

Code Template for ACM-ICPC

UIT.HTH

October 26, 2020

Contents

1	DataStructures	1
1.1	RMQ	1
2	String	1
2.1	Manacher	1
2.2	KMP	1
2.3	LyndonDecomposition	2

1 DataStructures

1.1 RMQ

```
// index from 1
int table[MAXLOG][MAXN];
int numlog[MAXN];

void buildTable() {
    numlog[1] = 0;
    for (int i = 2; i <= N; i++)
        numlog[i] = numlog[i / 2] + 1;

    for (int i = 0; i <= numlog[N]; i++) {
        int curlen = 1 << i;
        for (int j = 1; j <= N; j++) {
            if (i == 0) {
                table[i][j] = a[j];
                continue;
            }
            table[i][j] = max(table[i - 1][j], table[i - 1][j + curlen / 2]);
        }
    }
}

int getMax(int l, int r) {
    int curlog = numlog[r - l + 1];
    return max(table[curlog][l], table[curlog][r - (1 << curlog) + 1]);
}
```

2 String

2.1 Manacher

```
#include <bits/stdc++.h>
using namespace std;

int manacher(string s)
{
    int n=s.size()*2-1;
    vector<int> f=vector<int>(n, 0);
    string a=string(n, '.');
    for (int i=0;i<n;i+=2)
        a[i]=s[i/2];

    int l=0, r=-1, center, res=0, j=0;
    for (int i=0; i<n; i++)
    {
        if (i>r)
            j=1;
        else
            j=min(f[l+r-i], r-i)+1;
            while (i-j>=0 && i+j<n && a[i-j]==a[i+j])
                j++;
            j--;
            f[i]=j;
            if (i+j>r)
            {
                r=i+j;
                l=i-j;
            }
            int len=(f[i]+i%2)/2*2+1-i%2;
            if (len>res)
```

```
{
    res=len;
    center=i;
}

return res;
}
```

```
string st;
int main()
{
    cin>>st;
    cout<<manacher(st);
}
```

2.2 KMP

```
#include <bits/stdc++.h>
using namespace std;

int lps[10000003];
void kmp(string a,string b)
{
    int n=a.length();
    int m=b.length();
    int i=1;
    int len=0;
    while (i<m)
    {
        if (b[i]==b[len])
        {
            len++;
            lps[i]=len;
            i++;
        }
        else
        {
            if (len!=0)
                len=lps[len-1];
            else
                i++;
            lps[i]=0;
        }
    }
    i=0;
    int j=0;
    int ans=0;
    while(i<n)
    {
        while (a[i]==b[j])
        {
            i++;
            j++;
            if (i>=n || j>=m)
                break;
        }
        if (j==m)
        {
            cout<<i-j+1<<" ";
            j=lps[j-1];
        }
        else
            if (j!=0)
                j=lps[j-1];
        else
```

```
        i++;
    if (i>=n)
        break;
    }
}

string st1,st2;
int main()
{
    cin>>st1>>st2;
    kmp(st1,st2);
}
```

2.3 LyndonDecomposition

```
#include <bits/stdc++.h>
using namespace std;

void lyndon(string s)
{
    int n=s.length();
    int i=0;
    while (i<n)
    {
        int j=i+1;
        int k=i;
        while (j<n && s[k]<=s[j])
        {
            if (s[k]<s[j])
                k=i;
            else
                k++;
            j++;
        }
        while (i <= k)
        {
            cout<<s.substr(i,j-k)<<endl;
            i+=j-k;
        }
    }
    cout<<endl;
}

string st;
int main()
{
    cin>>st;
    lyndon(st);
}
```
