# 算法上机TRICK

## 在写代码前看的

### 关于初始化

```
#include <string>  //有的版本不需要
memset(a,0,sizeof(a));  //初始化为0
memset(b,-1,sizeof(b)); //初始化为-1
memset(c,0x3f,sizeof(c)); //初始化int型数组为INF
```

### 关于输入

```
while(cin>>a){
    //code here
}
while(~scanf("%d",&a)){
    //code here
}
```

### 文件读取

如果输入数据特别多，可以在.cpp目录下新建文件‘in.txt’,将输入数据输入进文件中。

```
int main(){
    //freopen("in.txt","r",stdin);
    return 0;
}
```

在提交前记得注释掉 `freopen("in.txt","r",stdin);`

### 优先队列

```
#include <queue>
```

创建优先队列

```
priority_queue <int> q;
priority_queue <int,vector<int>,greater<int> >; //注意这里最后的两个>不能写在一起，要有个空格。产生一个最小值在顶的堆。
priority_queue <int,vector<int>,less<int> >; //产生最大堆
```

函数--详细见[priority_queue C++ reference]

```
q.push();
q.pop();
q.top();
q.empty();
q.size();
```

## 关于结构体排序

```
如果给数组排序，只需要
#include <algorithm>
sort(a,a+n); //n是长度，排序范围是(a,a+n]
如果对结构体，如何排序?
struct Node
{
    int a,b,w;
    bool operator < (const Node & ano) const
    {
        return w < ano.w;
    }
}node[100];
sort(node,node+100); //从小到大排序，想从大到小改成w>ano.w即可
```

## 大数

超过int的数字类型，用long long

```
long long a;
cout<<a;
```

建议用cout，有的平台支持 `printf("%lld", a)` 。有的平台支持 `"%I64d"`

## 输出四舍五入

```
double ans;
ans = solve();
cout<< int(ans+0.5) <<endl;
```

### 关于构造图

数据量小，用邻接矩阵存图即可。 数据量>=1000则需要用链式结构存储。 下面提供一种我用的建图的方法：**仅供参考！**

最好能找一种自己喜欢的建图方法，毕竟网络流仍然需在图里跑=。=

```
const int maxn = 1e3+10;    //点数
const int maxm = 1e5+10;    //边数
int node[maxn]; //每个点的 边链表的 第一条边的编号,初始化为-1
int top;    //边数，需要初始化为0
//定义边
struct Side{
    int v,next,w; //边的另一个端点，这条边的下一条边，变的权值
}side[maxm];
//加边
void add_side(int u,int v,int w)
{
    side[top]=(Side){v,node[u],w};
    node[u]=top++;
}
int main(){
    //图的初始化
    memset(node,-1,sizeof(node));
    top=0;
    //给定一个点u，遍历该点上的边
    cin>>u;
    for(int i=node[u];i!=-1;i=side[i].next){
        int v = side[i].v;
        int w = side[i].w;
        //code...
    }
    return 0;
}
```

# 在你写完作业作业之后看的

会写一个最短路的算法，能够跑所有的最短路题不超时。推荐spfa，或堆优化的dijk。 附上spfa代码，亲测所有最短路的题都不会超时

```
//spfa 最短路S,E距离
int dis[maxn];
bool inqueue[maxn];
int spfa(int S, int E){
    memset(dis,0x3f,sizeof(dis));//dis是double类型的话要用for循环
    memset(inqueue,false,sizeof(inqueue));
    dis[S]=0;
    queue<int> q;
    q.push(S);
    while(!q.empty()){
        int u=q.front(); q.pop();
        inqueue[u]=false;
        for(int i=node[u];i!=-1;i=side[i].next){
            int v= side[i].v;
            if(dis[v]>dis[u]+side[i].w){
                dis[v] = dis[u]+side[i].w;
                if(!inqueue[v]){
                    q.push(v);
                    inqueue[v]=true;
                }
            }
        }
    }
    return dis[E];
}
```

***For Dorm 427***

***By your lovely yzs***