1}{2a^3} [(ax+b)^2 - 4b(ax+b) + 2b^2 \ln|ax+b|] + C \\
&\int \frac{1}{x(ax+b)} dx = -\frac{1}{b} \ln\left|\frac{ax+b}{x}\right| + C \\
&\int \frac{1}{x^2(ax+b)} dx = \frac{a}{b^2} \ln\left|\frac{ax+b}{x}\right| - \frac{1}{bx} + C \\
&== \text{含有 } \sqrt{a+bx} \text{ 的积分} == \\
&\int x\sqrt{a+bx} dx = \frac{2}{15b^2} (3bx-2a)(a+bx)^{\frac{3}{2}} + C \\
&\int x^2\sqrt{a+bx} dx = \frac{2}{105b^3} (15b^2x^2 - 12abx + 8a^2)(a+bx)^{\frac{3}{2}} + C \\
&\int x^n\sqrt{a+bx} dx = \frac{2}{b(2n+3)} x^n(a+bx)^{\frac{3}{2}} - \frac{2na}{b(2n+3)} \int x^{n-1}\sqrt{a+bx} dx \\
&\int \frac{\sqrt{a+bx}}{x} dx = 2\sqrt{a+bx} + a \int \frac{1}{x\sqrt{a+bx}} dx \\
&\int \frac{\sqrt{a+bx}}{x^n} dx = \frac{-1}{a(n-1)} \frac{(a+bx)^{\frac{3}{2}}}{x^{n-1}} - \frac{(2n-5)b}{2a(n-1)} \int \frac{\sqrt{a+bx}}{x^{n-1}} dx, n \neq 1 \\
&\int \frac{1}{x\sqrt{a+bx}} dx = \frac{1}{\sqrt{a}} \ln\left(\frac{\sqrt{a+bx}-\sqrt{a}}{\sqrt{a+bx}+\sqrt{a}}\right) + C, a > 0 = \frac{2}{\sqrt{-a}} \arctan \sqrt{\frac{a+bx}{-a}} + C, a < 0 \\
&\int \frac{1}{x^n\sqrt{a+bx}} dx = \frac{-1}{a(n-1)} \frac{\sqrt{a+bx}}{x^{n-1}} - \frac{(2n-3)b}{2a(n-1)} \int \frac{1}{x^{n-1}} \sqrt{a+bx} dx, n \neq 1 \\
&== \text{含有 } x^2 \pm \alpha^2 \text{ 的积分} == \\
&\int \frac{1}{x^2+\alpha^2} dx = \frac{\arctan \frac{x}{\alpha}}{\alpha} + C \\
&\int \frac{1}{\pm x^2 \mp \alpha^2} dx = \frac{\ln\left(\frac{x \mp \alpha}{\pm x + \alpha}\right)}{2\alpha} + C \\
&== \text{含有 } ax^2+b \text{ 的积分} == \\
&\int \frac{1}{ax^2+b} dx = \frac{1}{\sqrt{ab}} \arctan \frac{\sqrt{ax}}{\sqrt{b}} + C \\
&== \text{含有 } ax^2+bx+c \quad (a>0) \text{ 的积分} == \\
&\int ax^2+bx+cdx = \frac{ax^3}{3} + \frac{bx^2}{2} + cx + C \\
&== \text{含有 } \sqrt{a^2+x^2} \quad (a>0) \text{ 的积分} == \\
&\int \sqrt{a^2+x^2} dx = \frac{1}{2} x\sqrt{a^2+x^2} + \frac{1}{2} a^2 \ln(x+\sqrt{a^2+x^2}) + C \\
&\int x^2\sqrt{a^2+x^2} dx = \frac{1}{8} x(a^2+2x^2)\sqrt{a^2+x^2} - \frac{1}{8} a^4 \ln(x+\sqrt{a^2+x^2}) + C \\
&\int \frac{\sqrt{a^2+x^2}}{x} dx = \sqrt{a^2+x^2} - a \ln\left(\frac{a+\sqrt{a^2+x^2}}{x}\right) + C \\
&\int \frac{\sqrt{a^2+x^2}}{x^2} dx = \ln(x+\sqrt{a^2+x^2}) - \frac{\sqrt{a^2+x^2}}{x} + C \\
&\int \frac{1}{\sqrt{a^2+x^2}} dx = \ln(x+\sqrt{a^2+x^2}) + C \\
&\int \frac{x^2}{\sqrt{a^2+x^2}} dx = \frac{1}{2} x\sqrt{a^2+x^2} - \frac{1}{2} a^2 \ln(\sqrt{a^2+x^2}+x) + C \\
&\int \frac{1}{x\sqrt{a^2+x^2}} dx = \frac{1}{a} \ln\left(\frac{x}{a+\sqrt{a^2+x^2}}\right) + C \\
&\int \frac{1}{x^2\sqrt{a^2+x^2}} dx = -\frac{\sqrt{a^2+x^2}}{a^2x} + C \\
&== \text{含有 } \sqrt{x^2-a^2} \quad (x^2>a^2) \text{ 的积分} = \\
&\int \frac{1}{\sqrt{x^2-a^2}} dx = \ln|x+\sqrt{x^2-a^2}| + C \\
&== \text{含有 } \sqrt{a^2-x^2} \quad (a^2>x^2) \text{ 的积分} == \\
&\int \frac{1}{\sqrt{a^2-x^2}} dx = \arcsin \frac{x}{a} + C = -\arccos \frac{x}{a} + C \\
&\int \sqrt{a^2-x^2} dx = \frac{1}{2} x\sqrt{a^2-x^2} + \frac{a^2}{2} \arcsin \frac{x}{a} + C
\end{aligned}$$

$$\begin{aligned}
& \int x^2 \sqrt{a^2 - x^2} dx = \frac{1}{8} x (2x^2 - a^2) \sqrt{a^2 - x^2} + \frac{1}{8} a^4 \arcsin \frac{x}{a} + C \\
& \int \frac{\sqrt{a^2 - x^2}}{x} dx = \sqrt{a^2 - x^2} - a \ln \left(\frac{a + \sqrt{a^2 - x^2}}{x} \right) + C \\
& \int \frac{\sqrt{a^2 - x^2}}{x^2} dx = -\frac{\sqrt{a^2 - x^2}}{x} - \arcsin \frac{x}{a} + C \\
& \int \frac{1}{x \sqrt{a^2 - x^2}} dx = -\frac{1}{a} \ln \left(\frac{a + \sqrt{a^2 - x^2}}{x} \right) + C \\
& \int \frac{x^2}{\sqrt{a^2 - x^2}} dx = -\frac{1}{2} x \sqrt{a^2 - x^2} + \frac{1}{2} a^2 \arcsin \frac{x}{a} + C \\
& \int \frac{1}{x^2 \sqrt{a^2 - x^2}} dx = -\frac{\sqrt{a^2 - x^2}}{a^2 x} + C \\
& == \text{含有 } R = \sqrt{|a|x^2 + bx + c} \quad (a \neq 0) \text{ 的积分} == \\
& \int \frac{dx}{R} = \frac{1}{\sqrt{a}} \ln (2\sqrt{a}R + 2ax + b) \quad (\text{for } a > 0) \\
& \int \frac{dx}{R} = \frac{1}{\sqrt{a}} \operatorname{arsinh} \frac{2ax+b}{\sqrt{4ac-b^2}} \quad (\text{for } a > 0, 4ac - b^2 > 0) \\
& \int \frac{dx}{R} = \frac{1}{\sqrt{a}} \ln |2ax + b| \quad (\text{for } a > 0, 4ac - b^2 = 0) \\
& \int \frac{dx}{R} = -\frac{1}{\sqrt{-a}} \arcsin \frac{2ax+b}{\sqrt{b^2-4ac}} \quad (\text{for } a < 0, 4ac - b^2 < 0, (2ax + b) < \sqrt{b^2 - 4ac}) \\
& \int \frac{dx}{R^3} = \frac{4ax+2b}{(4ac-b^2)R} \\
& \int \frac{dx}{R^5} = \frac{4ax+2b}{3(4ac-b^2)R} \left(\frac{1}{R^2} + \frac{8a}{4ac-b^2} \right) \\
& \int \frac{dx}{R^{2n+1}} = \frac{2}{(2n-1)(4ac-b^2)} \left[\frac{2ax+b}{R^{2n-1}} + 4a(n-1) \int \frac{dx}{R^{2n-1}} \right] \\
& \int \frac{x}{R} dx = \frac{R}{a} - \frac{b}{2a} \int \frac{dx}{R} \\
& \int \frac{x}{R^3} dx = -\frac{2bx+4c}{(4ac-b^2)R} \\
& \int \frac{x}{R^{2n+1}} dx = -\frac{1}{(2n-1)aR^{2n-1}} - \frac{b}{2a} \int \frac{dx}{R^{2n+1}} \\
& \int \frac{dx}{xR} = -\frac{1}{\sqrt{c}} \ln \left(\frac{2\sqrt{c}R+bx+2c}{x} \right) \\
& \int \frac{dx}{xR} = -\frac{1}{\sqrt{c}} \operatorname{arsinh} \left(\frac{bx+2c}{|x|\sqrt{4ac-b^2}} \right) \\
& == \text{含有三角函数的积分} == \\
& \int \cos x dx = \sin x + C \\
& \int \sin x dx = -\cos x + C \\
& \int \sec^2 x dx = \tan x + C \\
& \int \csc^2 x dx = -\cot x + C \\
& \int \sec x \tan x dx = \sec x + C \\
& \int \csc x \cot x dx = -\csc x + C \\
& \int \tan x dx = -\ln |\cos x| + C = \ln |\sec x| + C \\
& \int \cot x dx = \ln |\sin x| + C \\
& \int \sec x dx = \ln |\sec x + \tan x| + C \\
& \int \csc x dx = -\ln |\csc x + \cot x| + C = \ln \left| \frac{\tan x - \sin x}{\sin x \tan x} \right| + C \\
& \int \sin^n x dx = -\frac{1}{n} \sin^{n-1} x \cos x + \frac{n-1}{n} \int \sin^{n-2} x dx + C \quad \forall n \geq 2 \\
& \int \sin^2 x dx = \frac{x}{2} - \frac{\sin 2x}{4} + C \\
& \int \cos^n x dx = \frac{1}{n} \cos^{n-1} x \sin x + \frac{n-1}{n} \int \cos^{n-2} x dx + C \quad \forall n \geq 2 \\
& \int \cos^2 x dx = \frac{x}{2} + \frac{\sin 2x}{4} + C \\
& \int \tan^n x dx = \frac{1}{n-1} \tan^{n-1} x - \int \tan^{n-2} x dx + C \quad \forall n \geq 2 \\
& \int \tan^2 x dx = \tan x - x + C \\
& \int \cot^n x dx = \frac{1}{n-1} \cot^{n-1} x - \int \cot^{n-2} x dx + C \quad \forall n \geq 2 \\
& \int \cot^2 x dx = -\cot x - x + C \\
& \int \sec^n x dx = \frac{1}{n-1} \sec^{n-2} x \tan x + \frac{n-2}{n-1} \int \sec^{n-2} x dx + C \quad \forall n \geq 2 \\
& \int \csc^n x dx = -\frac{1}{n-1} \csc^{n-2} x \cot x + \frac{n-2}{n-1} \int \csc^{n-2} x dx + C \quad \forall n \geq 2 \\
& == \text{含有反三角函数的积分} == \\
& \int \arcsin x dx = x \arcsin x + \sqrt{1-x^2} + C \\
& \int \arccos x dx = x \arccos x - \sqrt{1-x^2} + C \\
& \int \arctan x dx = x \arctan x - \ln \sqrt{1+x^2} + C
\end{aligned}$$

$$\int \operatorname{arccot}(x) dx = x \times \operatorname{arccot}(x) + \ln \sqrt{1+x^2} + C$$

$$\int \operatorname{arcsec}(x) dx = x \times \operatorname{arcsec}(x) - \operatorname{sgn}(x) \ln |x + \sqrt{x^2-1}| + C = x \times \operatorname{arcsec}(x) + \operatorname{sgn}(x) \ln |x - \sqrt{x^2-1}| + C$$

$$\int \operatorname{arccsc}(x) dx = x \times \operatorname{arccsc}(x) + \operatorname{sgn}(x) \ln |x + \sqrt{x^2-1}| + C = x \times \operatorname{arccsc}(x) - \operatorname{sgn}(x) \ln |x - \sqrt{x^2-1}| + C$$

== 含有指数函数的积分 ==

$$\int e^x dx = e^x + C$$

$$\int \alpha^x dx = \frac{\alpha^x}{\ln \alpha} + C$$

$$\int x e^{ax} dx = \frac{1}{a^2} (ax - 1) e^{ax} + C$$

$$\int x^n e^{ax} dx = \frac{1}{a} x^n e^{ax} - \frac{n}{a} \int x^{n-1} e^{ax} dx$$

$$\int e^{ax} \sin bx dx = \frac{e^{ax}}{a^2 + b^2} (a \sin bx - b \cos bx) + C$$

$$\int e^{ax} \cos bx dx = \frac{e^{ax}}{a^2 + b^2} (a \cos bx + b \sin bx) + C$$

== 含有对数函数的积分 ==

$$\int \ln x dx = x \ln x - x + C$$

$$\int \log_{\alpha} x dx = \frac{1}{\ln \alpha} (x \ln x - x) + C$$

$$\int x^n \ln x dx = \frac{x^{n+1}}{(n+1)^2} [(n+1) \ln x - 1] + C$$

$$\int \frac{1}{x \ln x} dx = \ln(\ln x) + C$$

== 含有双曲函数的积分 ==

$$\int \sinh x dx = \cosh x + C$$

$$\int \cosh x dx = \sinh x + C$$

$$\int \tanh x dx = \ln(\cosh x) + C$$

$$\int \coth x dx = \ln(\sinh x) + C$$

$$\int \operatorname{sech} x dx = \arcsin(\tanh x) + C = \arctan(\sinh x) + C$$

$$\int \operatorname{csch} x dx = \ln\left(\tanh \frac{x}{2}\right) + C$$

== 定积分 ==

$$\int_{-\infty}^{\infty} e^{-\alpha x^2} dx = \sqrt{\frac{\pi}{\alpha}}$$

$$\int_0^{\frac{\pi}{2}} \sin^n x dx = \int_0^{\frac{\pi}{2}} \cos^n x dx =$$

$$\begin{cases} \frac{n-1}{n} \cdot \frac{n-3}{n-2} \cdots \frac{4}{5} \cdot \frac{2}{3}, & \text{if } n > 1 \text{ 且 } n \text{ 为奇数} \\ \frac{n-1}{n} \cdot \frac{n-3}{n-2} \cdots \frac{3}{4} \cdot \frac{1}{2} \cdot \frac{\pi}{2}, & \text{if } n > 0 \text{ 且 } n \text{ 为偶数} \end{cases}$$