南 京 师 范 大 学

《数据结构》

课程设计报告

题　　目：排序算法&管道铺设施工的最佳方案选择

学　　院：计算机科学与技术学院

专　　业：软件工程

姓　　名：刘明玉

学　　号：19160209

任课教师：陈波

计算机科学与技术学院　　制

二〇一八年二月二十八日

目录

1. 实验环境.....2
2. 实验内容.....2
2.1. 必做题：排序算法.....2
2.1.1. 课程设计要求.....2
2.1.2. 算法思想.....2
2.1.3. 程序结构图.....4
2.1.4. 运行结果与分析.....4
2.1.5. 问题与解决.....8
2.2. 选做题：管道铺设施工的最佳方案选择.....10
2.2.1. 课程设计要求.....10
2.2.2. 算法思想.....10
2.2.3. 程序结构图.....12
2.2.4. 运行结果.....12
2.2.5. 问题与解决.....16
3. 心得体会.....16

1. 实验环境

硬件：win7 旗舰版
软件：vs2015

2. 实验内容

2.1. 必做题：排序算法

2.1.1. 课程设计要求

编程实现希尔、快速、堆排序、归并排序算法。要求首先随机产生 10,000 个数据存入磁盘文件，然后读入数据文件，分别采用不同的排序方法进行排序，并将结果存入存入文件中。

2.1.2. 算法思想

希尔排序：

希尔排序是对直接插入排序的一种改进，它利用了插入排序的两个性质：

1.若待排序记录按关键字值基本有序，则直接插入的效率很高；

2.若待排序记录个数较少，则直接插入排序效率也较高。

因此，希尔排序先将待排序列划分为若干个小序列，在这些小序列中进行插入排序；然后逐步扩大小序列的长度，减少小序列的个数，这样使待排序列逐渐处于更有序的状态；最后对全体序列进行一次直接插入排序，从而完成排序。

为了使整个序列逐步向基本有序发展，子序列的构成不能简单地铸锻分割，而是将相距某个“增量”的记录组成一个子序列，这样才能有效地保证在子序列内分别进行直接插入排序后得到的结果是基本有序而不是局部有序。增量 d 逐步减小，刚开始每个子序列中的记录个数较少，并且提供了记录跳跃移动的可能，排序效率较高；后来增量逐步缩小，每个子序列的记录个数增加，但已基本有序，效率也较高。

为了在子序列中进行直接插入排序，在每个子序列中，将待插入记录和同一子序列中的前一个记录比较。在插入记录 r[i]时，自 r[i-d]起往前跳跃式查找待插入位置。在查找过程中，记录后也是跳跃 d 个位置。

快速排序：

快速排序的基本思想是从带排序记录序列中选取一个记录（通常选取第一个记录）为枢

轴，其关键字值设为 k。将关键字值小于 k 的记录移到前面，大于 k 的移到后面。结果将待排序记录序列分成两个子表，最后将关键字值 k 的记录查到分界线处，对划分后的子表按照上述原则进行划分，直到所有子表的表长不超过 1。

利用递归：

1.取第一个记录的关键字值作为基准，存于 temp，设 i、j 分别指向最左、最右记录的位置；

2.将 j 指向的记录关键字值与基准进行比较，直到找到比基准值小的记录，若 i<j，则将 j 指向的记录移动到 i 所指位置；

3.将 i 指向的记录关键字值与基准进行比较，直到找到比基准值小的记录，若 i<j，则将 i 指向的记录移动到 j 所指位置；

4.重复步骤 2、3，直到 i=j。

堆排序：

选择排序的改进，充分利用上一趟排序的调整结果。

堆排序是利用堆的特性进行排序的方法，其基本思想是首先用待排序的记录序列构造成一个堆，选出此堆中的最大者，即堆顶记录；然后将她从堆中移走（把堆顶记录和堆中最后一个记录交换），并将剩余的记录再调整成堆，这样又找出了此大的记录；以此类推，直到堆中只有一个记录。

归并排序：

一次归并：

两个有序序列，三个参数分别指向两个序列的当前位置和新序列的最后一个记录，分别比较两个有序序列的当前值，小的放入新序列，参数记录的位置后移，直到一个序列结束，将另一个序列剩下的值放入新序列中。

一趟归并：

设一次归并的长度为 h，则两个序列可能出现：

两个序列都是完整的（长度都为 h）；

一个完整一个不完整（一个为 h，一个小于 h）；

只剩下一个不完整的（小于 h）；

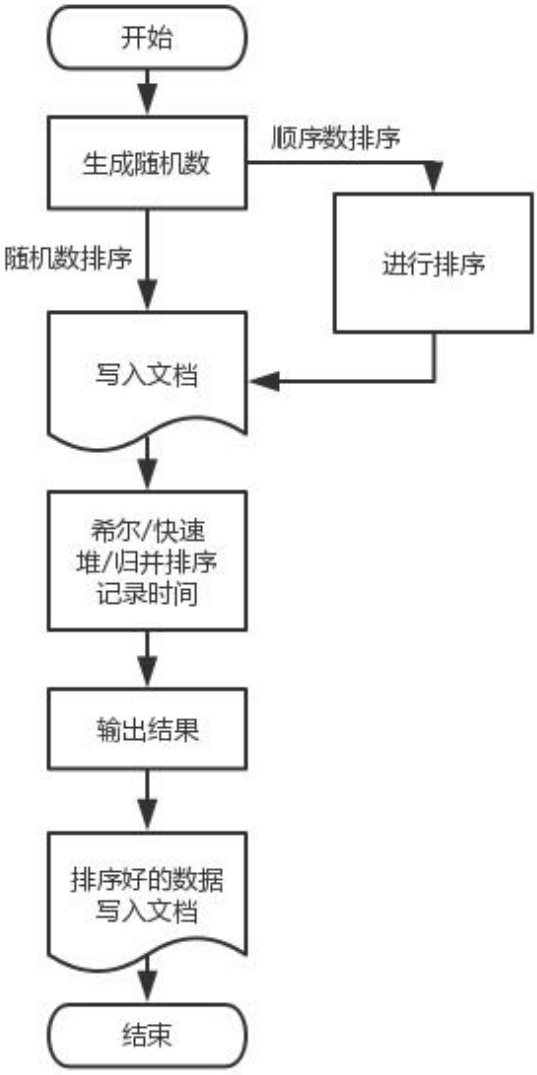
三种情况，对三种情况处理后进行排序。

归并排序：

两个有序序列 r、r1，将 r，r1 分别看做新序列，长度 h 依次成倍增加，直到排序完成。

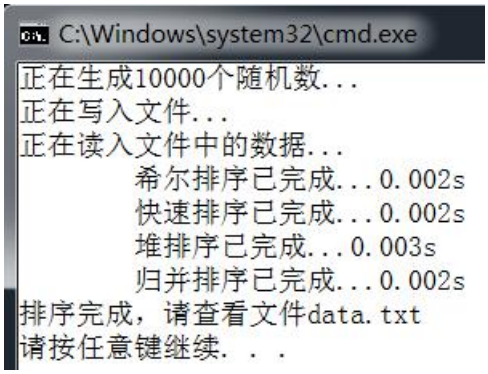
2.1.4. 程序结构图

主程序：



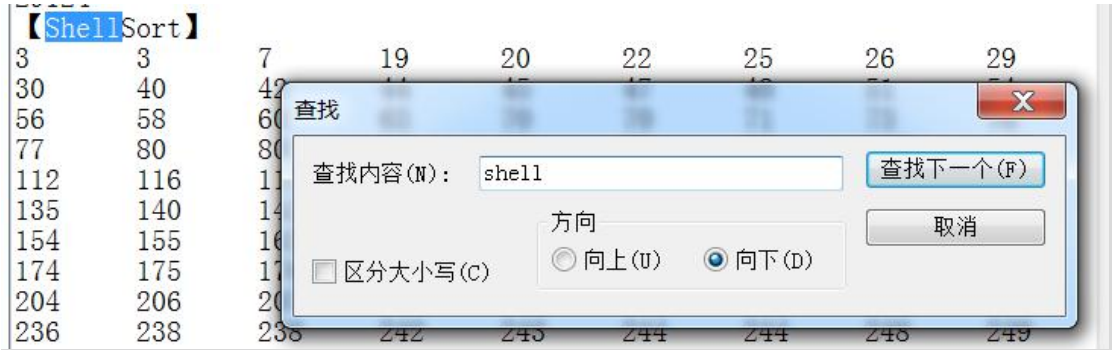
2.1.6. 运行结果与分析

测试 随机生成 1w 个随机数：



随机数:

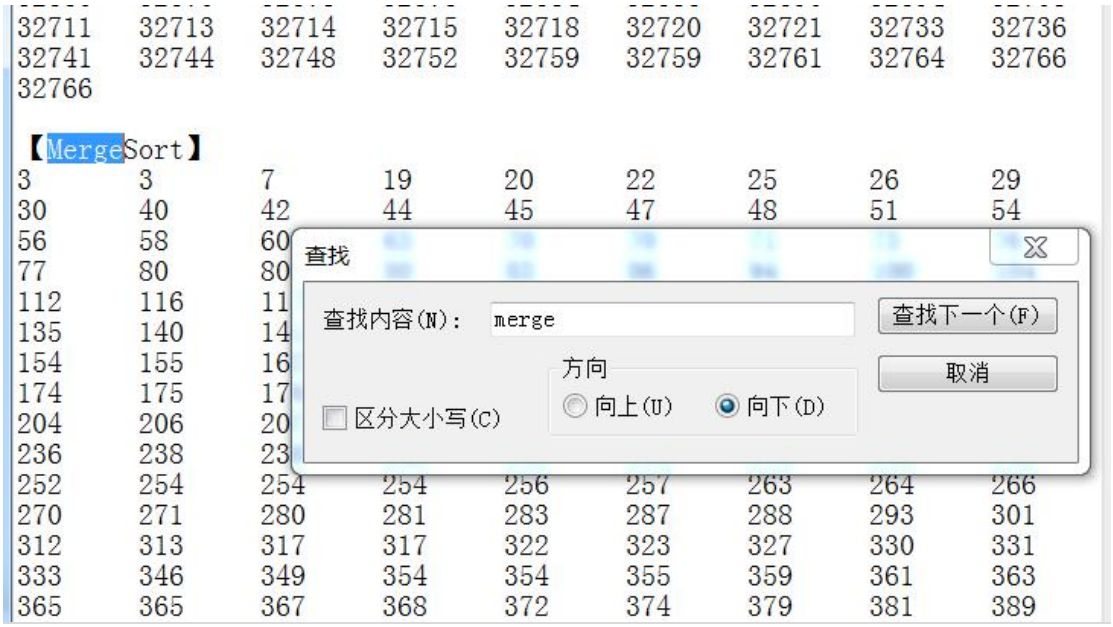
希尔排序结果：



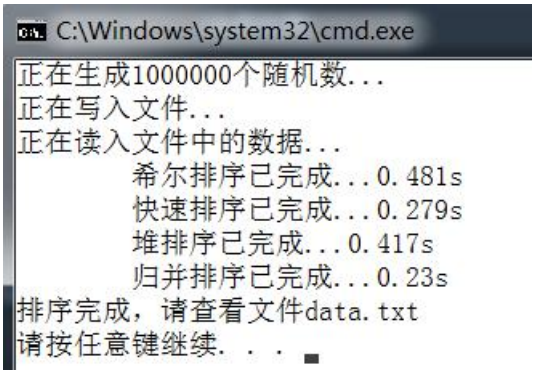
快速排序结果（接希尔）：

堆排序结果（接快排）：

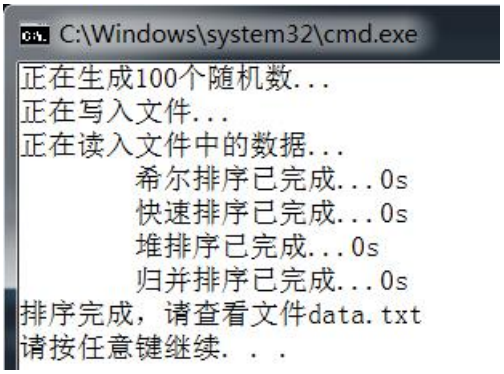
归并排序结果（接堆排序）：



测试 2：随机生成 100w 个随机数：



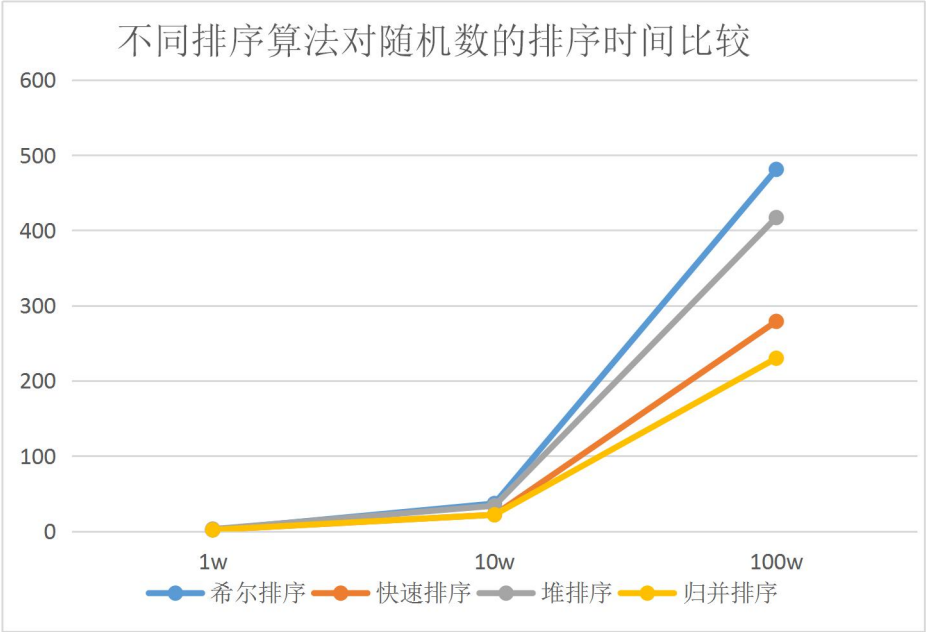
测试 3 随机生成 100 个随机数：

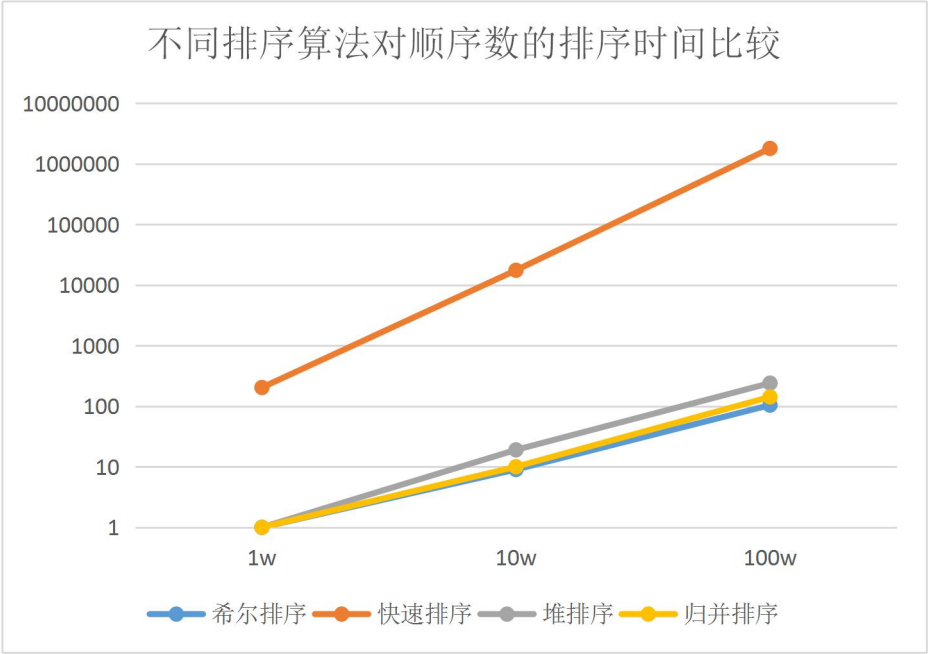


2.1.7. 性能分析

（单位 万/毫秒）

	随机数			有序数		
	1	10	100	1	10	100
希尔排序	2	37	481	1	9	104
快速排序	2	22	279	203	17488	1793750
堆排序	3	34	417	1	19	240
归并排序	2	22	230	1	10	142





总体来看，当数据较少时，四种排序算法的对随机数的排序效率差不多，数据越多差距越明显，效率从高到低依次为：归并>快速>堆>希尔；已排好的数据对快速排序来说是最不利的情况，所以所需时间较多效率低下，其他三种算法效率相差不大，从高到低依次为：希尔>归并>堆。

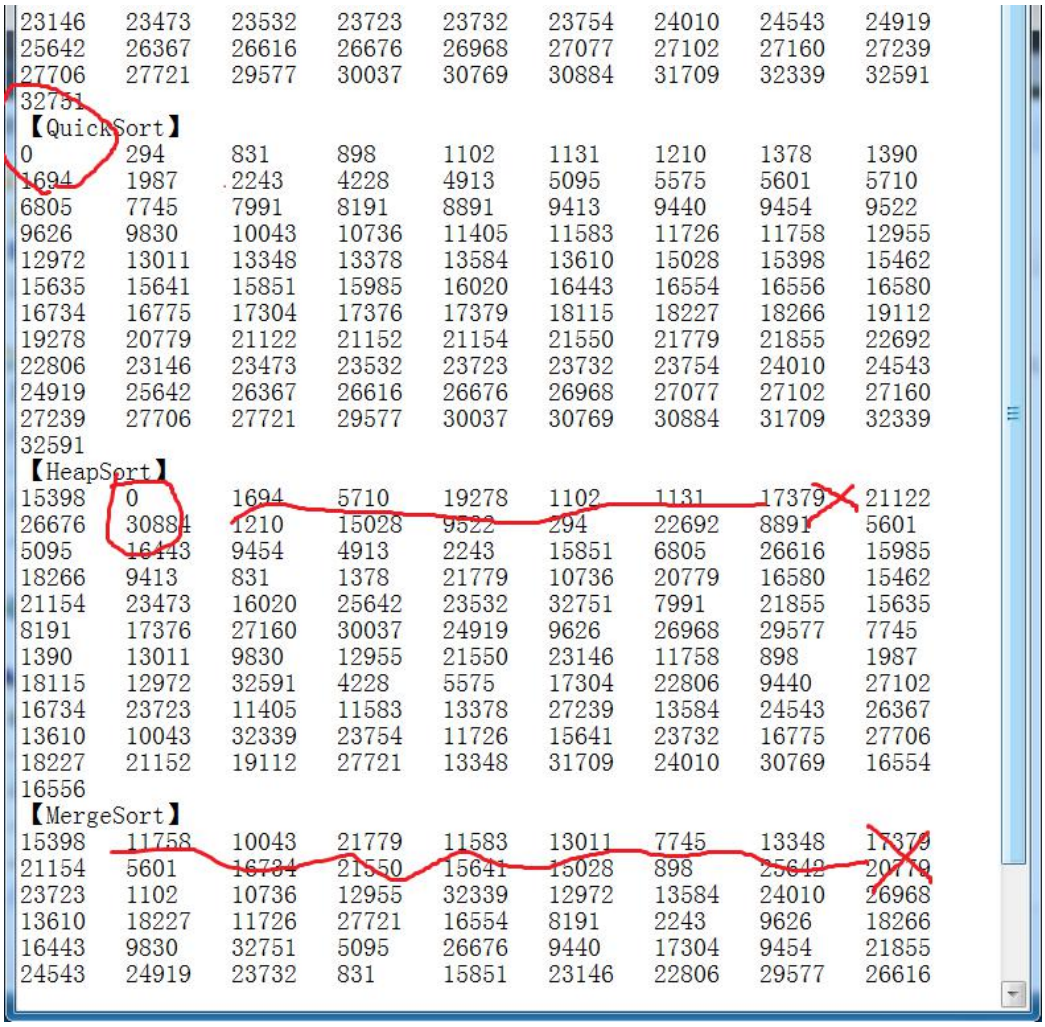
2.1.8. 问题与解决

问题 1：每次调试时生成的随机数都一样；

解决 1：加入语句“srand((unsigned)time(NULL));”，并添加头文件“time.h”“stdlib.h”；

问题 2：写到文件中的排序结果都是 0，注释掉写入文件的部分，直接从文件中读取数据则无误；
解决 2：打开文件后，操作完成的地方加入“in.close();”，关闭文件；

问题 3：除了希尔排序以外，其他三个排序结果都不正确、多了一个“0”；



解决 3：希尔排序、快速排序的数据存放在数组 0~n-1 中，堆排序、归并排序的数据则在数组 1~n 中；
堆排序中将 r[j]<r[j+1]写为 r[j]<r[i+1]；归并排序中将 j<=t 写为 j< -t，将 i<=n-2*h+1 写为 i<=n-2*h+i；

问题 4：调试时报错 Stack around the variable ‘heap’ was corrupted.
解决 4：堆排序、归并排序的数组大小没有改，改为 heap[10001] 、merge[10001] ；

问题 5：进行 100w 个随机数排序时，程序崩溃；
解决 5：将排序的数组都改为全局变量，因为在函数中数组为局部变量，储存在栈中，数组过大造成栈溢出；大数据的快速排序消耗大量栈空间，在编译器属性中修改工程栈大小。

2.2. 选做题：管道铺设施工的最佳方案选择

2.2.1. 课程设计要求

n（n>10）个居民区之间需要铺设煤气管道。假设任意两个居民区之间都可以铺设煤气管道，但代价不同。要求事先将任意两个居民区之间铺设煤气管道的代价存入磁盘文件中。设计一个最佳方案使得这 n 个居民区之间铺设煤气管道所需代价最小，并将结果以图形方式在屏幕上输出。

2.2.2. 算法思想

直接套用“图”（MGraph）类中的最小生成树的算法，主程序中加入文件读入、图形输出即可。

最小生成树算法：

生成树：多一根线有回路，少一根线不连通。

最小生成树，也就是一张图中，每两点之间是有权值的，当权值之和小时，这棵树就是最小生成树。

两种最小生成树的算法：Prim 以及 kruskal。

Prim：

需要定义一个 miniedges 数组，元素类型为：

```
template <class T>
struct Edge
{
    T adjvex;
    float lowcost;
};
```

数组一开始存储的是输入顶点的元素内容以及该顶点到对应其他顶点的权值，通过循环，每次：

1.找到 miniedge 中不为 0 的最小值权值的编号，返回，记为 k；

2.上述两个之间对应的是最小边，把其权值置为 0，表明已经访问过，

3.下面修改值，以找到的第 k 个元素来修正 miniedges 参数，访问邻接矩阵中的第 k 行，把权值与 miniedges 中的权值进行比较，若更小，则替换 miniegdes 中的元素及其权值。否则不变。

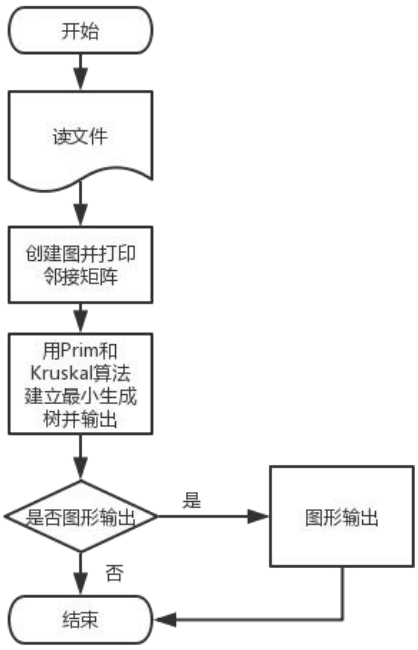
这样，一轮循环下来，会找到与已经知晓的点权值最小的点；按照这个方法，只要找全所有的点，就生成了最小树。

Kruskal：

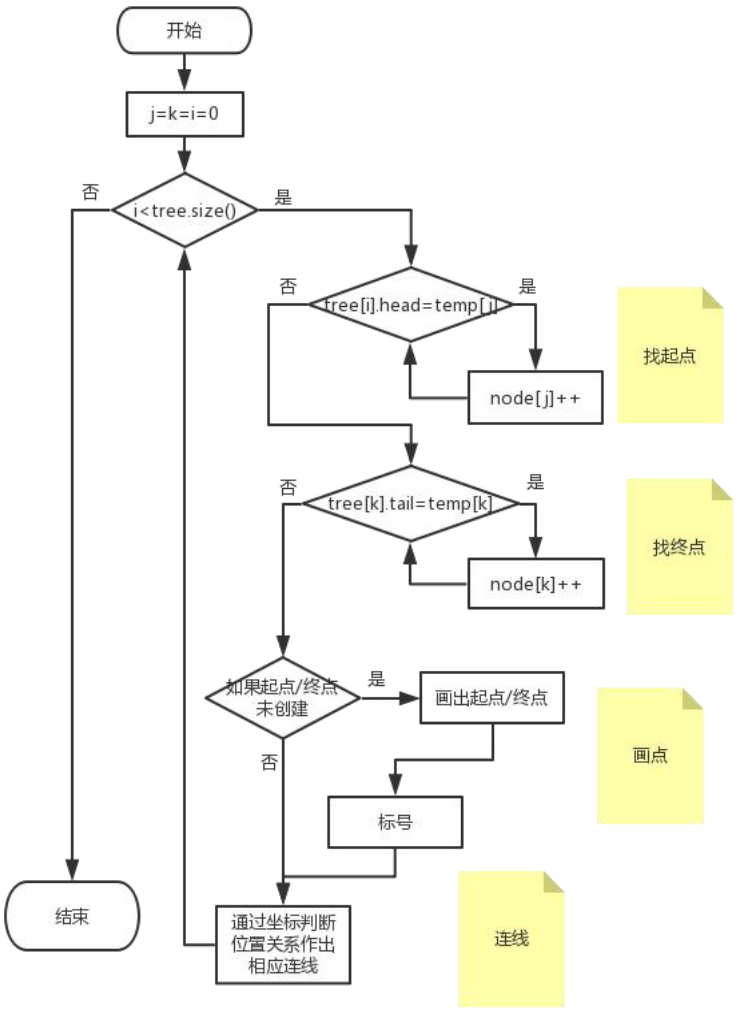
每次找最小权值的边连起来。先要把所有边包含顶点的信息放到一个数组或者向量里面，并进行排序，注意，要选取的是不构成回路的最小边，所以需要一个辅助数组来判断不构成回路：components 数组。通过统一编号判断是否形成回路。

2.2.3. 程序结构图

主程序

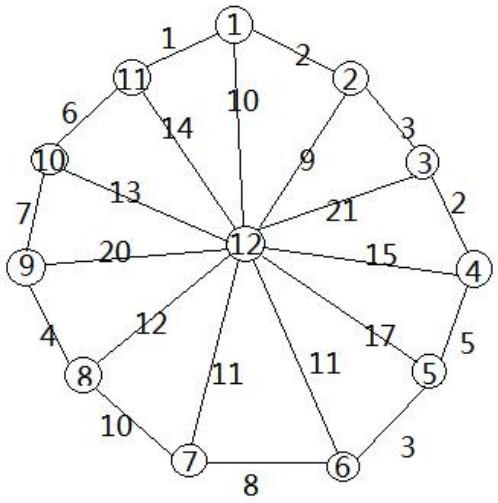


图形化：

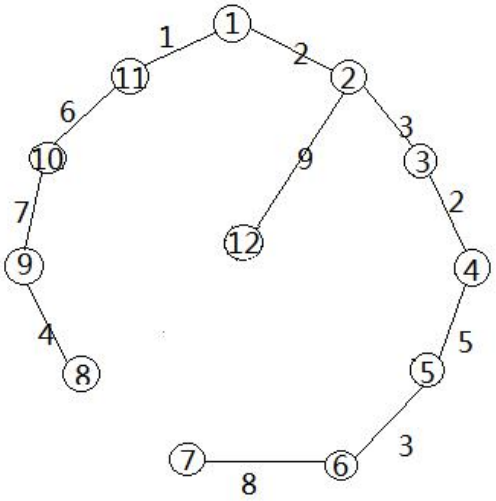


2.2.5. 运行结果

参考图



连通图



最小生成树

data.txt

```
小刘 小王 2
小王 小李 3
小李 小亮 2
小亮 小赵 5
小赵 小张 3
小张 小徐 8
小徐 小陈 10
小陈 小明 4
小明 小强 7
小强 小唐 6
小唐 小刘 1
小刘 小k 10
小王 小k 9
小李 小k 21
小亮 小k 15
小赵 小k 17
小张 小k 11
小徐 小k 11
小陈 小k 12
小明 小k 20
小强 小k 13
小唐 小k 14
```

测试：

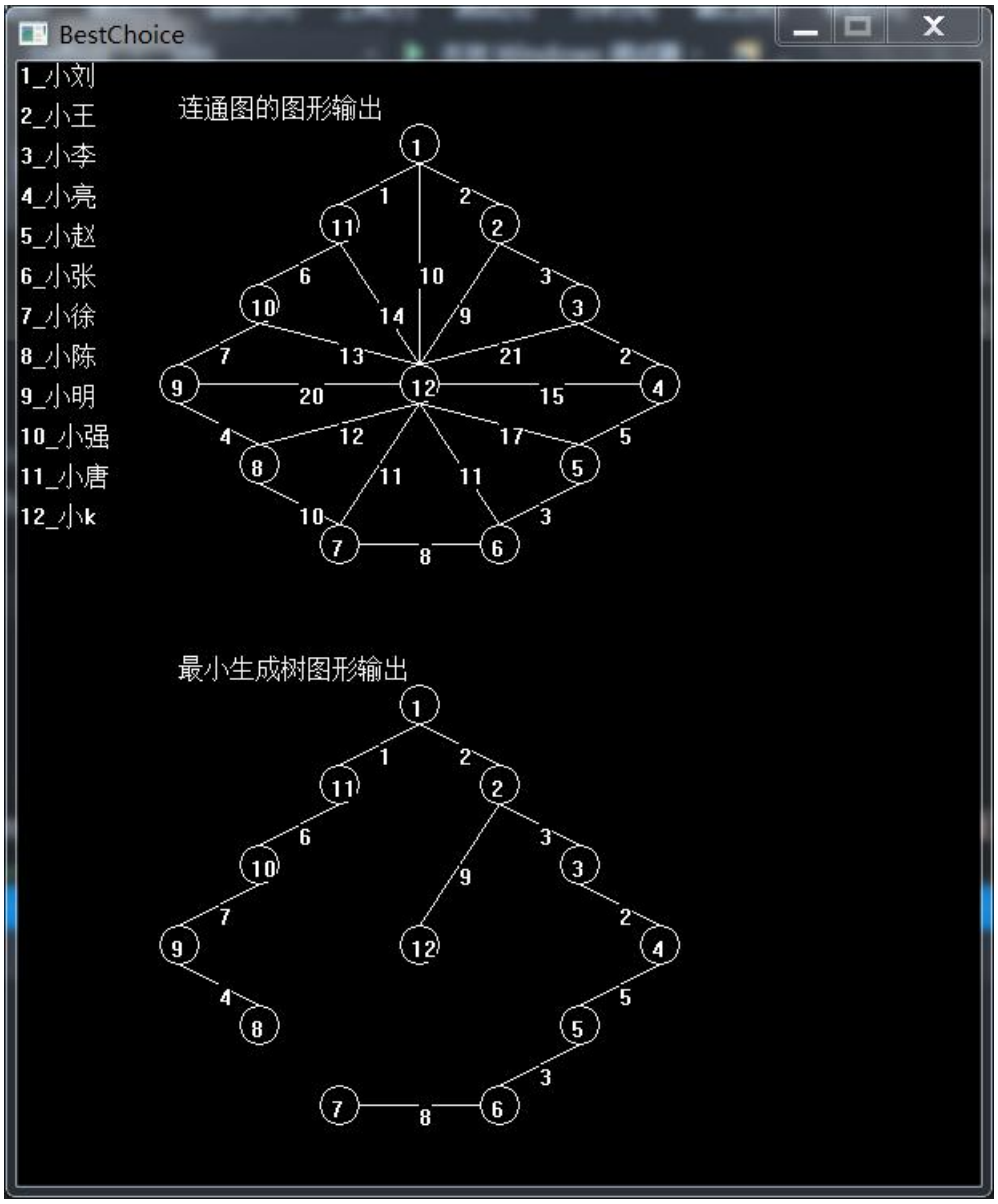
C:\Windows\system32\cmd.exe

```
打印：
**************************打印****************************
        小刘    小王    小李    小亮    小赵    小张    小徐    小陈    小明    小强    小唐    小k
小刘|   0       2       ∞       ∞       ∞       ∞       ∞       ∞       ∞       ∞       1       10
小王|   2       0       3       ∞       ∞       ∞       ∞       ∞       ∞       ∞       ∞       9
小李|   ∞       3       0       2       ∞       ∞       ∞       ∞       ∞       ∞       ∞       21
小亮|   ∞       ∞       2       0       5       ∞       ∞       ∞       ∞       ∞       ∞       15
小赵|   ∞       ∞       ∞       5       0       3       ∞       ∞       ∞       ∞       ∞       17
小张|   ∞       ∞       ∞       ∞       3       0       8       ∞       ∞       ∞       ∞       11
小徐|   ∞       ∞       ∞       ∞       ∞       8       0       10      ∞       ∞       ∞       11
小陈|   ∞       ∞       ∞       ∞       ∞       ∞       10      0       4       ∞       ∞       12
小明|   ∞       ∞       ∞       ∞       ∞       ∞       ∞       4       0       7       ∞       20
小强|   ∞       ∞       ∞       ∞       ∞       ∞       ∞       ∞       7       0       6       13
小唐|   1       ∞       ∞       ∞       ∞       ∞       ∞       ∞       ∞       6       0       14
小k|    10      9       21      15      17      11      11      12      20      13      14      0
顶点数:12个
边数:22条

最小生成树（Prim）：
小刘-->小唐
小刘-->小王
小王-->小李
小李-->小亮
小亮-->小赵
小赵-->小张
小唐-->小强
小强-->小明
小明-->小陈
小张-->小徐
小王-->小k
```

```
最小生成树（Kruskal）：
小刘-->小唐
小刘-->小王
小李-->小亮
小王-->小李
小赵-->小张
小陈-->小明
小亮-->小赵
小强-->小唐
小明-->小强
小张-->小徐
小王-->小k
是否要求图形输出？y/n
在图形输出界面按任意键+回车键返回。
y
```

回车后图形输出：



按任意键+回车后结束程序：

```
是否要求图形输出？ y/n
在图形输出界面按任意键+回车键返回。
y

y
请按任意键继续. . .
```

若不需图形输出则直接结束：

```
是否要求图形输出？ y/n
在图形输出界面按任意键+回车键返回。
n
```

2.2.6. 问题与解决

问题 1：图形输出的窗口一闪而过就关闭了；
解决 1：通过一个键盘输入使此窗口显示；

问题 2：图形化输出不美观；
解决 2：调整坐标参数、加入判断语句；

问题 3：图形化输出错误，圆圈（代表居民）有误；
解决 3：通过数组记录生成树的点的次数，只有次数为 1 时才在屏幕输出圆圈；

问题 4：outtextxy 只能输出字符串，图形化时无法定位输出数字；
解决 4：参考文档，通过_stprintf_s 函数将数字格式化为字符串。

3. 心得体会

1.关于时间函数：srand()就是给 rand()提供种子 seed，这个种子会对应一个随机数，如果使用相同的种子后面的 rand()函数会出现一样的随机数。通过 srand（time（NULL））使得随机数种子随时间的变化而变化。

PS：time 函数可以获取当前的系统时间，返回的结果是一个 time_t 类型，其实就是一个大整数，其值表示从 CUT（Coordinated Universal Time）时间 1970 年 1 月 1 日 00:00:00（称为 UNIX 系统的 Epoch 时间）到当前时刻的秒数。

2.关于文件读写：打开文件后要关闭文件，写入文件的内容才会保存；一般写入文件的代码是从文件开头写入的，覆盖清空原有内容，如果要接着原有内容写入，可以使用：

```
ofstream out("data.txt", ios::app);   //ios::app 表示在原文件末尾追加
```

3.关于栈：结论——任何函数里面不要开大数组。

原因——写在 main()外面的变量叫做全局变量，分配在静态数据区。

new 出来的变量叫动态变量，分配在堆上。

函数里面的变量叫局部变量，分配在栈上。

前两者基本只受你的内存大小影响，想开多大开多大。栈空间是和操作系统以及编译器有关。

在百万级的数据排序量时，发现程序很快崩溃，把数组设为全局变量得以解决，但快速排序依然在万级的数据量时就崩溃了，搜索后发现快排由于程序递归次数太多，大量的压栈使程序占用的栈空间超过了操作系统所规定的大小，从而出现的内存错误，后将属性设置的工程栈大小修改得以解决。

4.关于画图：

为了图形化输出，查找了一些关于 EasyX 的内容：为了 VC 方便的开发平台和 TC 简单的绘图功能，于是就有了这个 EasyX 库，只需要包含两个头文件即可：

```
#include <graphics.h>        // 引用图形库头文件
```

```
#include <conio.h>
```

只用到一些简单的绘图函数就可基本实现需求，并且不用考虑册窗口类、建消息循环等等，上手方便。

通过阅读文档我使用了以下函数：

```
initgraph(640, 480);   // 创建绘图窗口，大小为 640x480 像素

circle(200, 200, 100);  // 画圆，圆心(200, 200)，半径 100
line(10,10,20,20);      //从坐标(10,10)到(20,20)画直线

// 输出数值 1024，先将数字格式化输出为字符串 (VC2008 / VC2010 / VC2012)
TCHAR s[5];
_stprintf(s, _T("%d"), 1024);          // 高版本 VC 推荐使用 _stprintf_s 函数
outtextxy(10, 60, s);      //在坐标(10,60)处输出字符串

closegraph();              // 关闭绘图窗口
```