基本字符串算法

KMP

`f[i]`为`s`以`i`结束的能作为`s`前缀的最长串长

```
void getf(const string& s, vector<int>& f) {
    int n = s.size(); f.resize(n, 0);
    for (int i = 1, j = 0; i != n; ++i) {
        while (j && s[i] != s[j]) j = f[j - 1];
        if (s[i] == s[j]) ++j;
        f[i] = j;
    }
}
```

`l[i]`为`t`以`i`结束的能作为`s`前缀的的最长串长

```
void getl(const string& s, const string& t, const vector<int>& f, vector<int>&
l) {
    int n = s.size(), m = t.size(); l.resize(m, 0);
    for (int i = 0, j = 0; i != m; ++i) {
        while(j && t[i] != s[j]) j = f[j - 1];
        if (t[i] == s[j]) ++j;
        l[i] = j;
        if (j == n) j = f[j - 1];
    }
}
```

EXKMP

`z[i]`为`s`以`i`为起始的能作为`s`前缀的最长串长

```
void getz(const string& s, vector<int>& z) {
    int n = s.size(); z.resize(n, 0);
    for (int i = 1, l = 0, r = 0; i != n; ++i) {
        int j = z[i - l];
        if (i+j>r) for (j=max(0, r-i); i+j!=n&&s[ia+j]==s[j]; ++j);
        z[i] = j; if (i+j-1>r) l=i, r=i+j-1;
    }
    z[0] = n;
}
```

`e[i]`为`t`以`i`起始的能作为`s`前缀的最长串长

```
void gete(const string& s, const string& t, const vector<int>& z, vector<int>&
e) {
    int n = s.size(), m = t.size(); e.resize(m, 0);
    for (int i = 0, l = 0, r = 0; i != m; ++i) {
        int j = z[i - l];
        if (i+j>r) for (j=max(0, r-i); i+j!=m&&t[i+j]==s[j]; ++j);
        e[i] = j; if (i+j-1>r) l=i, r=i+j-1;
    }
}
```

Manacher

`r[i]`为`t`串以`i`为中心的回文半径：`t[i+r[i]]==t[i-r[i]]`

```
void getr(const string& s, vector<int>& r) {
    int n = s.size(), m = 2 * n + 1; r.resize(m, 0);
    string t(m, '#');
    for (int i = 0; i != n; ++i) t[2 * i + 1] = s[i];
    for (int i = 1, p = 0, w = 0; i != m; ++i) {
        int j = (i>p+w ? 0 : min(r[p-(i-p)], p+w-i));
        while(0<=i-j-1 && i+j+1<m && t[i-j-1]==t[i+j+1]) ++j;
        r[i] = j; if (i + j > p + w)  p = i, w = j;
    }
}
```

判断区间`[l,u)`是否为回文

```
bool ispal(const vector<int>& r, int l, int u) {
    return r[l + u] >= u - l;
}
```

哈希

```
typedef long long ll;
typedef pair<ll, ll> pll;
const ll p1 = 1145140019, b1 = 893;
const ll p2 = 1919810021, b2 = 364;
ll hs1[N], hb1[N];
ll hs2[N], hb2[N];

char s[N]; int n;
void genh() {
    hb1[0] = hb2[0] = 1;
    for (int i = 1; i <= n; ++i) {
        hs1[i] = (hs1[i - 1] * b1 + s[i]) % p1;
        hb1[i] = hb1[i - 1] * b1 % p1;
        hs2[i] = (hs2[i - 1] * b2 + s[i]) % p2;
        hb2[i] = hb2[i - 1] * b2 % p2;
    }
}

pll geth(int l, int r) {
    ll v1 = (hs1[r] - hs1[l - 1] * hb1[r - l + 1]) % p1;
    ll v2 = (hs2[r] - hs2[l - 1] * hb2[r - l + 1]) % p2;
    if (v1 < 0) v1 += p1; if (v2 < 0) v2 += p2;
    return { v1, v2 };
}

pll concat(pll lh, int ln, pll rh, int rn) {
    pll r;
    r.first = (lh.first * hb1[rn] + rh.first) % p1;
    r.second = (lh.second * hb2[rn] + rh.second) % p2;
    return r;
}

int lcp(int u, int v) {
    if (s[u] != s[v]) return 0;
    int l = 1, r = n - max(u, v) + 1;
    while (l <= r) {
        int mid = (l + r) / 2;
        if (geth(u, u + mid - 1) == geth(v, v + mid - 1))
            l = mid + 1;
        else
            r = mid - 1;
    }
    return r;
}

int lcs(int u, int v) {
    if (s[u] != s[v]) return 0;
    int l = 1, r = min(u, v);
    while (l <= r) {
        int mid = (l + r) / 2;
        if (geth(u - mid + 1, u) == geth(v - mid + 1, v))
            l = mid + 1;
        else
            r = mid - 1;
    }
    return r;
}
```

自动机

AC自动机

```
int g[N][26], f[N], e[N], nc;

int gn() {
    int p = nc++; f[p] = e[p] = 0;
    memset(g[p], 0, sizeof(g[p]));
    return p;
}

void clr() { nc = 0; gn(); }

void ins(const string& s) {
    int p = 0;
    for (int i = 0; i != s.size(); ++i) {
        int o = s[i] - 'a';
        if (!g[p][o]) g[p][o] = gn();
        p = g[p][o];
    }
    e[p] = 1;
}

void build() {
    queue<int> q;
    for (int o = 0; o != 26; ++o)
        if (g[0][o]) q.push(g[0][o]);
    while(!q.empty()) {
        int u = q.front(); q.pop();
        if (e[f[u]]) e[u] = 1;
        for (int o = 0; o != 26; ++o) {
            int& v = g[u][o];
            if (!v) v = g[f[u]][o];
            else {
                f[v] = g[f[u]][o];
                q.push(v);
            }
        }
    }
}
```

后缀自动机

注：将注释去掉后即为（假的）广义SAM

```
char s[N]; int n;
int g[N][26], f[N], l[N], w[N], nc;

inline int gn(int len, int q = 0) {
    int p = nc++; l[p] = len;
    if (q) memcpy(g[p], g[q], sizeof(g[p])), f[p] = f[q];
    else memset(g[p], 0, sizeof(g[p])), f[p] = 0;
    return p;
}

void clr() { nc = 0; gn(0); f[0] = -1; }

int extend(int p, int o) {
    //  if (g[p][o] && l[p] + 1 == l[g[p][o]]) return g[p][o];
    int np = gn(l[p] + 1);
    for (; ~p && !g[p][o]; p = f[p])
        g[p][o] = np;
    if (!~p) f[np] = 0;
    else {
        int q = g[p][o];
        if (l[q] == l[p] + 1) f[np] = q;
        else {
            int nq = gn(l[p] + 1, q);
            f[q] = f[np] = nq;
            for (; ~p && g[p][o] == q; p = f[p])
                g[p][o] = nq;
        }
    }
    return np;
}
```

```
int c[N], pos[N];
void sort_sam() {
    fill(c, c + n + 1, 0);
    for (int p = 0; p != nc; ++p) c[l[p]]++;
    for (int i = 1; i <= n; ++i) c[i] += c[i - 1];
    for (int p = 0; p != nc; ++p) pos[--c[l[p]]] = p;
}
```

```
struct Suffix_Automaton{
    int O,link[N],maxlen[N],trans[N][26];
    //link[i]: 后缀链接
    //trans[i]: 状态转移数组
    Suffix_Automaton(){O=1;}//根初始化为1
    inline int insert(Re ch,Re last){
        if(trans[last][ch]){
            Re p=last,x=trans[p][ch];
            if(maxlen[p]+1==maxlen[x])return x;//即最初的特判1
            else{
                Re y=++O;maxlen[y]=maxlen[p]+1;
                for(Re i=0;i<26;++i)trans[y][i]=trans[x][i];
                while(p&&trans[p][ch]==x)trans[p][ch]=y,p=link[p];
                link[y]=link[x],link[x]=y;
                return y;//即最初的特判2
            }
        }
        Re z=++O,p=last;maxlen[z]=maxlen[last]+1;
        while(p&&!trans[p][ch])trans[p][ch]=z,p=link[p];
        if(!p)link[z]=1;
        else{
            Re x=trans[p][ch];
            if(maxlen[p]+1==maxlen[x])link[z]=x;
            else{
                Re y=++O;maxlen[y]=maxlen[p]+1;
                for(Re i=0;i<26;++i)trans[y][i]=trans[x][i];
                while(p&&trans[p][ch]==x)trans[p][ch]=y,p=link[p];
                link[y]=link[x],link[z]=link[x]=y;
            }
        }
        return z;
    }
    inline void sakura(){
        LL ans=0;
        for(Re i=2;i<=O;++i)ans+=maxlen[i]-maxlen[link[i]];
        printf("%lld\n",ans);
    }
}SAM;
int main(){
//    freopen("123.txt","r",stdin);
    in(n);
    for(Re i=1;i<=n;++i){
        scanf("%s",ch+1);Re last=1;
        for(Re j=1;ch[j];++j)last=SAM.insert(ch[j]-'a',last);
    }
    SAM.sakura();
}
```

回文自动机

```
char s[N];
int g[N][26], f[N], l[N], nc;

int gn(int len) {
    int p = nc++; l[p] = len; f[p] = 0;
    memset(g[p], 0, sizeof(g[p]));
    return p;
}

int gf(int p, int i) {
    while (s[i] != s[i - l[p] - 1])
        p = f[p];
    return p;
}

void clr() {
    nc = 0; gn(0); gn(-1);
    f[0] = 1; f[1] = -1;
}

int extend(int p, int i) {
    int o = s[i] - 'a'; p = gf(p, i);
    if (!g[p][o]) {
        int q = gn(l[p] + 2); g[p][o] = q;
        if (p != 1) f[q] = g[gf(f[p], i)][o];
    }
    p = g[p][o];
    return p;
}
```

双端插入

```
namespace pam {

char s[N<<1]; int tl, tr;
int g[N][26], f[N], l[N], nc;
int pl, pr;

int gn(int len) {
    int q = nc++; l[q] = len; f[q] = 0;
    memset(g[q], 0, sizeof(g[q]));
    return q;
}

void clr() {
    fill(s + tl, s + tr + 1, 0);
    nc = 0; gn(0); gn(-1); f[0] = 1;
    tr = N; tl = tr + 1;
    pl = pr = 0;
}

int gfl(int p, int i) {
    while (s[i] != s[i + l[p] + 1]) p = f[p];
    return p;
}

int gfr(int p, int i) {
    while (s[i] != s[i - l[p] - 1]) p = f[p];
    return p;
}

int extl(int o) {
    s[--tl] = o + 'a'; int p = gfl(pl, tl);
    if (!g[p][o]) {
        int q = gn(l[p] + 2); g[p][o] = q;
        if (p != 1) f[q] = g[gfl(f[p], tl)][o];
    }
    pl = g[p][o];
    if (tl + l[pl] - 1 == tr) pr = pl;
    return h[pl];
}

int extr(int o) {
    s[++tr] = o + 'a'; int p = gfr(pr, tr);
    if (!g[p][o]) {
        int q = gn(l[p] + 2); g[p][o] = q;
        if (p != 1) f[q] = g[gfr(f[p], tr)][o];
    }
    pr = g[p][o];
    if (tr - l[pr] + 1 == tl) pl = pr;
    return h[pr];
}

}
```