ACM Template

Rien

July 17, 2021

rien_zhu@163.com

Contents

1 字符串

1.1 KMP

```
/*
 * Args:
 *   s[]: string
 * Return:
 *   fail[]: failure function
 */
int fail[N];
void getfail(char s[])
{
  fail[0] = -1;
  int p = -1;
  for (int i = 0; s[i]; i ++) {
    while (p!=-1 && s[i]!=s[p])  p = fail[p];
    fail[i+1] = ++p;
  }
}
```

1.2 Suffix Automaton

```
/*
 * 1 call init()
 * 2 call add(x) to add every character in order
 *
 * Args:
 * Return:
 *   an automaton
 *   link: link path pointer
 *    len: maximum length
 */
struct node{
  node* chd[26], *link;
  int len;
}a[3*N], *head, *last;
int top;
void init()
{
  memset(a, 0, sizeof(a));
  top = 0;
  head = last = &a[0];
}
```

```
void add(int x)
{
  node *p = &a[++top], *mid;
  p->len = last->len + 1;
  mid = last, last = p;
  for (; mid && !mid->chd[x]; mid = mid->link)  mid->chd[x] =
  ↪  p;
  if (!mid)  p->link = head;
  else{
    if (mid->len + 1 == mid->chd[x]->len) {
      p->link = mid->chd[x];
    } else {
      node *q = mid->chd[x], *r = &a[++top];
      *r = *q, q->link = p->link = r;
      r->len = mid->len + 1;
      for (; mid && mid->chd[x] == q; mid = mid->link)
      ↪  mid->chd[x] = r;
    }
  }
}
```

2 数学

2.1 快速幂

2.1.1 数字快速幂

```
1  //a 的 b 次方对 p 取余
2  long long ksm(long long a,long long b,long long p)
3  {
4          long long ret=1;
5          while(b){
6                  if(b&1) ret=ret*a%p;
7                  a=a*a%p;
8                  b>>=1;
9          }
10         return ret%p;
11 }
```

3 图论

3.1 Minimum Spanning Tree

3.1.1 Kruskal

```
/*
 * Args:
 *   edge: edges of graph, (u, v, w) = (edge[i].second.first,
↪  edge[i].second.first, edge[i].first)
 *   n: number of node, from 1 to n
 * Return:
 *   minimum spanning tree
 * 中文中文
 */
vector<pair<int, pair<int, int> > > edge;
int pre[N];
int find(int u)
{
  return u == pre[u] ? u : pre[u] = find(pre[u]);
}
int Union(int u, int v)
{
  pre[find(u)] = find(v);
}
int kruskal(int n)
{
  for (int i = 1; i <= n; i ++) pre[i] = i;
  sort(edge.begin(), edge.end());
  int ans = 0;
  for (auto x : edge) {
    int u = x.second.first, v = x.second.second, w = x.first;
    if (find(u) != find(v)) {
      Union(u, v);
      ans += w;
    }
  }
  return ans;
}
```

3.2 单源最短路

3.2.1 SPFA

```
/* 中文注释测试 */
/*
 * Args:
 *   g[]: graph, (u, v, w) = (u, g[u][i].first,
↪  g[u][i].second)
 *   st: source vertex
 * Return:
 *   dis[]: distance from source vertex to each other vertex
 */
vector<pair<int, int> > g[N];
int dis[N], vis[N];
void spfa(int st)
{
  memset(dis, -1, sizeof(dis));
  memset(vis, 0, sizeof(vis));
  queue<int> q;
  q.push(st);
  dis[st] = 0;
  vis[st] = true;
  while (!q.empty()) {
    int u = q.front();
    q.pop();
    vis[u] = false;
    for (auto x : g[u]) {
      int v = x.first, w = x.second;
      if (dis[v] == -1 || dis[u] + w < dis[v]) {
        dis[v] = dis[u] + w;
        if (!vis[v]) {
          vis[v] = true;
          q.push(v);
        }
      }
    }
  }
}
```

3.3 我也不知道

4 其他

4.1 输入输出

4.1.1 快读

```cpp
//快读
template <typename T> T &read(T &r) {
    r = 0; bool w = 0; char ch = getchar();
    while(ch < '0' || ch > '9') w = ch == '-' ? 1 : 0, ch = getchar();
    while(ch >= '0' && ch <= '9') r = (r << 3) + (r <<1) + (ch ^ 48), ch = getchar();
    return r = w ? -r : r;
}
//用法:
read(n);
```

4.1.2 关闭同步

```cpp
//关闭同步
ios::sync_with_stdio(0);
cin.tie(0);
```

4.2 高精度

```cpp
//高精度，支持乘法和加法但只支持正数
struct BigInt{
    const static int mod = 10000;
    const static int DLEN = 4;
    //根据题目要求可对 a 数组大小进行修改
    int a[6000],len;
    BigInt(){
        memset(a,0,sizeof(a));
        len=1;
    }
    BigInt(int v){
        memset(a,0,sizeof(a));
        len=0;
        do{
            a[len++]=v%mod;
            v/=mod;
        }while(v);
```

```
18      }
19      BigInt(const char s[]){
20          memset(a,0,sizeof(a));
21          int L=strlen(s);
22          len=L/DLEN;
23          if(L%DLEN)  len++;
24          int index = 0;
25          for(int i=L-1;i>=0;i-=DLEN){
26              int t=0;
27              int k=i-DLEN+1;
28              if(k<0) k=0;
29              for(int j=k;j<=i;++j)
30                  t=t*10+s[j]-'0';
31              a[index++]=t;
32          }
33      }
34      BigInt operator +(const BigInt &b)const {
35          BigInt res;
36          res.len=max(len,b.len);
37          for(int i=0;i<=res.len;++i)
38              res.a[i]=0;
39          for(int i=0;i<res.len;++i){
40              res.a[i]+=((i<len)?a[i]:0)+((i<b.len)?b.a[i]:0);
41              res.a[i+1]+=res.a[i]/mod;
42              res.a[i]%=mod;
43          }
44          if(res.a[res.len]>0) res.len++;
45          return res;
46      }
47      BigInt operator *(const BigInt &b)const {
48          BigInt res;
49          for(int i=0;i<len;++i){
50              int up= 0;
51              for(int j=0;j<b.len;++j){
52                  int temp=a[i]*b.a[j]+res.a[i+j]+up;
53                  res.a[i+j]=temp%mod;
54                  up=temp/mod;
55              }
56              if(up!=0)
57                  res.a[i+b.len]=up;
58          }
59          res.len=len+b.len;
60          while(res.a[res.len-1]==0 && res.len>1) res.len--;
61          return res;
```

```
62      }
63      void output(){
64          printf("%d",a[len-1]);
65          for(int i=len-2;i>=0;--i)
66              printf("%04d",a[i]);
67          printf("\n");
68      }
69  };
70  int  main()
71  {
72      //字符串读入
73      char a[2005],b[2005];
74      cin>>a>>b;
75      BigInt A,B;
76      A=BigInt(a),B=BigInt(b);
77      //可以直接用 cout 输出 char 数组内容
78      cout<<a<<" "<<b<<endl;
79      (A+B).output();//加法
80      (A+B).output();//乘法
81      return 0;
82  }
```

5 注意事项

- 注意初始点的设置, 初始点是否有效是否得到正确的更新
- 注意边界条件如 $<$ 和 $<=$, 注意特判如 $n = 1$