**Cyclic shift**

A large binary number is represented by a string A of size N and comprises of 0s and 1s. You must perform a cyclic shift on this string. The cyclic shift operation is defined as follows:

* If the string A is [A0,A1,A2,...,AN−1], then after performing one cyclic shift, the string becomes [A1,A2,...,AN−1,A0].

You performed the shift infinite number of times and each time you recorded the value of the binary number represented by the string. The maximum binary number formed after performing (possibly 0) the operation is B. Your task is to determine the number of cyclic shifts that can be performed such that the value represented by the string A will be equal to B for the Kth time.

using namespace std;

typedef long long int ll;

#include <iostream>

#include <vector>

vector<int> z\_function(string);

vector<int> sort\_cyclic\_shifts(string const&);

string flip(string);

int main()

{

ios\_base::sync\_with\_stdio(false);

cin.tie(NULL);

ll tc;

cin >> tc; //2

while (tc != 0)

{

string str;

ll n, p;

cin >> n >> p; //5 2

cin >> str;

str = flip(str); //10101

//str = 01010

vector<int> z = z\_function(str); ////str = 01010

ll period = n; //5

for (ll i = 1; i < n; i++)

{

if (i + z[i] == n && n % i == 0)

{

period = i;

break;

}

}

string sub = str.substr(0, period);

vector<int> v = sort\_cyclic\_shifts(sub);

ll ans = v[0] + (p - 1) \* period;

cout << ans << "\n";

tc--;

}

return 0;

}

vector<int> z\_function(string s) //str = 01010

{

int n = (int)s.length(); //5

vector<int> z(n);

for (int i = 1, l = 0, r = 0; i < n; ++i)

{

if (i <= r)

z[i] = min(r - i + 1, z[i - l]);

while (i + z[i] < n && s[z[i]] == s[i + z[i]])

++z[i];

if (i + z[i] - 1 > r)

l = i, r = i + z[i] - 1;

}

return z;

}

vector<int> sort\_cyclic\_shifts(string const &s)

{

int n = s.size();

const int alphabet = 256;

vector<int> p(n), c(n), cnt(max(alphabet, n), 0);

for (int i = 0; i < n; i++)

cnt[s[i]]++;

for (int i = 1; i < alphabet; i++)

cnt[i] += cnt[i - 1];

for (int i = 0; i < n; i++)

p[--cnt[s[i]]] = i;

c[p[0]] = 0;

int classes = 1;

for (int i = 1; i < n; i++)

{

if (s[p[i]] != s[p[i - 1]])

classes++;

c[p[i]] = classes - 1;

}

vector<int> pn(n), cn(n);

for (int h = 0; (1 << h) < n; ++h)

{

for (int i = 0; i < n; i++)

{

pn[i] = p[i] - (1 << h);

if (pn[i] < 0)

pn[i] += n;

}

fill(cnt.begin(), cnt.begin() + classes, 0);

for (int i = 0; i < n; i++)

cnt[c[pn[i]]]++;

for (int i = 1; i < classes; i++)

cnt[i] += cnt[i - 1];

for (int i = n - 1; i >= 0; i--)

p[--cnt[c[pn[i]]]] = pn[i];

cn[p[0]] = 0;

classes = 1;

for (int i = 1; i < n; i++)

{

pair<int, int> cur = {c[p[i]], c[(p[i] + (1 << h)) % n]};

pair<int, int> prev = {c[p[i - 1]], c[(p[i - 1] + (1 << h)) % n]};

if (cur != prev)

++classes;

cn[p[i]] = classes - 1;

}

c.swap(cn);

}

return p;

}

string flip(string str) //1 0 1 0 1

{

string s; // s=01010

for (ll i = 0; i < str.length(); i++)

{

if (str[i] == '0')

s += '1';

else

s += '0';

}

return s; //01010

}