# 今日学习内容：

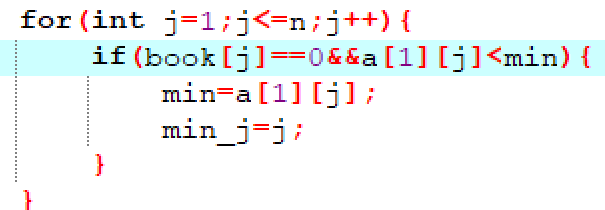
# Dijkstra算法（运用边的松弛）

背景：在上节求任意两顶点直接最短距离这种多源最短路径问题时，我们通常会使用floyd-warshall算法，但是若是我们只需要求得某个点到其他各点的距离这种单源最短路径问题时，我们若是还选用floyd-warshall算法的话，会造成时间复杂度过大，floyd-warshall算法的时间复杂度为O(N3)但是很明显，我们是可以降低它的时间复杂度到O（N2），这时候我们就需要dijkstra算法来降低其时间复杂度了。

原理：我们当寻求一个点到其他各点之间的最短距离时，我们可以先寻找离这个单点距离最近的一个点，这样我们可以确定下来这两个点之间最短路径，然后，和这个点可以直接相连的几个点目前的最短路径都可以确定下来，然后接着，我们沿着刚刚确定的最近的一个点往下走，那么从这个点出发可以直接连通的点的路径可以确定，这时候就可以比较更新第一个点到其他点的距离了，循环往复，就可以确定下来所有的点的最短路径。

那么根据以上原理，我们先设置最小值是99999以确保能进行打擂台算法：

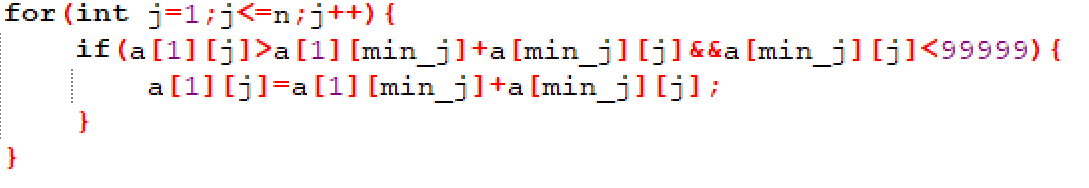
接着我们就要寻找没走过的最近的点了：



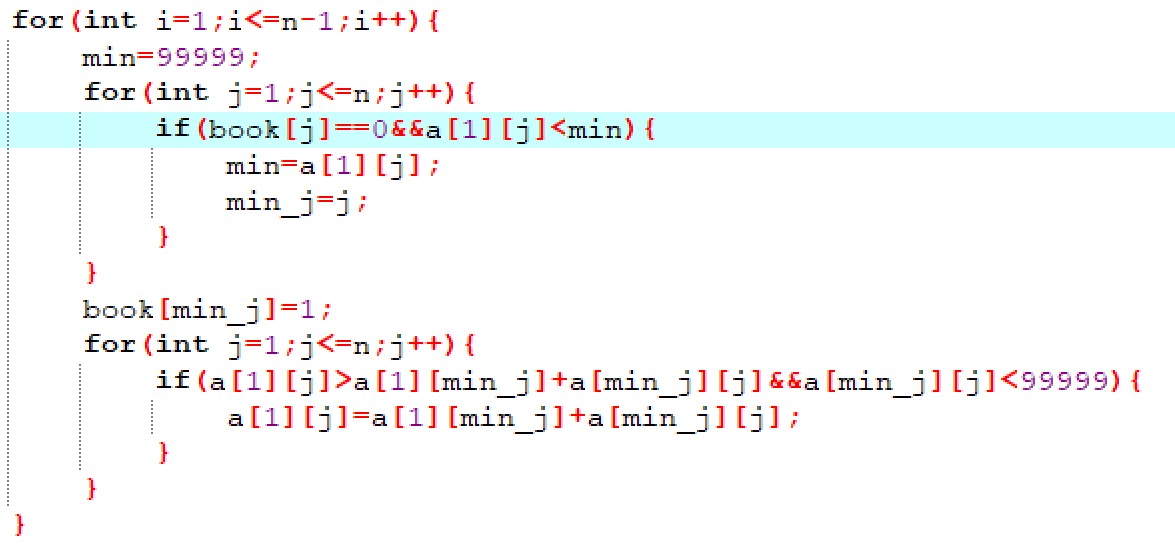
然后给这个点做好标记，表示已经走过了：



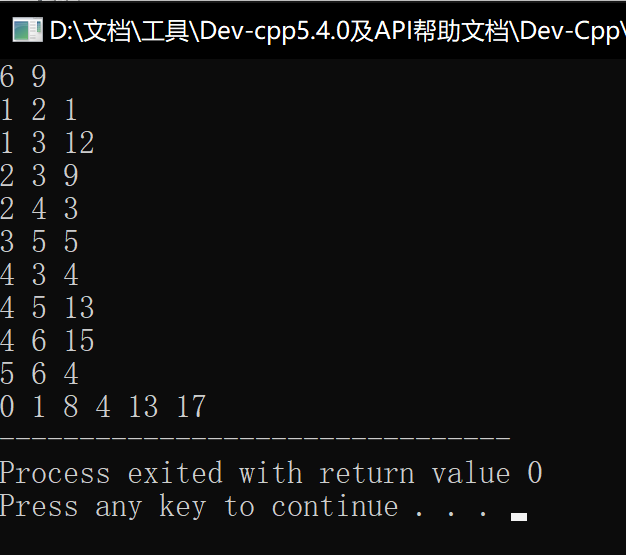
然后通过这个点更新起始点到各个点直接的最短路径，即进行“松弛”：



循环往复，这就是Dijkstra算法的核心：



最终代码运行结果如下：



最终代码如下：

#include<bits/stdc++.h>  
int main(){  
 int a[40][40];  
 int book[100]={0};  
 int p1,p2,p3;  
 int n,m,min,min\_j;  
 scanf("%d%d",&n,&m);  
 for(int i=1;i<=n;i++)  
 for(int j=1;j<=n;j++){  
 if(i==j) a[i][j]=0;  
 else a[i][j]=99999;  
 }  
 for(int i=1;i<=m;i++){  
 scanf("%d%d%d",&p1,&p2,&p3);  
 a[p1][p2]=p3;  
 }   
 book[1]=1;   
 for(int i=1;i<=n-1;i++){  
 min=99999;  
 for(int j=1;j<=n;j++){  
 if(book[j]==0&&a[1][j]<min){  
 min=a[1][j];  
 min\_j=j;  
 }  
 }  
 book[min\_j]=1;  
 for(int j=1;j<=n;j++){  
 if(a[1][j]>a[1][min\_j]+a[min\_j][j]&&a[min\_j][j]<99999){  
 a[1][j]=a[1][min\_j]+a[min\_j][j];  
 }  
 }  
 }  
 for(int i=1;i<=n;i++){  
 printf("%d ",a[1][i]);  
 }  
}   
*/\*  
6 9  
1 2 1  
1 3 12  
2 3 9  
2 4 3  
3 5 5  
4 3 4  
4 5 13  
4 6 15  
5 6 4  
\*/*

对于图的存储