#include <bits/stdc++.h>

using namespace std;

struct Forest{

long long weight;

int root;

};

int pos1, pos2;

Forest forest[256];

struct Tree{

int left, right, parent;

int freq2;

int symbol = -1;

};

struct Code {

int length;

int codik[8];

char bukva;

};

Code codes[256];

Tree tree[512];

FILE \*f;

FILE \*g;

map <char, int> m;

void getmin(int size, int &p1, int &p2){

int min1;

int min2;

if (forest[0].weight < forest[1].weight) {

min1 = forest[0].weight;

p1 = 0;

min2 = forest[1].weight;

p2 = 1;

}

else {

min1 = forest[1].weight;

p1 = 1;

min2 = forest[0].weight;

p2 = 0;

}

for (int i = 2; i < size; i++) {

if (forest[i].weight < min1) {

min2 = min1;

p2 = p1;

min1 = forest[i].weight;

p1 = i;

} else {

if (forest[i].weight < min2) {

min2 = forest[i].weight;

p2 = i;

}

}

}

}

int main()

{

f = fopen("input.txt", "rb");

g = fopen("output.haff", "wb");

freopen("output.haff", "w", stdout);

int freq[256];

for (int i = 0; i < 256; i++) {

freq[i] = 0;

}

char ch;

while (fscanf(f, "%c", &ch) != -1) {

freq[ch]++;

m[ch]++;

}

int size\_forest = 0;

int size\_tree = 0;

for (int i = 0; i < 256; i++) {

if (freq[i] != 0) {

forest[size\_forest].weight = freq[i];

forest[size\_forest].root = size\_forest;

size\_forest++;

tree[size\_tree].left = -1;

tree[size\_tree].right = -1;

tree[size\_tree].parent = -1;

tree[size\_tree].freq2 = freq[i];

size\_tree++;

}

}

int cnt\_symbol = size\_forest;

while (size\_forest > 1) {

getmin(size\_forest, pos1, pos2);

tree[forest[pos1].root].parent = size\_tree;

tree[forest[pos2].root].parent = size\_tree;

forest[pos1].weight = forest[pos1].weight + forest[pos2].weight;

tree[size\_tree].left = forest[pos1].root;

tree[size\_tree].right = forest[pos2].root;

tree[size\_tree].parent = -1;

forest[pos1].root = size\_tree;

forest[pos2] = forest[size\_forest-1];

size\_forest--;

size\_tree++;

}

for (auto it = m.begin(); it != m.end(); it++) {

for (int i = 0; i < cnt\_symbol; i ++) {

if (tree[i].freq2 == it -> second) {

tree[i].symbol = it -> first;

}

}

}

for (int i = 0; i < cnt\_symbol; i++) {

int parnum = tree[i].parent;

int childnum = i;

int o = 0;

codes[i].bukva = tree[i].symbol;

while (parnum != -1) {

if (tree[parnum].left == childnum) {

codes[i].codik[o] = 0;

codes[i].length++;

o++;

}

if (tree[parnum].right == childnum) {

codes[i].codik[o] = 1;

codes[i].length++;

o++;

}

childnum = parnum;

parnum = tree[parnum].parent;

}

}

int kolvo = 0;

f = fopen("input.txt", "rb");

unsigned char byte = 0;

int byte\_l = 0;

vector <char> bt;

while (fscanf(f, "%c", &ch) != -1) {

for (int l = 0; l < cnt\_symbol; l++) {

if (ch == codes[l].bukva) {

for (int i = codes[l].length - 1; i >= 0 ; i--){

if (codes[l].codik[i] == 0) {

byte = (byte << 1) | 0;

byte\_l++;

}

if (codes[l].codik[i] == 1) {

byte = (byte << 1) | 1;

byte\_l++;

}

if (byte\_l == 8) {

bt.push\_back(byte);

kolvo++;

byte = 0;

byte\_l = 0;

}

}

}

}

}

if (byte\_l != 0) {

bt.push\_back(byte);

cout << (8-byte\_l) << " ";

}

cout << size\_tree << " " << kolvo << endl;

for (int i = 0; i < size\_tree; i++) {

cout << tree[i].left << " " << tree[i].right << " " << tree[i].parent << " " << tree[i].symbol << endl;

}

cout << "\*";

for (int i = 0; i < bt.size(); i++) {

cout << bt[i];

}

return 0;

}

//aaaaaaaeeeeeeebbbbbbbbbbccccccccccdddddaaaaabbbbbbbccccbbbbbbeeeedddeeeeeeeebbbbbbbbbbbbbbbbbceeeeee