Nanyang Technological University
Joker
Reference Book

Contents

1 String

1.1 KMP

```cpp
std::vector<int> kmp(std::string s) {
  int n = s.length();
  std::vector<int> pi(n);
  for (int i = 1; i < n; ++i) {
    int j = pi[i - 1];
    while (j && s[i] != s[j]) {
      j = pi[j - 1];
    }
    if (s[i] == s[j]) {
      j++;
    }
    pi[i] = j;
  }
  return pi;
}
```

1.2 Z-function

```cpp
std::vector<int> z_function(std::string s) {
  int n = s.length();
  std::vector<int> z(n);
  z[0] = n;
  for (int i = 1, l = 0, r = 0; i < n; ++i) {
    if (i <= r && z[i - l] < r - i + 1) {
      z[i] = z[i - l];
    } else {
      z[i] = std::max(0, r - i + 1);
      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;
}
```

1.3 Aho-Corasick algorithm

```cpp
const int maxn = 200005;

int ans[maxn];

struct Aho_Corasick {
  std::vector<int> id[maxn];
  int son[maxn][26];
  int fail[maxn];
  int val[maxn];
  int cnt;

  Aho_Corasick() {
    cnt = 0;
    memset(son, 0, sizeof(son));
    memset(fail, 0, sizeof(fail));
    memset(val, 0, sizeof(val));
  }

  void insert(std::string s, int _id) {
    int now = 0;
    for (auto c : s) {
      const int x = c - 'a';
      if (!son[now][x]) {
        son[now][x] = ++cnt;
      }
      now = son[now][x];
    }
    id[now].push_back(_id);
  }

  std::vector<int> fas[maxn];

  void build() {
    std::queue<int> q;
    for (int i = 0; i < 26; ++i) {
      if (son[0][i]) {
        q.push(son[0][i]);
      }
    }
    while (!q.empty()) {
      int now = q.front();
      q.pop();
    for (int i = 0; i < 26; ++i) {
      if (son[now][i]) {
        fail[son[now][i]] = son[fail[now]][i];
        q.push(son[now][i]);
      } else {
        son[now][i] = son[fail[now]][i];
      }
    }
  }
}

void getval(std::string s) {
  int now = 0;
  for (auto c : s) {
    now = son[now][c - 'a'];
    val[now]++;
  }
}

void build_fail_tree() {
  for (int i = 1; i <= cnt; ++i) {
    fas[fail[i]].push_back(i);
  }
}

void dfs(int now = 0) {
  for (auto x : fas[now]) {
    dfs(x);
    val[now] += val[x];
  }
  if (!id[now].empty()) {
    for (auto x : id[now]) {
      ans[x] = val[now];
    }
  }
}
};

Aho_Corasick ac;

int n;

int main() {
  std::cin >> n;
  for (int i = 1; i <= n; ++i) {
    std::string s;
    std::cin >> s;
    ac.insert(s, i);
  }
  ac.build();
  std::string s;
  std::cin >> s;
  ac.getval(s);
  ac.build_fail_tree();
  ac.dfs();
  for (int i = 1; i <= n; ++i) {
    std::cout << ans[i] << std::endl;
  }
  return 0;
}
```

1.4 manachar

```
int n;
char s[N],ss[N];
int len[N];
void Manachar(char *s,int n){
    For(i,0,n+1)
        len[i]=0;
    int mx=1;
    For(i,1,n){
        len[i]=max(1,min(mx+len[mx]-i,len[mx*2-i]));
        while (s[i-len[i]]==s[i+len[i]])
            len[i]++;
        if (i+len[i]>mx+len[mx])
            mx=i;
    }
}
int solve(char *s,int n){
    ss[0]='#',ss[1]='*';
    For(i,1,n){
        ss[i<<1]=s[i];
        ss[i<<1|1]='*';
    }
    ss[n*2+2]='$';
    Manachar(ss,n*2+1);
    int ans=0;
    For(i,1,n*2+1)
        ans=max(ans,len[i]-1);
    return ans;
}
```

1.5 SuffixArray

```
struct SuffixArray {
    static const int N = 1000005; // the length of the string

    int n, m, cnt[N], sa[N], rk[N], id[N];

    void radixSort() {
        for (int i = 0; i < m; ++i) {
            cnt[i] = 0;
        }
        for (int i = 0; i < n; ++i) {
            ++cnt[rk[i]];
        }
        for (int i = 1; i < m; ++i) {
            cnt[i] += cnt[i - 1];
        }
        for (int i = n - 1; ~i; --i) {
            sa[--cnt[rk[id[i]]]] = id[i];
        }
    }

    bool cmp(int x, int y, int l) {
        return id[x] == id[y] && id[x + l] == id[y + l];
    }

    template<typename T>
    void initSA(T first, T last) {
        n = last - first, m = 0;
        for (int i = 0; i < n; ++i) {
            rk[i] = *(first + i);
            m = std::max(m, rk[i] + 1);
            id[i] = i;
        }
        radixSort();
        for (int l = 1, p = 0; p < n && l < n; m = p, l <<= 1) {
            p = 0;
            for (int i = n - l; i < n; ++i) {
                id[p++] = i;
            }
            for (int i = 0; i < n; ++i) {
                if (sa[i] >= l && p < n) {
                    id[p++] = sa[i] - l;
                }
            }
            radixSort();
            for (int i = 0; i < n; ++i) id[i] = rk[i];
            p = 1, rk[sa[0]] = 0;
            for (int i = 1; i < n; ++i) {
                if (!cmp(sa[i - 1], sa[i], l) && p < n) ++p;
                rk[sa[i]] = p - 1;
            }
        }
    }
} SA;

int main() {
    n = readStr(s);
    SA.initSA(s, s + n);
    for (int i = 0; i < n; ++i) {
        print(SA.sa[i] + 1, ' ');
    }
    putchar('\n');
}
```