Algorithm Code Book

Tanvir Hasan Anick

July 8, 2015

Contents

1 Data Structure **2**

2 Graph Theory **3**

3 Flow networks/ matching **4**

4 Dynamic programming **5**

5 Strings **6**
- 5.1 KMP . . . 6
- 5.2 Trie . . . 7
  - 5.2.1 Static Trie . . . 7
- 5.3 Aho Corasick . . . 8
  - 5.3.1 Aho Corasick with Dynamic Trie . . . 8
  - 5.3.2 Aho Corasick with Static Trie . . . 10
- 5.4 Manacher's Algorithm . . . 13

6 Computational geometry **14**

7 Math **15**

8 Number Theory **16**

Chapter 1

Data Structure

Chapter 2

Graph Theory

Chapter 3

Flow networks/ matching

Chapter 4

Dynamic programming

Chapter 5

Strings

5.1 KMP

Tutorial

```
#include<bits/stdc++.h>
using namespace std;
char TXT[10000000],ptr[10000000];
vector<int> compute_prefix(const char *p)
{
    int m=strlen(p+1);
    vector<int> prefix(m+1);
    prefix[1]=0;
    int k=0;
    for(int i=2; i<=m; i++)
    {
        while(k>0 and p[k+1]!=p[i])k=prefix[k];
        if(p[k+1]==p[i])k=k+1;
        prefix[i]=k;
    }
    return prefix;
}
vector<int> KMP_match(const char *txt,const char *ptrn)
{
    int n=strlen(txt+1);
    int m=strlen(ptrn+1);
    vector<int> Prefix=compute_prefix(ptrn);
    vector<int>Match_position;
    int q=0;
    for(int i=1; i<=n; i++)
    {
        while(q>0 and ptrn[q+1]!=txt[i])q=Prefix[q];
        if(ptrn[q+1]==txt[i])q=q+1;
        if(q==m)
        {
            Match_position.push_back(i-m);
            q=Prefix[q];
        }
    }
```

```
    return Match_position;
}
int main()
{
    scanf("%s %s",TXT+1,ptr+1);
    vector<int> Match_position=KMP_match(TXT,ptr);
    for(int i=0; i<Match_position.size(); i++)
    {
        if(!i)printf("%d",Match_position[i]);
        else printf(" %d",Match_position[i]);
    }
    return 0;
}
```

5.2 Trie

5.2.1 Static Trie

```
#define Max 10005
int getId(char c)
{
    return c>='a'?c-'a':c-'A'+26;
}
struct Trie
{
    struct Tree
    {
        int Next[52];
        bool word;
        void clear()
        {
            word=false;
            memset(Next,-1,sizeof(Next));
        }
    }T[Max];
    int ptr;
    void clear()
    {
        ptr=1;
        T[0].clear();
        memset(T[0].Next,0,sizeof(T[0].Next));
    }
    void Insert(const char *str)
    {
        int p=0;
        for(int i=0;str[i];i++)
        {
            int id=getId(str[i]);
            if(T[p].Next[id]<=0)
            {
                T[p].Next[id]=ptr;
                T[ptr++].clear();
            }
            p=T[p].Next[id];
        }
```

```
        T[p].word=true;
    }
    bool Search(const char *str)
    {
        int p=0;
        for(int i=0;str[i];i++)
        {
            int id=getId(str[i]);
            if(T[p].Next[id]>0)
            {
                p=T[p].Next[id];
            }
            else return false;
        }
        return T[p].word;
    }
};
Trie A;
```

5.3 Aho Corasick

5.3.1 Aho Corasick with Dynamic Trie

```
#include<bits/stdc++.h>
using namespace std;
#define Max 26
int getID(char c)
{
    return c>='a'?c-'a':c-'A';
}
char inp[1000005];
char text[1000005];
int ans[5000];
map<string,int>Map;
vector<int>v;
struct Trie
{
    Trie *next[26],*fail;
    int stringMap;
    Trie()
    {
        stringMap=0;
        for(int i=0;i<Max;i++)next[i]=NULL;
        fail=NULL;
    }
};
Trie *root;
void Insert(const char *str,int M)
{
    Trie *p=root;
    for(int i=0;str[i];i++)
    {
        int id=getID(str[i]);
        if(p->next[id]==NULL)p->next[id]=new Trie();
        p=p->next[id];
    }
```

```
34      p->stringMap=M;
35  }
36  void computeFailure()
37  {
38      Trie *u,*prefix;
39      queue<Trie*>Q;
40      Q.push(root);
41      while(!Q.empty())
42      {
43          u=Q.front(); ///Take a new node
44          Q.pop();
45          for(int i=0;i<Max;i++)
46          {
47              if(u->next[i]!=NULL) ///select fail position of ith
    node of parent u
48              {
49                  prefix=u->fail; /// Going to u node fail position/
    prefix position
50                  while(prefix!=NULL)
51                  {
52                      if(prefix->next[i]!=NULL) ///if match found
53                      {
54                          u->next[i]->fail=prefix->next[i];
55                          break;
56                      }
57                      prefix=prefix->fail; /// match not found, going
     to upper child prefix position
58                  }
59                  if(prefix==NULL)u->next[i]->fail=root;
60                  Q.push(u->next[i]);
61              }
62          }
63      }
64  }
65  void AhoCorasick(const char *str)
66  {
67      Trie *p=root;
68      int cnt=0;
69      for(int i=0;str[i];i++)
70      {
71          int id=getID(str[i]);
72          while(p->next[id]==NULL&&p!=root)p=p->fail,cnt++;
73          if(p->next[id]!=NULL)p=p->next[id];
74          Trie *tp=p;
75          while(tp!=root)
76          {
77              cnt++;
78              if(tp->stringMap>0)ans[tp->stringMap]++;
79              tp=tp->fail;
80          }
81      }
82  }
83  void Delete(Trie *u)
84  {
85      if(u==NULL)return;
86      for(int i=0;i<Max;i++)Delete(u->next[i]);
87      delete u;
```

```
}

int main()
{
    int test;
    scanf("%d",&test);
    for(int t=1;t<=test;t++)
    {
        Map.clear();
        v.clear();
        memset(ans,0,sizeof(ans));
        root=new Trie();
        int N;
        scanf("%d",&N);
        scanf("%s",text);
        int cnt=1;
        for(int i=0;i<N;i++)
        {
            scanf("%s",inp);
            if(Map.find(inp)==Map.end())Map[inp]=cnt++;
            Insert(inp,Map[inp]);
            v.push_back(Map[inp]);
        }
        computeFailure();
        AhoCorasick(text);
        printf("Case %d:\n",t);
        for(int i=0;i<N;i++)
        {
            printf("%d\n",ans[v[i]]);
        }
        Delete(root);
    }
    return 0;
}
```

5.3.2 Aho Corasick with Static Trie

```
#include<bits/stdc++.h>
using namespace std;
#define root 0
#define NuLL -1
#define Max 248878
#define MC 26
int ans[10000];
char text[1000005];
char inp[100000];
map<string,int>Map;
vector<int> v;
int getID(const char c)
{
    return c>='a'?c-'a':c-'A';
}
struct Trie
{
    struct node
    {
        int Next[26],fail;
```

```
        int stringMap;
        void clear()
        {
            memset(Next,-1,sizeof(Next));
            fail=-1;
            stringMap=0;
        }
    }T[Max];
    int ptr;
    void clear()
    {
        ptr=1;
        T[0].clear();
    }
    void Insert(char *str,int M)
    {
        int p=0;
        for(int i=0;str[i];i++)
        {
            int id=getID(str[i]);
            if(T[p].Next[id]==-1)
            {
                T[p].Next[id]=ptr;
                T[ptr++].clear();
            }
            int q=p;
            p=T[p].Next[id];
            if(p<0)
            {
                while(1);
            }
        }
        T[p].stringMap=M;
    }
    void ComputeFailure()
    {
        queue<int>Q;
        Q.push(root);
        int u,prefix;
        int cnt=0,cnt2=0;
        while(!Q.empty())
        {
            u=Q.front();
            Q.pop();
            for(int i=0;i<MC;i++)
            {
                if(T[u].Next[i]!=NuLL)
                {
                    int now=T[u].Next[i];
                    prefix=T[u].fail;
                    while(prefix!=NuLL)
                    {
                        cnt2++;
                        if(T[prefix].Next[i]!=NuLL)
                        {
                            T[now].fail=T[prefix].Next[i];
                            break;
```

```
                        }
                        prefix=T[prefix].fail;
                    }
                    if(prefix==NuLL)T[now].fail=root;
                    Q.push(now);
                }
            }
        }
    }
};
void AhoCorasick(const Trie &A, const char *str)
{
    int p=root;
    int cnt1=0,cnt2=0;
    for(int i=0;str[i];i++)
    {
        int id=getID(str[i]);
        while(A.T[p].Next[id]==NuLL&&p!=root)p=A.T[p].fail;
        if(p!=NuLL&&A.T[p].Next[id]!=NuLL)p=A.T[p].Next[id];
        int tp=p;
        while(tp!=root)
        {
            if(A.T[tp].stringMap>0)ans[A.T[tp].stringMap]++;
            tp=A.T[tp].fail;
        }
    }
}
Trie A;
int main()
{
    #ifdef _ANICK_
        freopen("input.txt","r",stdin);
    #endif // _ANICK_
    int test;
    scanf("%d",&test);
    for(int t=1;t<=test;t++)
    {
        Map.clear();
        v.clear();
        memset(ans,0,sizeof(ans));
        A.clear();
        int N;
        scanf("%d",&N);
        scanf("%s",text);
        int cnt=1;
        for(int i=0;i<N;i++)
        {
            scanf("%s",inp);
            if(Map.find(inp)==Map.end())Map[inp]=cnt++;
            A.Insert(inp,Map[inp]);
            v.push_back(Map[inp]);
        }
        A.ComputeFailure();
        AhoCorasick(A,text);
        printf("Case %d:\n",t);
        for(int i=0;i<N;i++)
        {
```

```
            printf("%d\n",ans[v[i]]);
        }
    }
    return 0;
}
```

5.4 Manacher's Algorithm

```
#include<bits/stdc++.h>
using namespace std;
string s, t;
char str[1000005];
void prepare_string()
{
    int i;
    t = "^#";
    for(i = 0; i < s.size(); i++)
        t += s[i], t += "#";
    t += "$";
}

int manacher()
{
    prepare_string();

    int P[t.size()], c = 0, r = 0, i, i_mirror, n = t.size() - 1;

    for(i = 1; i < n; i++)
    {
        i_mirror = (2 * c) - i;

        P[i] = r > i? min(r - i, P[i_mirror]) : 0;

        while(t[i + 1 + P[i]] == t[i - 1 - P[i]])
            P[i]++;

        if(i + P[i] > r)
        {
            c = i;
            r = i + P[i];
        }
    }
    return *max_element(P + 1, P + n);
}

int main()
{
    int kase = 1;
    while(scanf(" %s", str) && str[0] != 'E')
    {
        s = str;
        printf("Case %d: %d\n", kase++, manacher());
    }
    return 0;
}
```

Chapter 6

Computational geometry

Chapter 7

Math

Chapter 8

Number Theory