ACM Template

Rien

July 19, 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 最小生成树

无向图中，其某个子图中任意两个顶点都互相连通并且是一棵树，称之为生成树；若每个顶点有权值，则其权值和最小的生成树为最小生成树。

3.1.1 Prim 算法

```cpp
/*
Prim 求 MST
耗费矩阵 cost[][], 标号从 0 开始,0~n-1 注意注意!!!
返回最小生成树的权值, 返回-1 则表示原图不连通
*/
const int INF=0x3f3f3f3f;
const int MAXN=110;
bool vis[MAXN];
int lowc[MAXN];
//点从 0 到 n-1
int Prim(int cost[][MAXN],int n){
    int ans=0;
    memset(vis,0,sizeof vis);
    vis[0]=true;
    for(int i=1;i<n;++i)   lowc[i]=cost[0][i];
    for(int i=1;i<n;++i){
        int minc=INF;
        int p=-1;
        for(int j=0;j<n;++j){
            if(!vis[j]&&minc>lowc[j]){
                minc=lowc[j];
                p=j;
            }
        }
        if(minc==INF) return -1;//原图不连通
        ans+=minc;
        vis[p]=true;
        for(int j=0;j<n;++j){
            if(!vis[j] && lowc[j]>cost[p][j])
                lowc[j]=cost[p][j];
        }
    }
    return ans;
}
int main()
```

```
{
    int cost[MAXN][MAXN];
    //注意对耗费矩阵赋初值,memset 只能读 1 个字符, 这样和都赋给 INF 结果一样
    memset(cost,0x3f,sizeof(cost));
    while (k--){
        int u,v,w;
        cin>>u>>v>>w;
        //读入数据, 注意题目给出的节点是否从 0 开始且无向边要两次赋值
        u--;
        v--;
        cost[u][v]=w;
        cost[v][u]=w;
    }
    cout<<Prim(cost,n)<<endl;
    return 0;
}
```

3.1.2 Kruskal 算法

```
/*
Kruskal 算法求 MST
*/
//根据题目调试最大点数和边数
const int MAXN=1100;//最大点数
const int MAXM=200005;//最大边数
int F[MAXN];//并查集
struct Edge {
    int u,v,w;
}edge[MAXM];//储存边的信息: 起点, 终点, 权值
int tol=0;//边数, 记得赋值为 0
void addedge(int u,int v,int w){//加边
    edge[tol].u=u;
    edge[tol].v=v;
    edge[tol++].w=w;
}
bool cmp(Edge a,Edge b){//排序
    return a.w<b.w;
}
int find(int x){
    return F[x]==x?x:F[x]=find(F[x]);
}
//传入点数, 返回最小生成树权值, 如果不连通返回-1
```

```cpp
int Kruskal(int n){
    //根据点的编号 (从 0 或从 1 开始) 初始并查集
    for(int i=1;i<=n;++i) F[i]=i;
    sort(edge,edge+tol,cmp);
    int cnt=0;//计算加入边数
    int ans=0;
    for(int i=0;i<tol;++i){
        int u=edge[i].u;
        int v=edge[i].v;
        int w=edge[i].w;
        int t1=find(u);
        int t2=find(v);
        if(t1!=t2){
            ans+=w;
            F[t1]=t2;
            cnt++;
        }
        if(cnt==n-1) break;
    }
    if(cnt<n-1) return -1;//不连通
    else return ans;
}
int main()
{
    int n,m;cin>>n>>m;//点数和边数
    for(int i=0;i<m;++i){
        int u,v,w;cin>>u>>v>>w;
        addedge(u,v,w);//只用读一次就行
    }
    cout<<Kruskal(n);//返回最小生成树权值
    return 0;
}
```

3.2 单源最短路

3.2.1 SPFA

```cpp
/* 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);
        }
      }
    }
  }
}
```

4 其他

4.1 输入输出

4.1.1 快读

```
//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

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

4.1.3 7.19 所用板子

```
#include<bits/stdc++.h>
using namespace std;
#define ll  long long
#define ull unsigned long long
#define ld long double
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;
}
ll mod=1e9+7;
ll INF=1e15;
ll Inf=0x3f3f3f3f;
double pi=acos(-1.0);


```

```
int  main()
{



    return 0;
}
```

4.2 高精度

```
//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);
    }
    BigInt(const char s[]){
        memset(a,0,sizeof(a));
        int L=strlen(s);
        len=L/DLEN;
        if(L%DLEN)  len++;
        int index = 0;
        for(int i=L-1;i>=0;i-=DLEN){
            int t=0;
            int k=i-DLEN+1;
            if(k<0) k=0;
            for(int j=k;j<=i;++j)
                t=t*10+s[j]-'0';
            a[index++]=t;
        }
    }
    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

- 注意范围!!! 爆 int 多少回了? 有时候 ans 定义为 ll 也是不够的, 注意改为 ll 后 scanf 和 printf, 不行就 signed main,#define int ll
- 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 $n = 1$