CRACK-A- HACK

MORGAN AND A STRING

Course : Algorithmic Problem and Solving {17ECSE309}

**By: Om Prakash Das
01FE15BCS126**

Introduction

Lexicographical order

In mathematics, the lexicographic or lexicographical order (also known as lexical order, dictionary order, alphabetical order or lexicographic(al) product) is a generalization of the way words are alphabetically ordered based on the alphabetical order of their component letters. This generalization consists primarily in defining a total order over the sequences (often called strings in computer science) of elements of a finite totally ordered set, often called alphabet.

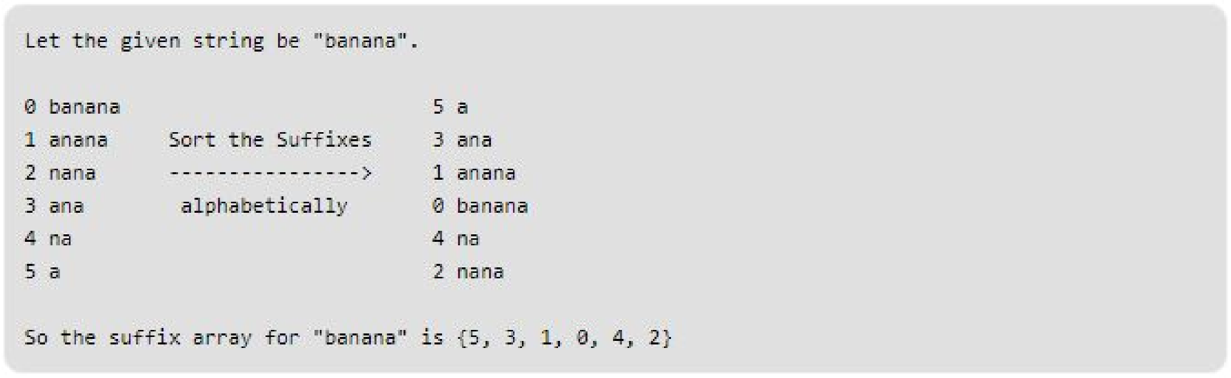
Lexicographically Minimal String

If two strings are merged to make a lexicographically minimal string , then it doesn't means that you can concat both and sort them lexicographically, It means you need to go character by character , maintain the string order and find the lexicographically minimal string.

Suffix Array

A suffix array is a sorted array of all suffixes of a given string.

A suffix array can be constructed from Suffix tree by doing a DFS traversal of the suffix tree. Infact Suffix array and suffix tree both can be constructed from each other in linear time.
Advantages of suffix arrays over suffix trees include improved space requirements, simpler linear time construction algorithms (e.g., compared to Ukkonen's algorithm) and improved cache locality.



```
Let the given string be "banana".

0 banana                          5 a
1 anana     Sort the Suffixes     3 ana
2 nana      ---------------->     1 anana
3 ana        alphabetically       0 banana
4 na                              4 na
5 a                               2 nana

So the suffix array for "banana" is {5, 3, 1, 0, 4, 2}
```

A simple method to construct suffix array is to make an array of all suffixes and then sort the array. Following is implementation of simple method.

Suffix array is an extremely useful data structure, it can be used for a wide range of problems. Following are some famous problems where Suffix array can be used.
1) Pattern Searching.
2) Finding the longest repeated substring.
3) Finding the longest common substring.
4) Finding the longest palindrome in a string.

CODE

```
#include<bits/stdc++.h>
using namespace std;
const int N_MAX = 100000 + 10;
char A[N_MAX], B[N_MAX], S[N_MAX << 1];
struct SuffixArray {
public:
    const static int N_MAX = 200000 + 10;
    int count[N_MAX], tr[2][N_MAX], ts[N_MAX];
    int SuffixArray[N_MAX], rk[N_MAX], ht[N_MAX], len;
    void makeSuffixArray(const char *s, int n, int m=256) {
        int i,j,k,*x=tr[0],*y=tr[1];
        this->len = n;
        memset(count,0,sizeof(count[0])*m);
        for (i=0; i<n; ++i) count[s[i]]++;
        partial_sum(count,count+m,count);
        for (i=0; i<n; ++i) rk[i]=count[s[i]]-1;
        for (k=1; k<=n; k<<=1) {
            for (i=0; i<n; ++i) x[i]=rk[i],y[i]=i+k<n?rk[i+k]+1:0;
            fill(count,count+n+1,0);
            for (i=0; i<n; ++i) count[y[i]]++;
            partial_sum(count,count+n+1,count);
            for (i=n-1; i>=0; --i) ts[--count[y[i]]]=i;
            fill(count,count+n+1,0);
            for (i=0; i<n; ++i) count[x[i]]++;
            partial_sum(count,count+n+1,count);
            for (i=n-1; i>=0; --i)
SuffixArray[--count[x[ts[i]]]]=ts[i];
            for (i=rk[SuffixArray[0]]=0; i+1<n; ++i) {

rk[SuffixArray[i+1]]=rk[SuffixArray[i]]+(x[SuffixArray[i]]!=x[SuffixA
rray[i+1]]||y[SuffixArray[i]]!=y[SuffixArray[i+1]]);
            }
        }
        for (i=0,k=0; i<n; ++i) {
            if (!rk[i]) continue;
            for (j=SuffixArray[rk[i]-1];
i+k<n&&j+k<n&&s[i+k]==s[j+k];) k++;
            ht[rk[i]]=k;
            if (k) k--;
        }
        rmq_init(n);
    }
    inline int lcp(int a, int b) {
        a=rk[a],b=rk[b];
        if (a == b) return len - a + 1;
        if (a>b) swap(a,b);
        return rmq(a+1,b);
    }
private:
    int mx[N_MAX][20], LOG[N_MAX];
    void rmq_init(int n) {
        for (int i=-(LOG[0]=-1); i<n; ++i) LOG[i]=LOG[i>>1]+1;
        for (int i=0; i<n; ++i) mx[i][0]=ht[i];
        for (int i,j=1; (1<<j)<n; ++j) {
            for (i=0; i+(1<<j)<=n; ++i)
                mx[i][j]=min(mx[i][j-1],mx[i+(1<<(j-1))][j-1]);
        }
    }
    inline int rmq(int a, int b) {
        int k=LOG[b-a+1];
        return min(mx[a][k],mx[b-(1<<k)+1][k]);
    }
} SuffixArray;
int main() {
    int T;
    scanf("%d", &T);
    while (T --) {
        scanf("%s%s", A, B);
        int n = strlen(A), m = strlen(B), s = 0;
        for (int i = 0; i < n; ++ i) S[s ++] = A[i];
        S[s ++] = 'a';
        for (int i = 0; i < m; ++ i) S[s ++] = B[i];
        S[s] = 0;
        A[n] = B[m] = 'Z' + 1;
        SuffixArray.makeSuffixArray(S, s);
        int i = 0, j = 0;
        for (; i < n && j < m; ) {
            int l = SuffixArray.lcp(i, j + n + 1);
            if (A[i + l] < B[j + l]) putchar(A[i ++]);
            else putchar(B[j ++]);
        }
        while (i < n) putchar(A[i ++]);
        while (j < m) putchar(B[j ++]);
        puts("");}
      return 0;}
```

References

1. https://discuss.codechef.com/questions/54413/lexicographical-strings
2. https://en.wikipedia.org/wiki/Lexicographical_order
3. https://www.geeksforgeeks.org/suffix-array-set-1-introduction/