**后缀数组**

|  |
| --- |
| #define LL long long  #define ULL unsigned long long  using namespace std;  const int MAXN=400000;  //以下为倍增算法求后缀数组  int wa[MAXN],wb[MAXN],wv[MAXN],Ws[MAXN];  int cmp(int \*r,int a,int b,int l)  {  return r[a]==r[b]&&r[a+l]==r[b+l];  }  /\*\*< 传入参数：str,sa,len+1,ASCII\_MAX+1 \*/  void da(const char r[],int sa[],int n,int m)  {  int i,j,p,\*x=wa,\*y=wb,\*t;  for(i=0; i<m; i++) Ws[i]=0;  for(i=0; i<n; i++) Ws[x[i]=r[i]]++;//以字符的ascii码为下标  for(i=1; i<m; i++) Ws[i]+=Ws[i-1];  for(i=n-1; i>=0; i--) sa[--Ws[x[i]]]=i;  /\*cout<<"SA"<<endl;;  for(int i=0;i<n+1;i++)cout<<sa[i]<<' ';\*/  for(j=1,p=1; p<n; j\*=2,m=p)  {  for(p=0,i=n-j; i<n; i++) y[p++]=i;  for(i=0; i<n; i++) if(sa[i]>=j) y[p++]=sa[i]-j;  for(i=0; i<n; i++) wv[i]=x[y[i]];  for(i=0; i<m; i++) Ws[i]=0;  for(i=0; i<n; i++) Ws[wv[i]]++;  for(i=1; i<m; i++) Ws[i]+=Ws[i-1];  for(i=n-1; i>=0; i--) sa[--Ws[wv[i]]]=y[i];  for(t=x,x=y,y=t,p=1,x[sa[0]]=0,i=1; i<n; i++)  x[sa[i]]=cmp(y,sa[i-1],sa[i],j)?p-1:p++;  }  return;  }  int sa[MAXN],Rank[MAXN],height[MAXN];  //求height数组  /\*\*< str,sa,len \*/  void calheight(const char \*r,int \*sa,int n)  {  int i,j,k=0;  for(i=1; i<=n; i++) Rank[sa[i]]=i;  for(i=0; i<n; height[Rank[i++]]=k)  for(k?k--:0,j=sa[Rank[i]-1]; r[i+k]==r[j+k]; k++);  // Unified  for(int i=n; i>=1; --i) ++sa[i],Rank[i]=Rank[i-1];  }  char str[MAXN];  int main()  {  while(scanf("%s",str)!=EOF)  {  int len=strlen(str);  da(str,sa,len+1,130);  calheight(str,sa,len);  puts("--------------All Suffix--------------");  for(int i=1; i<=len; ++i)  {  printf("%d:\t",i);  for(int j=i-1; j<len; ++j)  printf("%c",str[j]);  puts("");  }  puts("");  puts("-------------After sort---------------");  for(int i=1; i<=len; ++i)  {  printf("sa[%2d ] = %2d\t",i,sa[i]);  for(int j=sa[i]-1; j<len; ++j)  printf("%c",str[j]);  puts("");  }  puts("");  puts("---------------Height-----------------");  for(int i=1; i<=len; ++i)  printf("height[%2d ]=%2d \n",i,height[i]);  puts("");  puts("----------------Rank------------------");  for(int i=1; i<=len; ++i)  printf("Rank[%2d ] = %2d\n",i,Rank[i]);  puts("------------------END-----------------");  }  return 0;  } |
|  |

**最长回文子串**

|  |
| --- |
| #include<iostream>  using namespace std;  const int MAXN=110010;  char Ma[MAXN\*2];  int Mp[MAXN\*2];  void Manacher(char s[],int len)  {  int l=0;  Ma[l++]='$';  Ma[l++]='#';  for(int i=0; i<len; i++)  {  Ma[l++]=s[i];  Ma[l++]='#';  }  Ma[l]=0;  int mx=0,id=0;  for(int i=0; i<l; i++)  {  Mp[i]=mx>i?min(Mp[2\*id-i],mx-i):1;  while(Ma[i+Mp[i]]==Ma[i-Mp[i]])Mp[i]++;  if(i+Mp[i]>mx)  {  mx=i+Mp[i];  id=i;  }  }  }  char s[MAXN];  int main()  {  while(scanf("%s",s)==1)  {  int len=strlen(s);  Manacher(s,len);  int ans=0;  for(int i=0; i<2\*len+2; i++) ans=max(ans,Mp[i]-1);  printf("%d\n",ans);  }  return 0;  } |

**AC自动机**

|  |
| --- |
| using namespace std;  struct Trie  {  int next[500010][26],fail[500010],end[500010];  int root,L;  int newnode()  {  for(int i = 0; i < 26; i++) next[L][i] = -1;  end[L++] = 0;  return L-1;  }  void init()  {  L = 0;  root = newnode();  }  void insert(char buf[])  {  int len = strlen(buf);  int now = root;  for(int i = 0; i < len; i++)  {  if(next[now][buf[i]-'a'] == -1)  next[now][buf[i]-'a'] = newnode();  now = next[now][buf[i]-'a'];  }  end[now]++;  }  void build()  {  queue<int>Q;  fail[root] = root;  for(int i = 0; i < 26; i++)  if(next[root][i] == -1) next[root][i] = root;  else  {  fail[next[root][i]] = root;  Q.push(next[root][i]);  }  while( !Q.empty() )  {  int now = Q.front();  Q.pop();  for(int i = 0; i < 26; i++)  if(next[now][i] == -1)  next[now][i] = next[fail[now]][i];  else  {  fail[next[now][i]]=next[fail[now]][i];  Q.push(next[now][i]);  }  }  }  int query(char buf[])  {  int len = strlen(buf);  int now = root;  int res = 0;  for(int i = 0; i < len; i++)  {  now = next[now][buf[i]-'a'];  int temp = now;  while( temp != root )  {  res += end[temp];  end[temp] = 0;  temp = fail[temp];  }  }  return res;  }  void debug()  {  for(int i = 0; i < L; i++)  {  printf("id = %3d,fail = %3d,end = %3d,chi = [",i,fail[i],end[i]);  for(int j = 0; j < 26; j++)  printf("%2d",next[i][j]);  printf("]\n");  }  }  };  char buf[1000010];  Trie ac;  int main()  {  int T;  int n;  scanf("%d",&T);  while( T-- )  {  scanf("%d",&n);  ac.init();  for(int i = 0; i < n; i++)  {  scanf("%s",buf);  ac.insert(buf);  }  ac.build();  scanf("%s",buf);  printf("%d\n",ac.query(buf));  }  return 0;  } |

**头文件**

|  |
| --- |
| #include<bits/stdc++.h>  #define \_CRT\_SBCURE\_NO\_DEPRECATE  #include <set>  #include <cmath>  #include <queue>  #include <stack>  #include <vector>  #include <string>  #include <cstdio>  #include <cstdlib>  #include <cstring>  #include <iostream>  #include <algorithm>  #include <functional> |

**次小生成树**

|  |
| --- |
| /\*  \* 次小生成树  \* 求最小生成树时，用数组Max[i][j]来表示MST中i到j最大边权  \* 求完后，直接枚举所有不在MST中的边，替换掉最大边权的边，更新答案  \* 点的编号从0开始  \*/  const int MAXN=110;  const int INF=0x3f3f3f3f;  bool vis[MAXN];  int lowc[MAXN];  int pre[MAXN];  int Max[MAXN][MAXN];//Max[i][j]表示在最小生成树中从i到j的路径中的最大边权  bool used[MAXN][MAXN];  int Prim(int cost[][MAXN],int n)  {  int ans=0;  memset(vis,false,sizeof(vis));  memset(Max,0,sizeof(Max));  memset(used,false,sizeof(used));  vis[0]=true;  pre[0]=-1;  for(int i=1; i<n; i++)  {  lowc[i]=cost[0][i];  pre[i]=0;  }  lowc[0]=0;  for(int i=1; i<n; i++)  {  int minc=INF;  int p=-1;  for(int j=0; j<n; j++)  if(!vis[j]&&minc>lowc[j])  {  minc=lowc[j];  p=j;  }  if(minc==INF)return -1;  ans+=minc;  vis[p]=true;  used[p][pre[p]]=used[pre[p]][p]=true;  for(int j=0; j<n; j++)  {  if(vis[j])Max[j][p]=Max[p][j]=max(Max[j][pre[p]],lowc[p]);  if(!vis[j]&&lowc[j]>cost[p][j])  {  lowc[j]=cost[p][j];  pre[j]=p;  }  }  }  return ans;  } |

**输入输出挂**

|  |
| --- |
| void read(int &x){  char ch = getchar();x = 0;  for (; ch < '0' || ch > '9'; ch = getchar());  for (; ch >='0' && ch <= '9'; ch = getchar()) x = x \* 10 + ch - '0';  }  void out(int a)  { // 输出外挂  if (a < 0)  {  putchar('-');  a = -a;  }  if (a >= 10)  {  out(a / 10);  }  putchar(a % 10 + '0');  } |

**树上任意两点距离和**

|  |
| --- |
| #include<iostream>  #include<stdio.h>  #include<string.h>  #include<vector>  #define ll long long  #define maxx 100010  #define mod 1000000007  using namespace std;  struct node  {  int v;  ll w;  };  int N;  ll jc[maxx];  ll sum[maxx],dp[maxx];  vector<node> tree[maxx];  void init()  {  jc[0]=1;  for(int i=1;i<maxx;i++)  {  jc[i]=(jc[i-1]\*i)%mod;  }  }  void dfs(int cur,int father)  {  sum[cur]=1;  for(int i=0;i<tree[cur].size();i++)  {  int son=tree[cur][i].v;  ll len=tree[cur][i].w;  if(father==son)  {  continue;  }  dfs(son,cur);  sum[cur]=(sum[cur]+sum[son])%mod;  dp[cur] = (dp[cur]+dp[son]+(N-sum[son])\*sum[son]\*len)%mod;  }  }  int main()  {  init();  while(scanf("%d",&N)!=EOF)  {  ll u,v,w;  for(int i = 0;i<=N; i++)  tree[i].clear();  memset(sum,0,sizeof(sum));  memset(dp,0,sizeof(dp));  for(int i=0;i<N-1;i++)  {  scanf("%lld%lld%lld",&u,&v,&w);  node t1,t2;  t1.v=v;  t1.w=w;  t2.v=u;  t2.w=w;  tree[u].push\_back(t1);  tree[v].push\_back(t2);  }  dfs(1,-1);  ll ans=dp[1]\*2%mod\*jc[N-1]%mod;  printf("%lld\n",ans);  }  return 0;  } |

**博弈论**

|  |
| --- |
| 巴什博弈：在箱子中有N个球，每次取1到M个，如果则先手胜，否则后手胜  尼姆博弈：有N堆石子，两个人可以同时取1到M堆石子中的任意个数的石子，将N堆石子转换为二进制数然后将每一位相加，如果每一位模以M+1都等于0那么先手输，否则先手赢（当然存在争议，因为没看懂原理）  威佐夫博弈：当某一人面对（0,0）局势时必败，设第k+1个必败局势为（,）,<=,则有如下性质： = (sqrt(5)+1)\*k/2, = +k  斐波那契博弈：有一堆个数为n的石子，游戏双方轮流取石子，满足：1)先手不能在第一次把所有的石子取完；2)之后每次可以取的石子数介于1到对手刚取的石子数的2倍之间（包含1和对手刚取的石子数的2倍）。当石子堆中的数量为斐波那契数时，先手必输，否则先手必赢。 |

**优先队列**

|  |
| --- |
| #include<queue>  priority\_queue <int> i;  priority\_queue <double> d;  q.size();//返回q里元素个数  q.empty();//返回q是否为空，空则返回1，否则返回0  q.push(k);//在q的末尾插入k  q.pop();//删掉q的第一个元素  q.top();//返回q的第一个元素  q.back();//返回q的末尾元素 |