Suffix Array

```
char str [MAX_N];

int N, m, SA [MAX_N], LCP [MAX_N];

int x [MAX_N], y [MAX_N], w [MAX_N], c [MAX_N];


inline bool cmp (const int a, const int b, const int l) { return (y [a] == y [b] && y [a + l] == y [b +
l]); }


void Sort () {

   for (int i = 0; i < m; ++i) w [i] = 0;

   for (int i = 0; i < N; ++i) ++w [x [y [i]]];

   for (int i = 0; i < m - 1; ++i) w [i + 1] += w [i];

   for (int i = N - 1; i >= 0; --i) SA [--w [x [y [i]]]] = y [i];

}


void DA () {

   ++N;

   for (int i = 0; i < N; ++i) x [i] = str [i], y[i] = i;

   Sort ();

   for (int i, j = 1, p = 1; p < N; j <<= 1, m = p) {

      for (p = 0, i = N - j; i < N; i++) y [p++] = i;

      for (int k = 0; k < N; ++k) if (SA [k] >= j) y [p++] = SA [k] - j;

      Sort ();

      for (swap (x, y), p = 1, x [SA [0]] = 0, i = 1; i < N; ++i) x [SA [i]] = cmp (SA [i - 1], SA [i], j) ? p -
```

```
1 : p++;
   }
   for (int i = 1; i < N; ++i) SA [i - 1] = SA [i]; --N;
}

void kasaiLCP () {
   for (int i = 0; i < N; ++i) c [SA [i]] = i;
   LCP [0] = 0;
   for (int i = 0, h = 0; i < N; ++i) if (c[i] > 0) {
         int j = SA [c [i] - 1];
         while (i + h < N && j + h < N && str [i + h] == str [j + h]) ++h;
         LCP [c [i]] = h;
         if (h > 0) --h;
      }
}

void suffixArray () {
   m = 256;
   N = strlen (str);
   DA ();
   kasaiLCP ();
}
```

Suffix Array DC3

```
#define F(x) ((x)/3+((x)%3==1?0:tb))
#define G(x) ((x)<tb?(x)*3+1:((x)-tb)*3+2)
using namespace std;
int wa[N],wb[N],wv[N],wS[N];
int rnk[N],height[N];   // rank hocche inverse sa, height hocche lcp array
int sa[N],r[N];
int c[N];


int c0(int *y,int a,int b)
{
   return y[a]==y[b]&&y[a+1]==y[b+1]&&y[a+2]==y[b+2];
}
int c12(int k,int *y,int a,int b)
{
   if(k==2) return y[a]<y[b]||y[a]==y[b]&&c12(1,y,a+1,b+1);
   else return y[a]<y[b]||y[a]==y[b]&&wv[a+1]<wv[b+1];
}
void sort(int *r,int *a,int *b,int n,int m)
{
   int i;
   for(i=0; i<n; i++) wv[i]=r[a[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--) b[--wS[wv[i]]]=a[i];
    return;
}
void build_suffix(int *r,int *sa,int n,int m)
{
    int i,j,*rn=r+n,*san=sa+n,ta=0,tb=(n+1)/3,tbc=0,p;
    r[n]=r[n+1]=0;
    for(i=0; i<n; i++) if(i%3!=0) wa[tbc++]=i;
    sort(r+2,wa,wb,tbc,m);
    sort(r+1,wb,wa,tbc,m);
    sort(r,wa,wb,tbc,m);
    for(p=1,rn[F(wb[0])]=0,i=1; i<tbc; i++)
        rn[F(wb[i])]=c0(r,wb[i-1],wb[i])?p-1:p++;
    if(p<tbc) build_suffix(rn,san,tbc,p);
    else for(i=0; i<tbc; i++) san[rn[i]]=i;
    for(i=0; i<tbc; i++) if(san[i]<tb) wb[ta++]=san[i]*3;
    if(n%3==1) wb[ta++]=n-1;
    sort(r,wb,wa,ta,m);
    for(i=0; i<tbc; i++) wv[wb[i]=G(san[i])]=i;
    for(i=0,j=0,p=0; i<ta && j<tbc; p++)
        sa[p]=c12(wb[j]%3,r,wa[i],wb[j])?wa[i++]:wb[j++];
    for(; i<ta; p++) sa[p]=wa[i++];
```

```
    for(; j<tbc; p++) sa[p]=wb[j++];
    return;
}
void get_lcp(int n)
{
    int i,j,k=0;
    for(i=0; i<=n; i++) rnk[sa[i]]=i;
    for(i=0; i<n; height[rnk[i++]]=k)
        for(k?k--:0,j=sa[rnk[i]-1]; r[i+k]==r[j+k]; k++);
    return;
}

int main()
{
    int n, k;
    cin >> n >> k;
    for(int i = 0; i<n; i++)
    {
        cin >> c[i];
        r[i] = c[i] + 1;
    }
    r[n] = 0;
    build_suffix(r, sa, n + 1, 256);
    get_lcp(n);
```

```
    int ans = 0;
    multiset< pair<int,int> > SET;
    for(int i = 1; i<=n; i++)
    {
        SET.insert(make_pair(height[i], i));
        if(SET.size() == k)
        {
            SET.erase(SET.find(make_pair(height[i - k + 1], i - k + 1)));
            ans = max(ans, SET.begin()->first);
        }
    }
    cout << ans << endl;
    ///resubmit
    return 0;
}
```

Longest Pallindromic Substrings

```
#define N 200007
#define F(x) ((x)/3+((x)%3==1?0:tb))
#define G(x) ((x)<tb?(x)*3+1:((x)-tb)*3+2)
using namespace std;
int wa[N],wb[N],wv[N],wS[N];
int rnk[N],height[N];   // rank hocche inverse sa, height hocche lcp array
int sa[3*N],r[3*N];
```

```
char str[N];
long long ara[N];


int c0(int *y,int a,int b)
{
   return y[a]==y[b]&&y[a+1]==y[b+1]&&y[a+2]==y[b+2];
}
int c12(int k,int *y,int a,int b)
{
   if(k==2) return y[a]<y[b]||y[a]==y[b]&&c12(1,y,a+1,b+1);
   else return y[a]<y[b]||y[a]==y[b]&&wv[a+1]<wv[b+1];
}
void sort(int *r,int *a,int *b,int n,int m)
{
   int i;
   for(i=0; i<n; i++) wv[i]=r[a[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--) b[--wS[wv[i]]]=a[i];
   return;
}
void build_suffix(int *r,int *sa,int n,int m)
{
```

```
    int i,j,*rn=r+n,*san=sa+n,ta=0,tb=(n+1)/3,tbc=0,p;
    r[n]=r[n+1]=0;
    for(i=0; i<n; i++) if(i%3!=0) wa[tbc++]=i;
    sort(r+2,wa,wb,tbc,m);
    sort(r+1,wb,wa,tbc,m);
    sort(r,wa,wb,tbc,m);
    for(p=1,rn[F(wb[0])]=0,i=1; i<tbc; i++)
        rn[F(wb[i])]=c0(r,wb[i-1],wb[i])?p-1:p++;
    if(p<tbc) build_suffix(rn,san,tbc,p);
    else for(i=0; i<tbc; i++) san[rn[i]]=i;
    for(i=0; i<tbc; i++) if(san[i]<tb) wb[ta++]=san[i]*3;
    if(n%3==1) wb[ta++]=n-1;
    sort(r,wb,wa,ta,m);
    for(i=0; i<tbc; i++) wv[wb[i]=G(san[i])]=i;
    for(i=0,j=0,p=0; i<ta && j<tbc; p++)
        sa[p]=c12(wb[j]%3,r,wa[i],wb[j])?wa[i++]:wb[j++];
    for(; i<ta; p++) sa[p]=wa[i++];
    for(; j<tbc; p++) sa[p]=wb[j++];
    return;
}
void get_lcp(int n)
{
    int i,j,k=0;
    for(i=0; i<=n; i++) rnk[sa[i]]=i;
```

```
    for(i=0; i<n; height[rnk[i++]]=k)
        for(k?k--:0,j=sa[rnk[i]-1]; r[i+k]==r[j+k]; k++);
    return;
}
void print(int n)
{
    for(int i = 1; i<=n; i++)
    {
        int from = sa[i];
        while(from < n)
        {
            printf("%c", str[from++]);
        }
        printf("\n");
    }
}
int tree[4 * N];
void update(int node,int b,int e)
{
    if(b == e)
    {
        tree[node] = height[b];
        return;
    }
```

```
    int mid = (b + e) / 2;
    update(node + node, b, mid); update(node + node + 1, mid + 1, e);
    tree[node] = min(tree[node + node], tree[node + node + 1]);
}
int query(int node,int b, int e, int x, int y)
{
    if(e < x || b > y || e < b) return INT_MAX;
    if(b >= x && e <= y) return tree[node];
    int mid = (b + e) / 2;
    return  min(query(node + node, b, mid, x, y), query(node + node + 1, mid + 1, e, x, y));
}
int main()
{
    int n;
    scanf("%d",&n);
    scanf("%s",str);
    int j = n - 1;
    for(int i = 0; i<=n + n; i++)
    {
        if(i < n) r[i] = int(str[i]);
        else if(i == n) r[i] = int('#');
        else r[i] = int(str[j--]);
    }
    n = n + n + 1;
```

```
    r[n] = 1;
    build_suffix(r, sa, n + 1, 256);
    get_lcp(n);
    update(1,1,n);
    int ans = 1, go = n / 2;
    for(int i = 0; i < go; i++)
    {
       int l = rnk[i], r = rnk[2 * go - i];
       if(l > r) swap(l, r);
       int q = query(1, 1, n, l + 1, r);
       ans = max(ans, 2 * q - 1);
    }
    for(int i = 1; i < go; i++)
    {
      int l = rnk[i], r = rnk[2 * go - i + 1];
      if(l > r) swap(l, r);
      int q = query(1, 1, n, l + 1, r);
      ans = max(ans, 2 * q);
    }
    printf("%d\n",ans);
    ///resubmi
    return 0;
}
```

MANACHAR LPS O(N)

```
#include <bits/stdc++.h>

#include <ext/algorithm>

#include <ext/numeric>


using namespace std;

using namespace __gnu_cxx;


#define endl '\n'


vector<int> manacher(const string &s)

{

	int n = 2 * s.length();

	vector<int> rad(n);


	for (int i = 0, j = 0, k; i < n; i += k, j = max(j - k, 0))

	{

		for( ; i >= j && i + j + 1 < n

			&& s[(i - j) / 2] == s[(i + j + 1) / 2]; ++j);


		rad[i] = j;


		for (k = 1; i >= k && rad[i] >= k
```

```
                && rad[i - k] != rad[i] - k; ++k)
                rad[i + k] = min(rad[i - k], rad[i] - k);
        }

        return rad;
}

int main()
{
        ios_base::sync_with_stdio(0);
        cin.tie(0);

        int n;
        string s;
        cin >> n >> s;

        auto rad = manacher(s);
        cout << *max_element(rad.begin(), rad.end()) << endl;

        return 0;
}
```

kth substring

```
#define N 100007
#define F(x) ((x)/3+((x)%3==1?0:tb))
#define G(x) ((x)<tb?(x)*3+1:((x)-tb)*3+2)
using namespace std;
int wa[N],wb[N],wv[N],wS[N];
int rnk[N],height[N];   // rank hocche inverse sa, height hocche lcp array
int sa[3*N],r[3*N];
char str[N];
long long ara[N];


int c0(int *y,int a,int b)
{
   return y[a]==y[b]&&y[a+1]==y[b+1]&&y[a+2]==y[b+2];
}
int c12(int k,int *y,int a,int b)
{
   if(k==2) return y[a]<y[b]||y[a]==y[b]&&c12(1,y,a+1,b+1);
   else return y[a]<y[b]||y[a]==y[b]&&wv[a+1]<wv[b+1];
}
void sort(int *r,int *a,int *b,int n,int m)
{
   int i;
   for(i=0; i<n; i++) wv[i]=r[a[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--) b[--wS[wv[i]]]=a[i];
    return;
}
void build_suffix(int *r,int *sa,int n,int m)
{
    int i,j,*rn=r+n,*san=sa+n,ta=0,tb=(n+1)/3,tbc=0,p;
    r[n]=r[n+1]=0;
    for(i=0; i<n; i++) if(i%3!=0) wa[tbc++]=i;
    sort(r+2,wa,wb,tbc,m);
    sort(r+1,wb,wa,tbc,m);
    sort(r,wa,wb,tbc,m);
    for(p=1,rn[F(wb[0])]=0,i=1; i<tbc; i++)
        rn[F(wb[i])]=c0(r,wb[i-1],wb[i])?p-1:p++;
    if(p<tbc) build_suffix(rn,san,tbc,p);
    else for(i=0; i<tbc; i++) san[rn[i]]=i;
    for(i=0; i<tbc; i++) if(san[i]<tb) wb[ta++]=san[i]*3;
    if(n%3==1) wb[ta++]=n-1;
    sort(r,wb,wa,ta,m);
    for(i=0; i<tbc; i++) wv[wb[i]=G(san[i])]=i;
    for(i=0,j=0,p=0; i<ta && j<tbc; p++)
        sa[p]=c12(wb[j]%3,r,wa[i],wb[j])?wa[i++]:wb[j++];
    for(; i<ta; p++) sa[p]=wa[i++];
```

```
    for(; j<tbc; p++) sa[p]=wb[j++];
    return;
}
void get_lcp(int n)
{
    int i,j,k=0;
    for(i=0; i<=n; i++) rnk[sa[i]]=i;
    for(i=0; i<n; height[rnk[i++]]=k)
        for(k?k--:0,j=sa[rnk[i]-1]; r[i+k]==r[j+k]; k++);
    return;
}
void print(int n)
{
    for(int i = 1; i<=n; i++)
    {
        int from = sa[i];
        while(from < n)
        {
            printf("%c", str[from++]);
        }
        printf("\n");
    }
}
int main()
```

```
{
    scanf("%s",str);
    int n = strlen(str);
    for(int i = 0; i<n; i++) r[i] = int(str[i]);
    r[n] = 1;
    build_suffix(r, sa, n + 1, 256);
    get_lcp(n);
    for(int i = 1; i<=n; i++)
    {
        ara[i] = ara[i - 1] + n - sa[i] - height[i];
    }
    int q;
    scanf("%d",&q);
    //print(n);
    while(q--)
    {
        long long k;
        scanf("%lld",&k);
        int pos = upper_bound(ara, ara + n + 1, k - 1) - ara;
        int need = k - ara[pos - 1], to = sa[pos] + height[pos] + need, from = sa[pos];
        //cout << pos << " " << need << endl;
        while(from < to)
        {
            printf("%c",str[from]);
```

```
        from++;
      }
      printf("\n");
    }
    ///resubmi
    return 0;
}
```

a(i,k) = b(j,k) triple(i,j,k)

```
#define F(x) ((x)/3+((x)%3==1?0:tb))
#define G(x) ((x)<tb?(x)*3+1:((x)-tb)*3+2)
using namespace std;
namespace suffixarray {
  int wa[N],wb[N],wv[N],wS[N];
  int rnk[N],height[N];
  int sa[3*N],r[3*N];
  int type[3*N], vis[3*N];
  ll st[N][2];
  int Timer;

  void init() {
    Timer = 0;
    for(int i = 0; i < 3 * N; i++) {
      vis[i] = 0;
```

```
  }
}

int c0(int *y,int a,int b) {
   return y[a]==y[b]&&y[a+1]==y[b+1]&&y[a+2]==y[b+2];
}

int c12(int k,int *y,int a,int b) {
   if(k==2)
      return y[a]<y[b]||y[a]==y[b]&&c12(1,y,a+1,b+1);
   else
      return y[a]<y[b]||y[a]==y[b]&&wv[a+1]<wv[b+1];
}

void sort(int *r,int *a,int *b,int n,int m) {
   int i;
   for(i=0; i<n; i++)
      wv[i]=r[a[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--)
        b[--wS[wv[i]]]=a[i];
    return;
}

void build_suffix(int *r,int *sa,int n,int m) {
    int i,j,*rn=r+n,*san=sa+n,ta=0,tb=(n+1)/3,tbc=0,p;
    r[n]=r[n+1]=0;
    for(i=0; i<n; i++)
        if(i%3!=0)
            wa[tbc++]=i;
    sort(r+2,wa,wb,tbc,m);
    sort(r+1,wb,wa,tbc,m);
    sort(r,wa,wb,tbc,m);
    for(p=1,rn[F(wb[0])]=0,i=1; i<tbc; i++)
        rn[F(wb[i])]=c0(r,wb[i-1],wb[i])?p-1:p++;
    if(p<tbc)
        build_suffix(rn,san,tbc,p);
    else
        for(i=0; i<tbc; i++)
            san[rn[i]]=i;
    for(i=0; i<tbc; i++)
        if(san[i]<tb)
            wb[ta++]=san[i]*3;
```

```
    if(n%3==1)
        wb[ta++]=n-1;
    sort(r,wb,wa,ta,m);
    for(i=0; i<tbc; i++)
        wv[wb[i]=G(san[i])]=i;
    for(i=0,j=0,p=0; i<ta && j<tbc; p++)
        sa[p]=c12(wb[j]%3,r,wa[i],wb[j])?wa[i++]:wb[j++];
    for(; i<ta; p++)
        sa[p]=wa[i++];
    for(; j<tbc; p++)
        sa[p]=wb[j++];
    return;
}

void get_lcp(int n) {
    int i,j,k=0;
    for(i=0; i<=n; i++)
        rnk[sa[i]]=i;
    for(i=0; i<n; height[rnk[i++]]=k)
        for(k?k--:0,j=sa[rnk[i]-1]; r[i+k]==r[j+k]; k++);
    return;
}

ll getans(int k, int curLen, int len1,int len2) {
```

```
ll ans = 0, contribution = 0, top = 0;
for(int i = 3; i <= curLen; i++) {
  if(height[i] < k) {
    top = 0; contribution = 0;
  }
  else {
    int siz = 0;
    if(sa[i - 1] > len1) siz++, contribution += height[i] - k + 1;
    while(top > 0 && height[i] <= st[top - 1][0]) {
      contribution -= st[top - 1][1] * (st[top - 1][0] - height[i]);
      siz += st[top - 1][1];
      top--;
    }
    st[top][0] = height[i];
    st[top++][1] = siz;
    if(sa[i] < len1) ans += contribution;
  }
}
contribution = 0; top = 0;
for(int i = 3; i <= curLen; i++) {
  if(height[i] < k) {
    top = 0; contribution = 0;
  }
  else {
```

```
    int siz = 0;
    if(sa[i - 1] < len1) siz++, contribution += height[i] - k + 1;
    while(top > 0 && height[i] <= st[top - 1][0]) {
      contribution -= st[top - 1][1] * (st[top - 1][0] - height[i]);
      siz += st[top - 1][1];
      top--;
    }
    st[top][0] = height[i];
    st[top++][1] = siz;
    if(sa[i] > len1) ans += contribution;
   }
 }
  return ans;
}
void solve(int k) {
  int curLen = 0, len1, len2;
  for(int i = 0; i < 2; i++) {
    char str[N];
    scanf("%s", &str);
    i == 0 ? len1 = strlen(str) : len2 = strlen(str);
    for(int j = 0; j < (i == 0 ? len1 : len2); j++) {
      r[curLen++] = int(str[j]);
    }
    r[curLen++] = i + 1;
```

```
    }
    curLen--;
    build_suffix(r, sa, curLen + 1, 256);
    get_lcp(curLen);
    printf("%lld\n", suffixarray::getans(k, curLen, len1, len2));
    return;
  }
}

int main(int argc, char const *argv[]) {
  int k;
  while(scanf("%d",&k)) {
    if(k == 0) break;
    suffixarray::solve(k);
  }
  return 0;
}
```