软工23C1《数据结构与算法》- 图论和排序

开始时间 2024/05/28 15:26:00　　结束时间 2024/07/08 17:26:00　　答题时长 59160分钟

答卷类型 标准答案　　试卷总分 210

判断题　　26 分

1-1 对N个记录进行快速排序，在最坏的情况下，其时间复杂度是$O(NlogN)$。　　1分

○ T　　● F

| 参考答案

答案　F

1-2 如果无向图G必须进行两次广度优先搜索才能访问其所有顶点，则G一定有2个连通分量。　　1分

● T　　○ F

| 参考答案

答案　T

1-3 Prim 算法是通过每步添加一条边及其相连的顶点到一棵树，从而逐步生成最小生成树。　　1分

● T　　○ F

| 参考答案

答案　T

1-4 无向图中的一条边，在其邻接表存储结构中对应两个弧结点。　　1分

● T　　○ F

| 参考答案

答案　T

1-5 最小生成树是指边数最少的生成树。　　1分

○ T　　● F

| 参考答案

答案　F

1-6 若图G为连通图，则G必有唯一的一棵最小生成树。　　1分

○ T　　● F

| 参考答案

答案 F

1-7 若图G有环，则G不存在拓扑排序序列。 1分

T F

| 参考答案

答案 T

1-8 拓扑序一定是唯一的。 1分

T F

| 参考答案

答案 F

1-9 有向图的邻接矩阵一定是不对称的 1分

T F

| 参考答案

答案 F

1-10 最小生成树的Kruskal算法是一个贪心法。 1分

T F

| 参考答案

答案 T

1-11 图的深度优先遍历非递归算法通常采用栈实现，广度优先遍历非递归算法通常采用队列实现。 1分

T F

| 参考答案

答案 T

1-12 Kruskal 算法是通过每步添加一条边及其相连的顶点到一棵树，从而逐步生成最小生成树。 1分

T F

| 参考答案

答案 F

1-13 一个无向图G，若某顶点v到其它每个顶点都有至少一条路径，则图G只有1个连通分量。 1分

T F

| 参考答案

答案 T

1-14 如果无向图G必须进行两次广度优先搜索才能访问其所有顶点，则G中一定有回路。 1分

○ T ◉ F

| 参考答案

答案 F

1-15 在一个有权无向图中，若 b 到 a 的最短路径距离是12，且 c 到 b 之间存在一条权为2的边，则 c 到 a 的最短路径距离一定不小于10。 1分

◉ T ○ F

| 参考答案

答案 T

1-16 希尔排序是稳定的算法。 1分

○ T ◉ F

| 参考答案

答案 F

1-17 无向连通图边数一定大于顶点个数减1。 1分

○ T ◉ F

| 参考答案

答案 F

1-18 用邻接表法存储图，占用的存储空间数只与图中结点个数有关，而与边数无关。 1分

○ T ◉ F

| 参考答案

答案 F

1-19 用邻接矩阵法存储图，占用的存储空间数只与图中结点个数有关，而与边数无关。 1分

◉ T ○ F

| 参考答案

答案 T

1-20 在一个有向图中，所有顶点的入度与出度之和等于所有边之和的2倍。 1分

◉ T ○ F

| 参考答案

答案 T

1-21 对N个记录进行简单选择排序，比较次数和移动次数分别为$O(N^2)$和$O(N)$。 1分

T F

| 参考答案

答案 T

1-22 快速排序是稳定的算法。 1分

T F

| 参考答案

答案 F

1-23 求解带有负边（权值为负数的边）的图的单源最短路径问题，迪杰斯特拉（Dijkstra）算法不适用。 1分

T F

| 参考答案

答案 T

1-24 无向连通图的最小生成树是唯一的。 1分

T F

| 参考答案

答案 F

1-25 若连通图上各边的权值均不相同，则该图的最小生成树是唯一的。 1分

T F

| 参考答案

答案 T

1-26 有向图G存在拓扑排序序列，则G必定无环且连通。 1分

T F

| 参考答案

答案 F

单选题 84 分

2-1 在用邻接表表示有N个结点E条边的图时，深度优先遍历算法的时间复杂度为： 2分

A. $O(N)$

B. $O(N+E)$

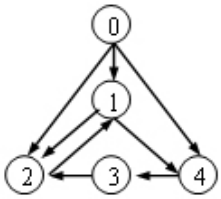
C. $O(N^2)$

D. $O(N^2 \times E)$

| 参考答案

答案	B

2-2 下面给出的有向图中，有__个强连通分量。 2分



A. 1 ({0,1,2,3,4})

B. 1 ({1,2,3,4})

C. 2 ({1,2,3,4}, {0})

D. 5 ({0}, {1}, {2}, {3}, {4})

| 参考答案

答案	C

2-3 下列关于无向连通图特征的叙述中，正确的是: 2分

1. 所有顶点的度之和为偶数
2. 边数大于顶点个数减1
3. 至少有一个顶点的度为1

A. 只有1

B. 只有2

C. 1和2

D. 1和3

| 参考答案

答案	A

2-4 对于有向图，其邻接矩阵表示比邻接表表示更易于: 2分

A. 求一个顶点的入度

B. 求一个顶点的出边邻接点

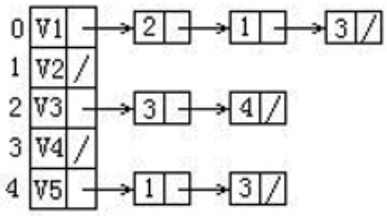
C. 进行图的深度优先遍历

D. 进行图的广度优先遍历

| 参考答案

答案	A

2-5 给定一有向图的邻接表如下。从顶点V1出发按广度优先搜索法进行遍历，则得到的一种顶点序列为: 2分



A. V1,V2,V3,V4,V5

B. V1,V2,V3,V5,V4

C. V1,V3,V2,V4,V5

D. V1,V4,V3,V5,V2

| 参考答案

答案 C

2-6 图的广度优先遍历类似于二叉树的: 2分

A. 层次遍历

B. 中序遍历

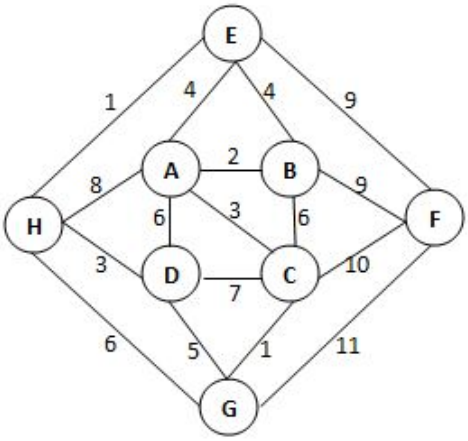
C. 后序遍历

D. 先序遍历

| 参考答案

答案 A

2-7 给定有权无向图如下。关于其最小生成树，下列哪句是对的? 2分



A. 最小生成树不唯一，其总权重为23

B. 最小生成树唯一，其总权重为20

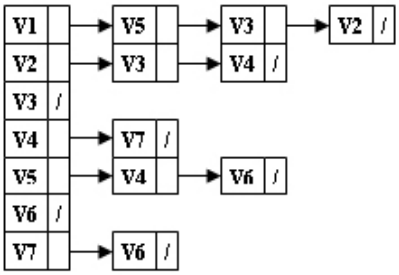
C. 边(B, F)一定在树中，树的总权重为23

D. 边(H, G)一定在树中，树的总权重为20

| 参考答案

答案 A

2-8 给定一有向图的邻接表如下。从顶点V1出发按深度优先搜索法进行遍历，则得到的一种顶点序列为: 2分



A. V1,V5,V4,V7,V6,V2,V3

B. V1,V5,V4,V7,V6,V3,V2

C. V1,V2,V3,V4,V7,V6,V5

D. V1,V5,V6,V4,V7,V2,V3

| 参考答案

答案	B

2-9 设无向图为 G=(V，E)，其中 V={v1,v2,v3,v4}，E={(v1,v2)，(v3,v4)，(v4,v1)，(v2,v3)，(v1,v3)}。则每个顶点的度依次为: 2分

A. 2, 1, 1, 1

B. 1, 1, 2, 1

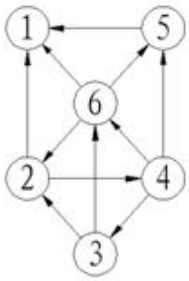
C. 3, 2, 3, 2

D. 2, 3, 2, 3

| 参考答案

答案	C

2-10 对于给定的有向图如下，其邻接矩阵为: 2分



A. $\begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$

B. $\begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$

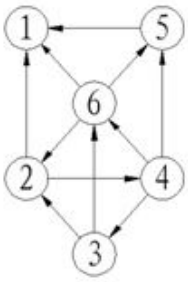
C. $\begin{bmatrix}0&0&0&0&0&0\\1&0&0&1&0&0\\0&1&0&0&0&0\\0&0&1&0&1&1\\1&0&0&0&0&0\\1&1&1&0&1&0\end{bmatrix}$

D. $\begin{bmatrix}0&0&0&0&0&0\\1&0&0&1&0&0\\0&1&0&0&0&1\\0&0&1&0&1&1\\1&0&0&0&0&0\\1&1&0&0&1&0\end{bmatrix}$

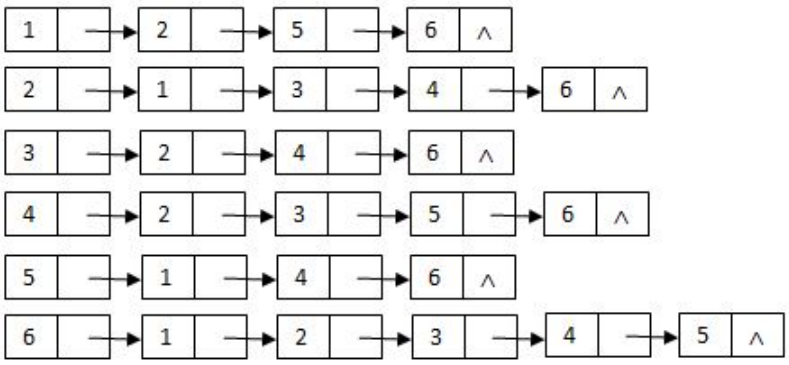
| 参考答案

答案	D

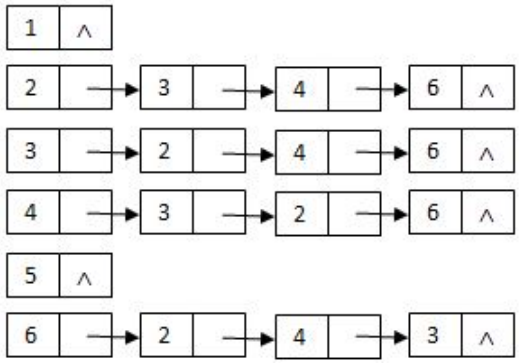
2-11 对于给定的有向图如下，其邻接表为： 2分



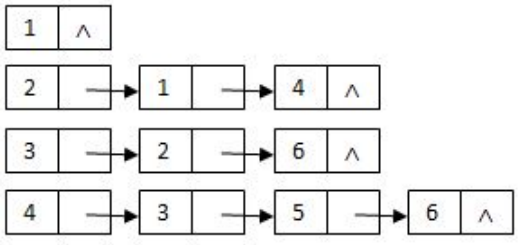
A.



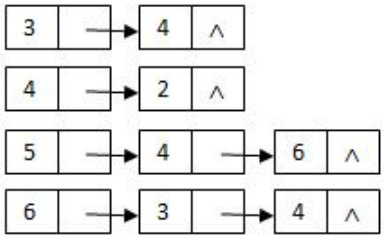
B.



C.



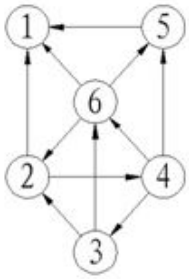
D.



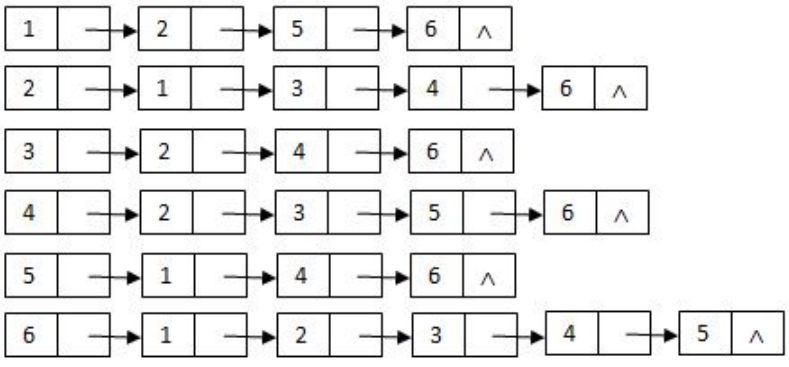
| 参考答案

答案	C

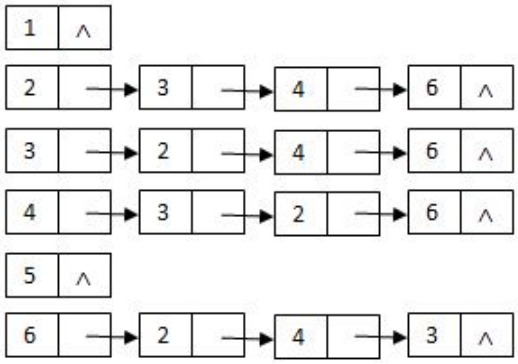
2-12 对于给定的有向图如下，其逆邻接表为： 2分



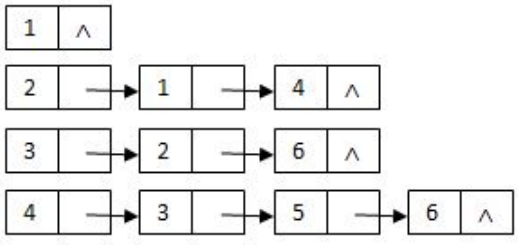
A.



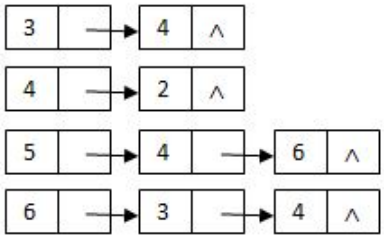
B.



C.



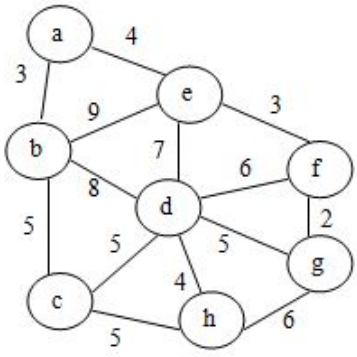
D.



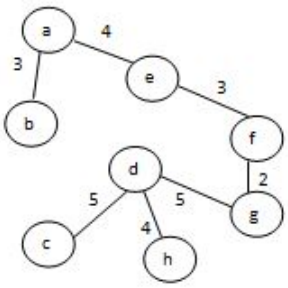
| 参考答案

答案	D

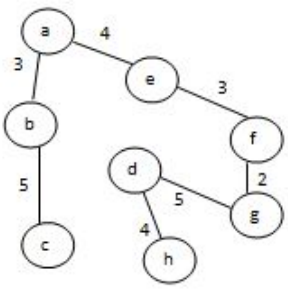
2-13 以下哪个**不是**给定无向带权图的最小生成树? 2分



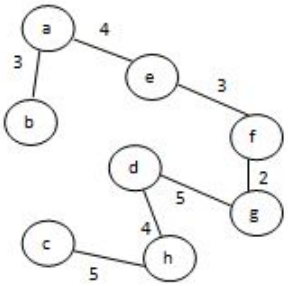
A.



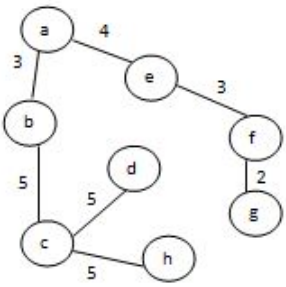
B.



C.



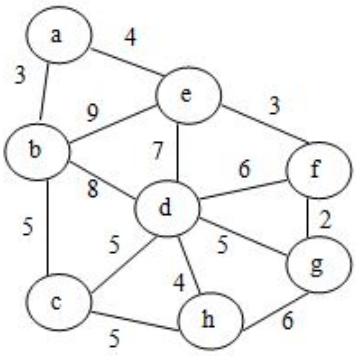
D.



| 参考答案

答案	D

2-14 给定无向带权图如下，以下哪个是从顶点 a 出发深度优先搜索遍历该图的顶点序列（多个顶点可以选择时按字母序）？ 2分



A. abecdfhg

B. abcdehgf

C. abcdefgh

D. abchgfde

| 参考答案

答案	C

2-15 给定一个图的邻接矩阵如下，则从V1出发的深度优先遍历序列（DFS，有多种选择时小标号优先）是: 2分

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
V1	0	1	1	1	0	0	0	0	0	0
V2	0	0	0	1	1	0	0	0	0	0
V3	0	0	0	1	0	1	0	0	0	0
V4	0	0	0	0	0	1	1	0	1	0
V5	0	0	0	0	0	0	1	0	0	0
V6	0	0	0	0	0	0	0	1	1	0
V7	0	0	0	0	0	0	0	0	1	0
V8	0	0	0	0	0	0	0	0	0	1
V9	0	0	0	0	0	0	0	0	0	1
V10	0	0	0	0	0	0	0	0	0	0

A. V1, V2, V4, V3, V6, V8, V10, V9, V7, V5

B. V1, V2, V3, V4, V5, V6, V7, V9, V8, V10

C. V1, V2, V4, V6, V8, V10, V9, V7, V5, V3

D. V1, V2, V3, V5, V7, V9, V10, V6, V8, V4

| 参考答案

答案	C

2-16 给定一个图的邻接矩阵如下，则从V1出发的宽度优先遍历序列（BFS，有多种选择时小标号优先）是: 2分

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
V1	0	1	1	1	0	0	0	0	0	0
V2	0	0	0	1	1	0	0	0	0	0
V3	0	0	0	1	0	1	0	0	0	0
V4	0	0	0	0	0	1	1	0	1	0
V5	0	0	0	0	0	0	1	0	0	0
V6	0	0	0	0	0	0	0	1	1	0
V7	0	0	0	0	0	0	0	0	1	0
V8	0	0	0	0	0	0	0	0	0	1
V9	0	0	0	0	0	0	0	0	0	1
V10	0	0	0	0	0	0	0	0	0	0

A. V1, V2, V4, V3, V6, V8, V10, V9, V7, V5

B. V1, V2, V3, V4, V5, V6, V7, V9, V8, V10

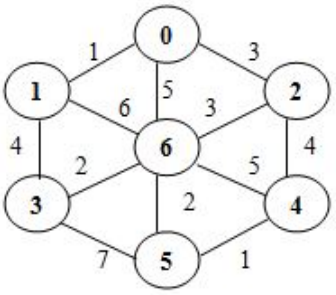
C. V1, V2, V4, V6, V8, V10, V9, V7, V5, V3

D. V1, V2, V3, V5, V7, V9, V10, V6, V8, V4

| 参考答案

答案	B

2-17 给出如下图所示的具有 7 个结点的网 G，哪个选项对应其正确的邻接矩阵? 2分



A. $\begin{bmatrix} \infty & 1 & 3 & \infty & \infty & \infty & 5 \\ 0 & \infty & \infty & 4 & \infty & \infty & 6 \\ 0 & \infty & \infty & \infty & 4 & \infty & 3 \\ \infty & 0 & \infty & \infty & \infty & 7 & 2 \\ \infty & \infty & 0 & \infty & \infty & 1 & 5 \\ \infty & \infty & \infty & 0 & 0 & \infty & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & \infty \end{bmatrix}$

B. $\begin{bmatrix} \infty & 1 & 3 & \infty & \infty & \infty & 5 \\ 1 & \infty & \infty & 4 & \infty & \infty & 6 \\ 3 & \infty & \infty & \infty & 4 & \infty & 3 \\ \infty & 4 & \infty & \infty & \infty & 7 & 2 \\ \infty & \infty & 4 & \infty & \infty & 1 & 5 \\ \infty & \infty & \infty & 7 & 1 & \infty & 2 \\ 5 & 6 & 3 & 2 & 5 & 2 & \infty \end{bmatrix}$

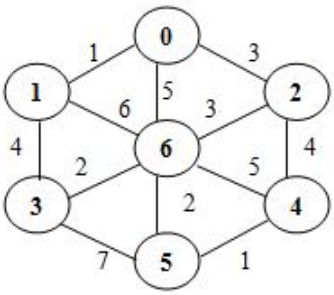
C. $\begin{bmatrix} 0 & 1 & 3 & 0 & 0 & 0 & 5 \\ 1 & 0 & 0 & 4 & 0 & 0 & 6 \\ 3 & 0 & 0 & 0 & 4 & 0 & 3 \\ 0 & 4 & 0 & 0 & 0 & 7 & 2 \\ 0 & 0 & 4 & 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 7 & 1 & 0 & 2 \\ 5 & 6 & 3 & 2 & 5 & 2 & 0 \end{bmatrix}$

D. $\begin{bmatrix} \infty & 1 & 3 & \infty & \infty & \infty & 5 \\ \infty & \infty & \infty & 4 & \infty & \infty & 6 \\ \infty & \infty & \infty & \infty & 4 & \infty & 3 \\ \infty & \infty & \infty & \infty & \infty & 7 & 2 \\ \infty & \infty & \infty & \infty & \infty & 1 & 5 \\ \infty & \infty & \infty & \infty & \infty & \infty & 2 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty \end{bmatrix}$

| 参考答案

答案	B

2-18 给出如下图所示的具有 7 个结点的网 G，采用Prim算法，从4号结点开始，给出该网的最小生成树。下列哪个选项给出了正确的树结点收集顺序? 2分



A. 4501362

B. 4526301

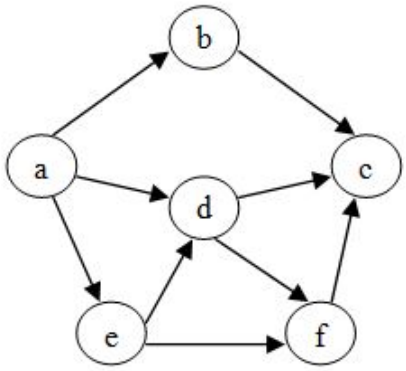
C. 4561023

D. 4563201

| 参考答案

答案	D

2-19 给定有向图如下。下列哪个选项**不是**对应的拓扑序列? 2分

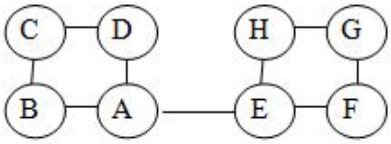


A. abedfc
B. aedbfc
C. aedfbc
D. abdfce

| 参考答案

答案 D

2-20 对下图从顶点C出发进行深度优先搜索，哪个是错误的搜索序列？ 2分

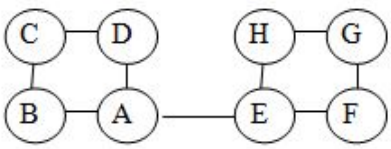


A. CBADEFGH
B. CDABEHFG
C. CDAEHGFB
D. CBAEFGHD

| 参考答案

答案 B

2-21 对下图从顶点C出发进行广度优先搜索，哪个是正确的搜索序列？ 2分

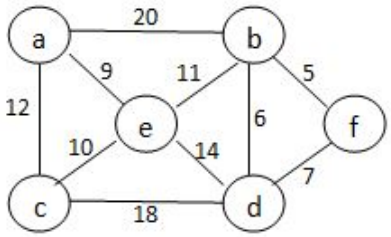


A. CBDAEFGH
B. CDABEHFG
C. CBAEHGFD
D. CBDAEHFG

| 参考答案

答案 D

2-22 已知无向图 G 如下所示，使用克鲁斯卡尔（Kruskal）算法求图 G 的最小生成树，加入到最小生成树中的边依次是： 2分



A. (b,f), (b,d), (a,e), (c,e), (b,e)

B. (b,f), (b,d), (b,e), (a,e), (c,e)

C. (a,e), (b,e), (c,e), (b,d), (b,f)

D. (a,e), (c,e), (b,e), (b,f), (b,d)

| 参考答案

答案　　A

2-23 图的深度优先遍历类似于二叉树的: 2分

A. 先序遍历

B. 中序遍历

C. 后序遍历

D. 层次遍历

| 参考答案

答案　　A

2-24 图的广度优先遍历类似于二叉树的: 2分

A. 先序遍历

B. 中序遍历

C. 层次遍历

D. 后序遍历

| 参考答案

答案　　C

2-25 具有5个顶点的有向完全图有多少条弧? 2分

A. 10

B. 16

C. 20

D. 25

| 参考答案

答案　　C

2-26 下列说法不正确的是: 2分

A. 图的遍历是从给定的源点出发每一个顶点仅被访问一次

B. 遍历的基本算法有两种：深度遍历和广度遍历

C. 图的深度遍历是一个递归过程

D. 图的深度遍历不适用于有向图

| 参考答案

答案	D

2-27 给定有权无向图的邻接矩阵如下，其最小生成树的总权重是： 2分

$$\begin{bmatrix} \infty & 3 & 1 & 15 & 1 & 9 \\ 3 & \infty & 3 & 13 & \infty & 8 \\ 1 & 3 & \infty & 1 & 1 & 2 \\ 15 & 13 & 1 & \infty & 18 & \infty \\ 1 & \infty & 1 & 18 & \infty & 2 \\ 9 & 8 & 2 & \infty & 2 & \infty \end{bmatrix}$$

A. 20

B. 22

C. 8

D. 15

| 参考答案

答案	C

2-28 选择一个排序算法时，除算法的时空效率外，下列因素中，还需要考虑的是： 2分

- I、数据的规模
- II、数据的存储方式
- III、算法的稳定性
- IV、数据的初始状态

A. 仅 III

B. 仅 I、II

C. 仅 II、III、IV

D. I、II、III、IV

| 参考答案

答案	D

2-29 对大部分元素已有序的数组进行排序时，直接插入排序比简单选择排序效率更高，其原因是： 2分

- (I). 直接插入排序过程中元素之间的比较次数更少
- (II). 直接插入排序过程中所需要的辅助空间更少
- (III). 直接插入排序过程中元素的移动次数更少

A. 仅 I

B. 仅 III

C. 仅 I、II

D. I、II 和 III

| 参考答案

答案　　A

2-30 对一组包含10个元素的非递减有序序列，采用直接插入排序排成非递增序列，其可能的比较次数和移动次数分别是：　　2分

- A. 100, 100
- B. 100, 54
- C. 54, 63
- D. 45, 44

| 参考答案

答案　　D

2-31 输入10^6个只有一位数字的整数，可以用$O(N)$复杂度将其排序的算法是：　　2分

- A. 快速排序
- B. 桶排序
- C. 插入排序
- D. 希尔排序

| 参考答案

答案　　B

2-32 给定初始待排序列{ 15，9，7，8，20，-1，4 }。如果希尔排序第一趟结束后得到序列为{ 15，-1，4，8，20，9，7 }，则该趟增量为：　　2分

- A. 1
- B. 2
- C. 3
- D. 4

| 参考答案

答案　　D

2-33 下列排序算法中，占用辅助空间最多的是：( )　　2分

- A. 归并排序
- B. 快速排序
- C. 希尔排序
- D. 堆排序

| 参考答案

答案　　A

2-34 下列排序算法中，不稳定的是：　　2分

I、 希尔排序

II、 归并排序

III、 快速排序
IV、 堆排序
V、 基数排序

A. 仅 I、II　　B. 仅 II、V　　C. 仅 I、III、IV　　D. 仅 III、IV、V

| 参考答案

答案	C

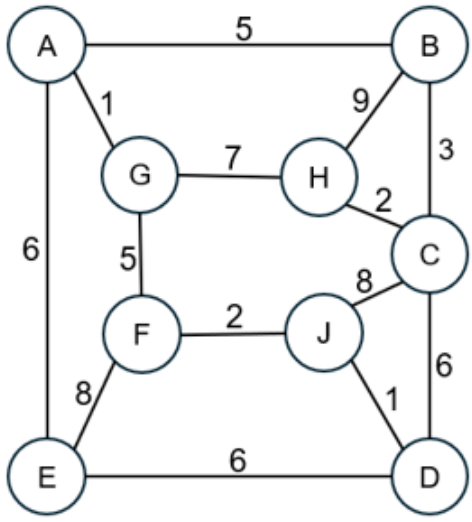
2-35 使用快速排序算法对数据进行升序排序，若经过一次划分后得到的数据序列是 68, 11, 70, 23, 80, 77, 48, 81, 93, 88，则该次划分的枢轴是：　　2分

A. 11　　B. 70　　C. 80　　D. 81

| 参考答案

答案	D

2-36 图G如下所示，　　2分



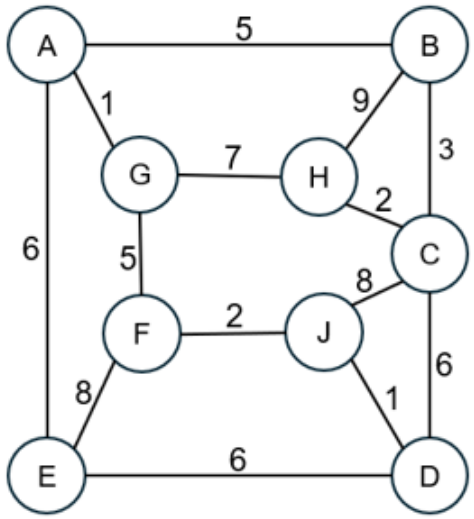
使用Prim（普利姆）算法从**顶点C**开始寻找 **最小生成树** ，选择的第3条边是：

A. (C, D)　　B. (A, G)　　C. (A, B)
D. (D, J)　　E. (F, J)　　F. (C, H)
G. (B, C)　　H. (C, J)　　I. (D, E)

| 参考答案

答案	C

2-37 图G如下所示， 2分



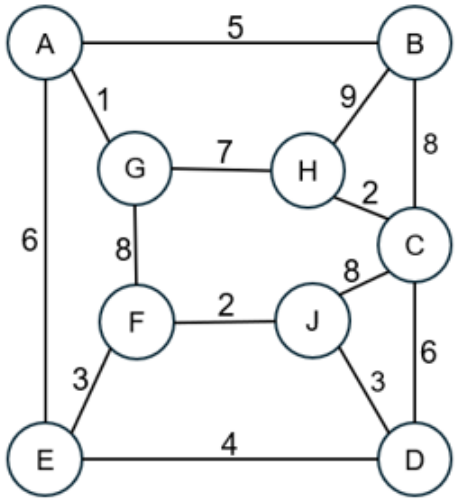
使用Prim（普利姆）算法从**顶点B**开始寻找 **最小生成树** ， 选择的第3条边是：

A. (B, C) B. (A, B) C. (C, H)
D. (C, D) E. (A, G) F. (F, J)
G. (D, J) H. (B, H) I. (G, H)

| 参考答案

答案 B

2-38 图G如下所示， 2分



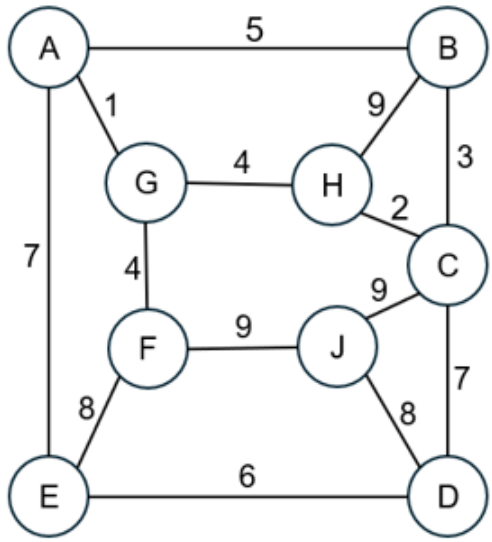
使用Kruskal（克鲁斯卡尔）算法寻找 **最小生成树** ， 选择的第6条边是：

A. (D, E)
B. (A, B)
C. (A, E)或(C, D)
D. (D, J)或(E, F)

| 参考答案

答案 B

2-39 图G如下所示， 2分



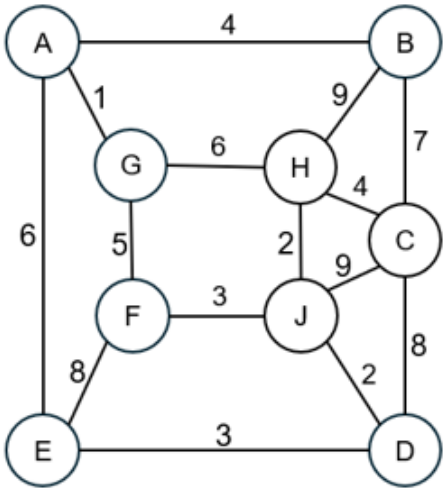
使用Kruskal（克鲁斯卡尔）算法寻找 **最小生成树** ， 选择的第6条边是：

- A. (D, E)
- B. (A, B)
- C. (A, E)或(C, D)
- D. (D, J)或(E, F)

| 参考答案

答案	A

2-40 图G如下所示， 2分



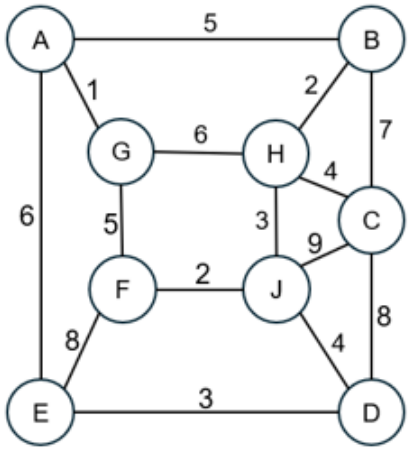
使用**迪杰斯特拉（Dijkstra）算法**求 **顶点C** 到其他每个顶点的最短路径（显然，顶点C到自己的最短路径长度为0），则其他顶点中，第3个确定最短路径的是：

- A. 顶点B
- B. 顶点D
- C. 顶点F
- D. 顶点J

| 参考答案

答案	A

2-41 图G如下所示， 2分



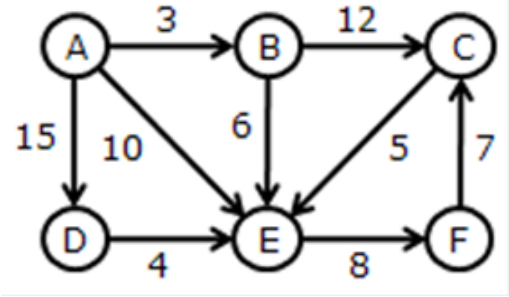
使用**迪杰斯特拉（Dijkstra）算法**求 **顶点C** 到其他每个顶点的最短路径（显然，顶点C到自己的最短路径长度为0），则其他顶点中，第3个确定最短路径的是：

- A. 顶点B
- B. 顶点D
- C. 顶点F
- D. 顶点J

| 参考答案

答案	D

2-42 用Dijkstra算法求下图顶点A到其余各顶点的最短路径时，将按照什么的次序，依次求出A到它们的最短路径。 2分



- A. BEDFC
- B. BCEDF
- C. EDFCB
- D. BEDCF

| 参考答案

答案	D

函数题 40 分

6-1 在图的邻接表存储结构下（基于顶点列表和单链表实现），本题要求图类里实现2个方法函数 def addVertex(self, vex_val): def addEdge(self, f, t, cost=0):

函数接口定义：

```
在这里描述函数接口。例如：
 def addVertex(self, vex_val):
```

```

def addEdge(self, f, t, cost=0):
```

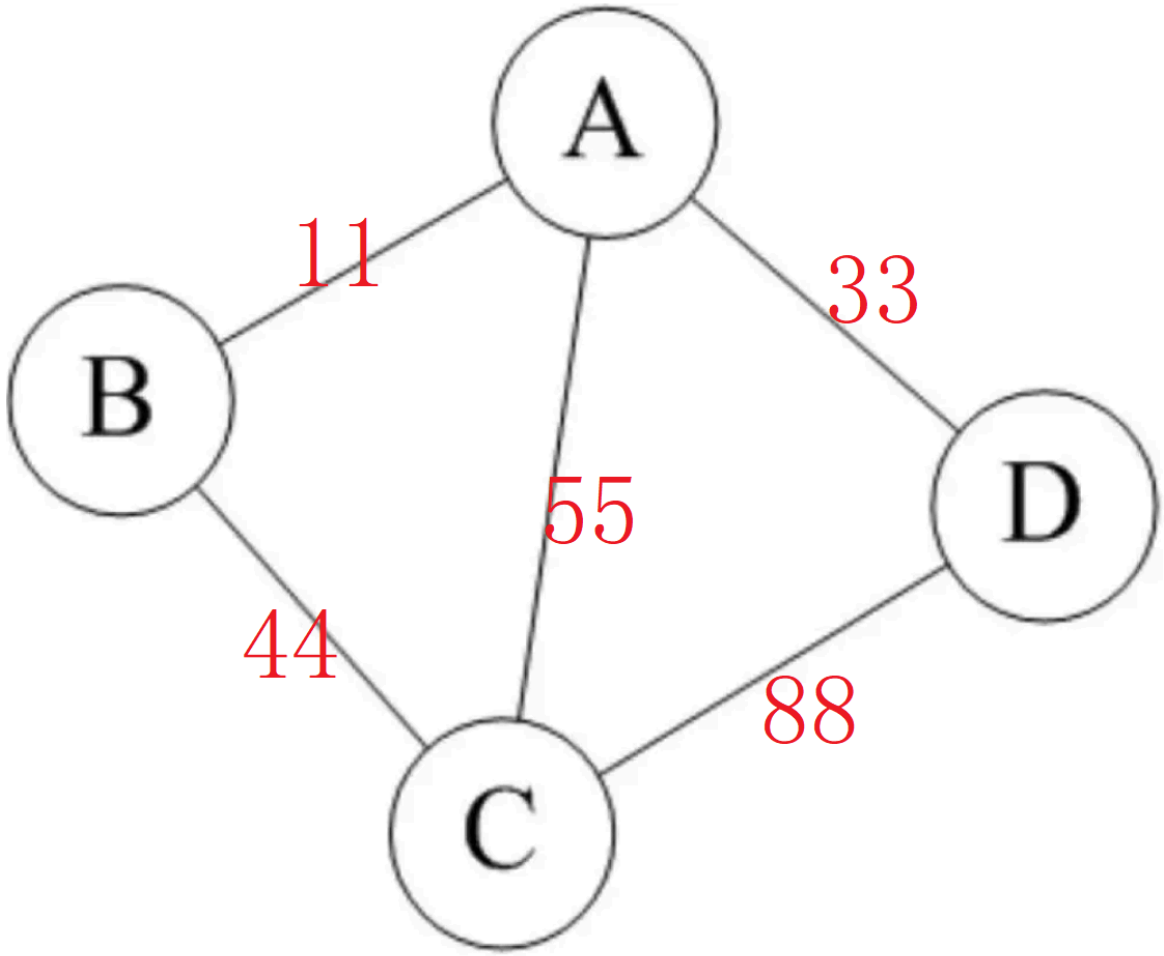
在这里解释接口参数。例如：其中 f和t分别是构成边的顶点在列表中的序号。

裁判测试程序样例：

```
在这里给出函数被调用进行测试的例子。例如：
class arcnode:
    def __init__(self,adjvex,weight,link=None):
        self.adjvex = adjvex
        self.weight = weight
        self.link=link

class vexnode:
    def __init__(self,data,first_arc=None):
        self.data = data
        self.first_arc = first_arc

class Graph:
    def __init__(self):
        self.vex_list=[]
        self.vex_num=0
        self.edge_num=0
    # 请在这里填写答案


   # 请在这里填写答案


    def print_graph(self):
        for i in range(self.vex_num):
            print(self.vex_list[i].data,end="->")
            cur = self.vex_list[i].first_arc
            while cur:
                print("adj:{},weight:{}".format(cur.adjvex,cur.weight),end="->")
                cur = cur.link
            print('None')

if __name__ =="__main__":
    g = Graph()
    s =input()
    for vertex in s:
        g.addVertex(vertex)

    g.addEdge(0,1,11)
    g.addEdge(0,2,55)
    g.addEdge(2,3,88)
    g.addEdge(0,3,33)
    g.addEdge(1,2,44)
    g.print_graph()
```

输入样例：

在这里给出一组输入。例如：




```
ABCD
```

输出样例：

在这里给出相应的输出。例如：

```
A->adj:3,weight:33->adj:2,weight:55->adj:1,weight:11->None
B->adj:2,weight:44->adj:0,weight:11->None
C->adj:1,weight:44->adj:3,weight:88->adj:0,weight:55->None
D->adj:0,weight:33->adj:2,weight:88->None
```

| 参考答案

答案　　编译器: PYTHON3

6-2 本题要求实现一个函数，试实现邻接矩阵存储图的深度优先遍历。　　10分

函数接口定义：

```
#顶点v(编号)出发对图G进行深度优先遍历
def dfs(G,v)
```

其中图G的定义如下：

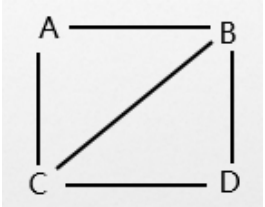
```
class adjMatrixGraph:
    # 构造方法，n个顶点m条边
    def __init__(self,n,m):
        self.verNum = n     #顶点数
        self.edgeNum = m    #边数
        self.vertex = [0] * n       #顶点列表
        self.edge = [[0 for i in range(self.verNum)] \
                        for j in range(self.verNum)]   #邻接矩阵二维列表
        self.vis = [False] * n     #顶点的访问列表，默认没访问过
    def addVertex(self,ls): #添加顶点列表
        self.vertex = ls
    def addEdge(self,fr,to):#添加边(fr,to)
        ifr = self.vertex.index(fr)     #起点下标
        ito = self.vertex.index(to)     #终点下标
        self.edge[ifr][ito] = self.edge[ito][ifr] = 1 #邻接矩阵

#邻接矩阵建图
def createGraph():
    n,m = map(int,input().split())  #输入n个顶点和m条边
    g = adjMatrixGraph(n,m)         #创建无向图G
    g.addVertex(list(input().split()))  #输入顶点列表
    for i in range(m):              #输入m条边
        fr,to = input().split()
        g.addEdge(fr,to)
    return g                        #返回无向图g
```

裁判测试程序样例:

```
g = createGraph()   #创建无向图g
v = int(input())        #输入出发顶点的编号
print("DFS from " + g.vertex[v] + " :",end = "")
dfs(g,v)                    #顶点v(编号)出发对图G进行深度优先遍历

#你的代码将被嵌在这里
```

输入样例:

例如无向图



第一行给出图的顶点数n和边数m。第二行给出n个顶点字符串（中间用1个空格分隔），表示n个顶点的数据元素的值。后面是m行，给出每一条边的两个顶点（中间用1个空格分隔）。最后一行输入出发顶点的编号。

```
4 5
A B C D
A B
A C
B C
B D
C D
0
```

输出样例（对于图中给出的树）:

输出顶点v(编号)出发进行深度优先遍历的顶点（每个顶点前都有1个空格），格式参照样例。

```
DFS from A : A B C D
```

| 参考答案

答案　　编译器: PYTHON3

6-3 本题要求实现一个函数，输出无向图每个顶点的数据元素的值，以及每个顶点度的值。　　10分

函数接口定义:

```
#输出无向图G每个顶点的度数
def printDegree(G)
```

其中图G的定义如下:

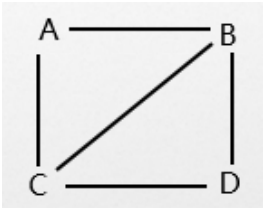
```
class adjMatrixGraph:
    # 构造方法，n个顶点m条边
    def __init__(self,n,m):
        self.verNum = n     #顶点数
        self.edgeNum = m    #边数
        self.vertex = [0] * n       #顶点列表
        self.edge = [[0 for i in range(self.verNum)] \
                        for j in range(self.verNum)]   #邻接矩阵二维列表
        self.vis = [False] * n     #顶点的访问列表，默认没访问过
    def addVertex(self,ls): #添加顶点列表
        self.vertex = ls
    def addEdge(self,fr,to):#添加边(fr,to)
        ifr = self.vertex.index(fr)     #起点下标
        ito = self.vertex.index(to)     #终点下标
        self.edge[ifr][ito] = self.edge[ito][ifr] = 1 #邻接矩阵

#邻接矩阵建图
def createGraph():
    n,m = map(int,input().split())  #输入n个顶点和m条边
    g = adjMatrixGraph(n,m)         #创建无向图G
    g.addVertex(list(input().split()))  #输入顶点列表
    for i in range(m):              #输入m条边
        fr,to = input().split()
        g.addEdge(fr,to)
    return g                        #返回无向图g
```

裁判测试程序样例:

```
g = createGraph()   #创建无向图g
printDegree(g)      #输出g的度数

#你的代码将被嵌在这里
```

输入样例:

例如无向图



第一行给出图的顶点数n和边数m。第二行给出n个顶点字符串（中间用1个空格分隔），表示n个顶点的数据元素的值。后面是m行，给出每一条边的两个顶点（中间用1个空格分隔）。

```
4 5
A B C D
A B
A C
B C
B D
C D
```

输出样例（对于图中给出的树）：

输出n个顶点以及各顶点的度数，格式参照样例。

```
A:2
B:3
C:3
D:2
```

| 参考答案

答案　　编译器: PYTHON3

6-4 本题要求实现一个函数，试实现邻接矩阵存储图的广度优先遍历。　　10分

函数接口定义:

```
#顶点v(编号)出发对图G进行广度优先遍历
def bfs(G,v)
```

其中图G的定义如下:

```
class adjMatrixGraph:
    # 构造方法，n个顶点m条边
    def __init__(self,n,m):
        self.verNum = n     #顶点数
        self.edgeNum = m    #边数
        self.vertex = [0] * n       #顶点列表
        self.edge = [[0 for i in range(self.verNum)] \
                        for j in range(self.verNum)]   #邻接矩阵二维列表
        self.vis = [False] * n     #顶点的访问列表，默认没访问过
    def addVertex(self,ls): #添加顶点列表
        self.vertex = ls
    def addEdge(self,fr,to):#添加边(fr,to)
        ifr = self.vertex.index(fr)     #起点下标
        ito = self.vertex.index(to)     #终点下标
        self.edge[ifr][ito] = self.edge[ito][ifr] = 1 #邻接矩阵

#邻接矩阵建图
def createGraph():
    n,m = map(int,input().split())  #输入n个顶点和m条边
    g = adjMatrixGraph(n,m)         #创建无向图G
```

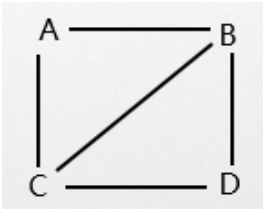
```
    g.addVertex(list(input().split()))  #输入顶点列表
    for i in range(m):              #输入m条边
        fr,to = input().split()
        g.addEdge(fr,to)
    return g                        #返回无向图g

#定义抽象类型队列Queue，FIFO（First In,First Out)
class Queue:
    #1.构造方法,定义一个空的列表
    def __init__(self):
        self.items = []
    #2.入队,队尾（列表尾部）入队
    def push(self,item):
        self.items.append(item)
    #3.出队，队首（列表头部）出队
    def pop(self):
        return self.items.pop(0)
    #4.判断队列是否为空
    def isEmpty(self):
        return self.items == []
    #5.取队首
    def getFront(self):
        return self.items[0]
    #6.求队列大小
    def getSize(self):
        return len(self.items)

#你的代码将被嵌在这里
```

裁判测试程序样例：

```
g = createGraph()   #创建无向图g
v = int(input())    #输入出发顶点的编号
print("BFS from " + g.vertex[v] + " :",end = "")
bfs(g,v)            #顶点v(编号)出发对图G进行广度优先遍历
```

输入样例：

例如无向图



第一行给出图的顶点数n和边数m。第二行给出n个顶点字符串（中间用1个空格分隔），表示n个顶点的数据元素的值。后面是m行，给出每一条边的两个顶点（中间用1个空格分隔）。最后一行输入出发顶点的编号。

```
4 5
A B C D
A B
A C
B C
B D
C D
0
```

输出样例：

对于给定的无向图，输出顶点v(编号)出发进行广度优先遍历的顶点（每个顶点前都有1个空格），格式参照样例。

```
BFS from A : A B C D
```

| 参考答案

答案　　编译器: PYTHON3

编程题　　60 分

7-1 采用邻接矩阵表示法创建无向图G ，依次输出各顶点的度。　　10分

输入格式:

输入第一行中给出2个整数i(0<i≤10)，j(j≥0)，分别为图G的顶点数和边数。
输入第二行为顶点的信息，每个顶点只能用一个字符表示。
依次输入j行，每行输入一条边依附的顶点。

输出格式:

依次输出各顶点的度，行末没有最后的空格。

输入样例:

```
5 7
ABCDE
AB
AD
BC
BE
CD
CE
DE
```

输出样例:

```
2 3 3 3 3
```

| 参考答案

答案　　编译器: NO_COMPILER

7-2 采用邻接表创建无向图G ，依次输出各顶点的度。　　10分

输入格式:

输入第一行中给出2个整数i(0<i≤10)，j(j≥0)，分别为图G的顶点数和边数。
输入第二行为顶点的信息，每个顶点只能用一个字符表示。
依次输入j行，每行输入一条边依附的顶点。

输出格式:

依次输出各顶点的度，行末没有最后的空格。

输入样例:

```
5 7
ABCDE
AB
AD
BC
BE
CD
CE
DE
```

输出样例:

```
2 3 3 3 3
```

| 参考答案

答案	编译器: NO_COMPILER

7-3 本题要求输出两个顶点之间是否存在路径 10分

输入格式:

输入包括两部分，第一部分是邻接矩阵表示方法中对应1的两个顶点，用0 0 表示结束
第二部分是两个顶点，例如 Vi和Vj

输出格式:

如果Vi和Vj存在路径，输出1；否则输出0

输入样例:

```
0 1
1 0
0 4
4 0
1 4
4 1
1 2
2 1
1 3
3 1
2 3
3 2
4 5
5 4
4 6
6 4
0 0
0 5
```

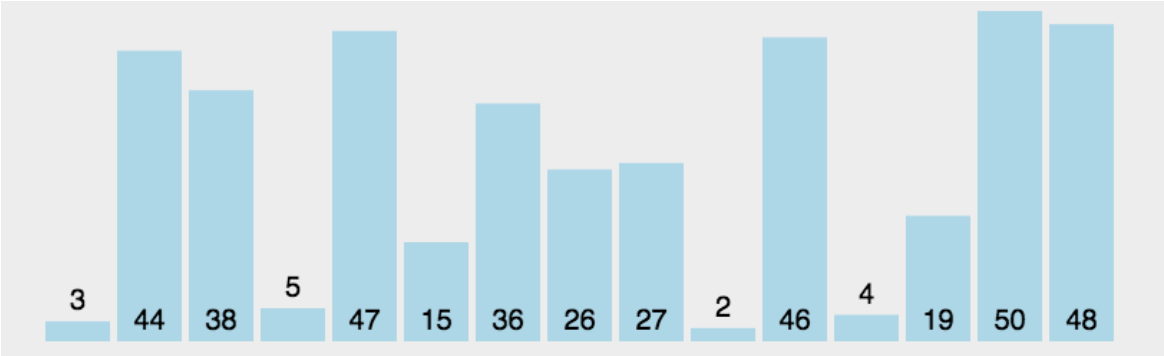
输出样例:

```
1
```

| 参考答案

答案　　编译器: NO_COMPILER

7-4 选择排序，从头至尾扫描序列,找出最小的一个元素，和第一个元素交换,接着从剩下的元素中继续这种选择和交换方式，最终得到一个有序序列。　　10分



输入格式:

输入在第1行中给出N（1<N≤100），在第2行中给出N个待排序的整数，数字间以空格分隔，并保证数字没有重复的出现。

输出格式:

给出选择排序每一遍后的中间结果数列，数字间以空格分隔，但末尾不得有多余空格。注意：当排序完成时应立即停止。

输入样例1:

```
7
4 5 7 6 3 2 1
```

输出样例1:

```
1 5 7 6 3 2 4
1 2 7 6 3 5 4
1 2 3 6 7 5 4
1 2 3 4 7 5 6
1 2 3 4 5 7 6
1 2 3 4 5 6 7
```

输入样例2:

```
5
1 2 3 5 4
```

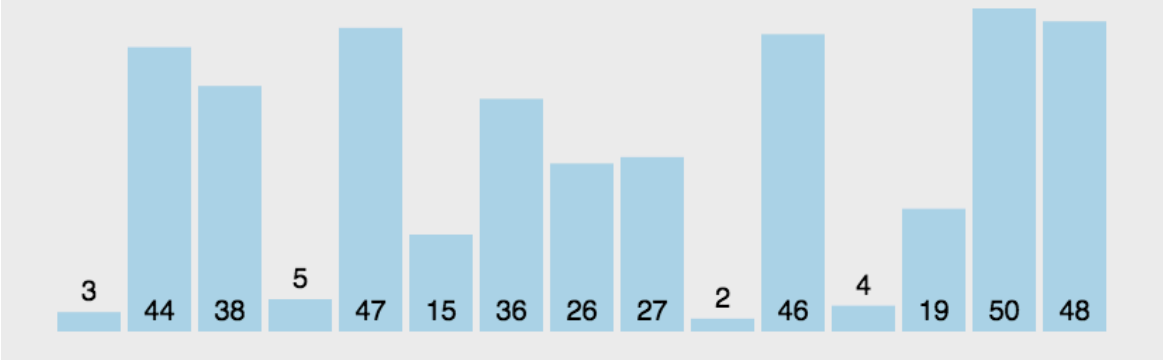
输出样例2:

```
1 2 3 4 5
```

| 参考答案

答案　　编译器: NO_COMPILER

7-5 冒泡排序，将一个列表中的两个元素进行比较，并将最小的元素交换到顶部。两个元素中较小的会冒到顶部，而较大的会沉到底部，该过程将被重复执行，直到所有元素都被排序。 10分



输入格式:

输入在第1行中给出N（1<N≤100），在第2行中给出N个待排序的整数，数字间以空格分隔，并保证数字没有重复的出现。

输出格式:

给出冒泡排序每一遍后的中间结果数列，数字间以空格分隔，但末尾不得有多余空格。注意：当排序完成时应立即停止。

输入样例1:

```
7
4 5 7 6 3 2 1
```

输出样例1:

```
4 5 6 3 2 1 7
4 5 3 2 1 6 7
4 3 2 1 5 6 7
3 2 1 4 5 6 7
2 1 3 4 5 6 7
1 2 3 4 5 6 7
```

输入样例2:

```
6
1 2 3 6 5 4
```

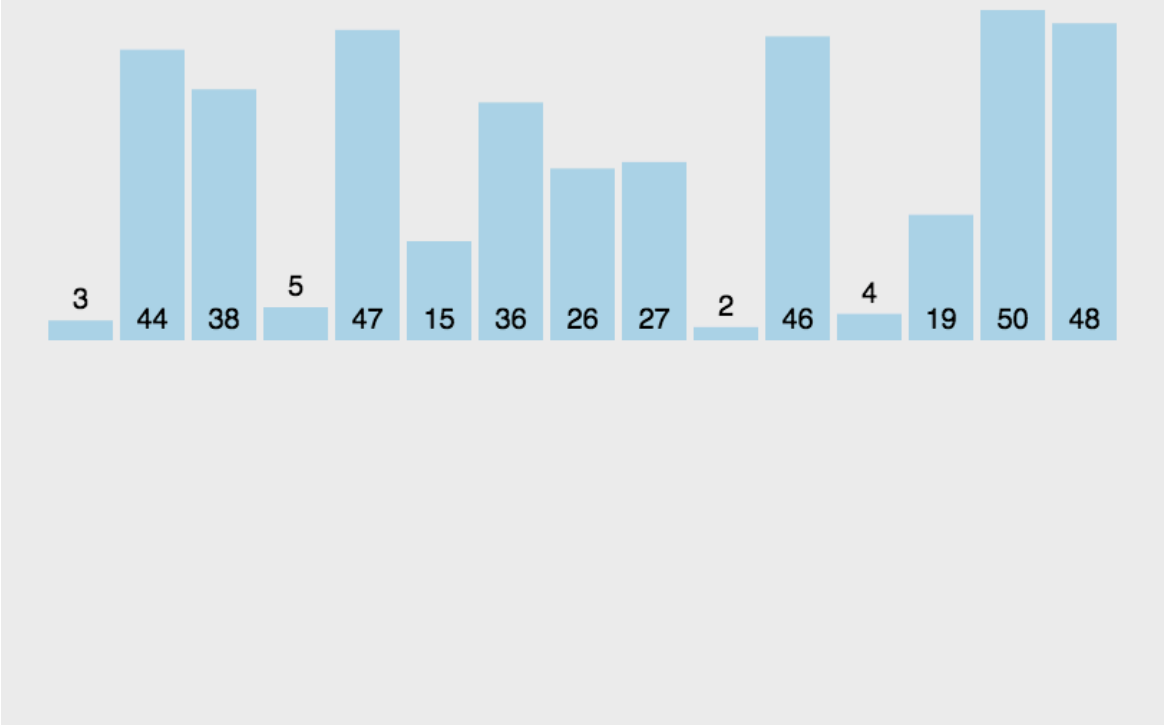
输出样例2:

```
1 2 3 5 4 6
1 2 3 4 5 6
```

| 参考答案

答案 编译器: NO_COMPILER

7-6 插入排序是一种简单直观的排序算法。它的工作原理是通过构建有序序列，对于未排序数据，在已排序序列中从后向前扫描，找到相应位置并插入。 10分



输入格式:

输入在第1行中给出N（1<N≤100），在第2行中给出N个待排序的整数，数字间以空格分隔，并保证数字没有重复的出现。

输出格式:

给出插入排序每一遍后的中间结果数列，数字间以空格分隔，但末尾不得有多余空格。

输入样例:

```
7
4 5 7 6 3 2 1
```

输出样例:

```
4 5 7 6 3 2 1
4 5 7 6 3 2 1
4 5 6 7 3 2 1
3 4 5 6 7 2 1
2 3 4 5 6 7 1
1 2 3 4 5 6 7
```

| 参考答案

答案　　编译器: NO_COMPILER