算法分析模拟试卷参考答案及评分标准

1. **判断正误题，正确的后面标 T，错误的后面标 F**（本题 10 分，每空 1 分）

1-5　T T F T F　6-10 F T T F F

1. T n is O ($(\log n)^{\log n}$)

2. $n = 2^{\log n}$ and $(\log n)^{\log n} = 2^{\log\log n \cdot \log n}$. 显然，$\log n$ ▮ $\log\log n \cdot \log n$ n .

3. T

4. F，基于比较的排序算法时间复杂度为下界为 O(nlogn)

5. 假设一个具有不同边权的连通无向图，贪婪地删除不会影响图的连通性的权最重的边，直到不能再删除任何边，这样得到的结果是原始图的最小生成树。　T。

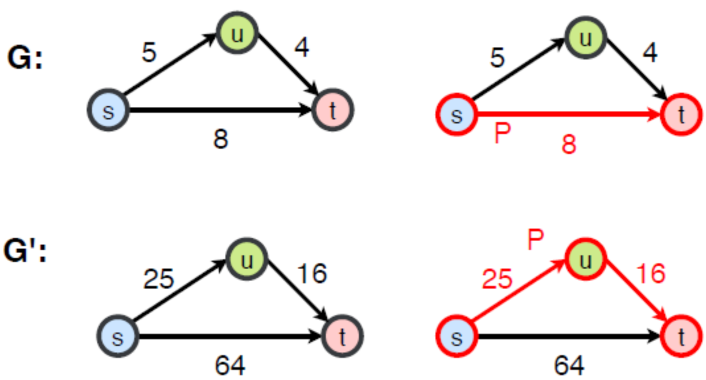
6. 设 G=（V，E）是一个加权图，设 T 是 G 的最小生成树。T 中任何一对顶点 u 和 V 之间的路径必须是 G 中的最短路径。

F。反例：V={a，b，c}和 E={（a，b），（b，c），（c，a）}，其中 w（a，b）=3，w（b，c）=3 和 w（c，a）=4。显然，T={（a，b），（b，c）}。但是，c 和 a 之间的最短路径是权重 4 的边缘（c，a），而不是来自总权重为 3+3=6 的 MST 的路径。

7. 给定有向加权图 G=（V，E），设 P 是两个给定 s 和 t 节点之间的最小权重（最短距离）s–t 路径。

假设我们将每个边权重 w_e 替换为 w_e 的平方 w_e^2，从而创建具有相同顶点但不同的边权重的图 G'。那么，对于新的图 G'，P 仍然是 s 和 t 之间的最小权重（最短）路径。

F。作为反例，请考虑以下图形 G 和 G'：



8. 给定n个整数a1、，an，可以在O（n）时间内找到第3个最小数。T

9. 给定一个由 n 个整数组成的数组，每个整数都属于{-1，0， 1}，在最坏的情况下，我们可以在 O（n）时间内对数组进行排序。 T。使用计数排序，例如，在所有数字加 1 之后。

10. 3SAT 不能在多项式时间内求解，即使 P=NP。

F。如果 P=NP，那么 P 中的所有问题也是 NP 难的，并且问题有多项式时间算法。

11. 伪多项式时间算法 F

二、简答题（本题 12 分，每小题 6 分）

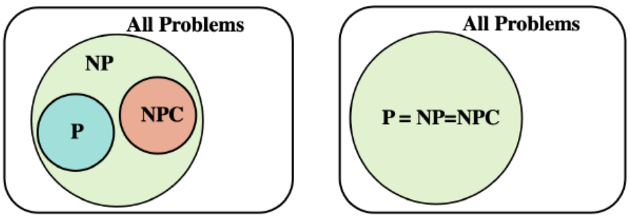
1. 有这样一类特殊 0-1 背包问题：可选物品重量越轻的物品价值越高。n=6，c=20，P=(4，8， 15，1，6，3)，W=(5，3，2，10，4，8)。其中 n 为物品个数，c 为背包载重量，P 表示物品的价值，W 表示物品的重量。请问对于此特殊的 0-1 背包问题，为使放进背包的物品总价值最大，应如何选择放进去的物品？此例能获得的最大总价值多少？

解：因为该 0—1 背包问题比较特殊，恰好重量越轻的物品价值越高，所以优先取重量轻的物品放进背包。最终可以把重量分别为 2，3，4，5 的四个物品放进背包，得到的价值和为 15 + 8 + 6 + 4 = 33，为最大值。

其他解答：按正确程度酌情给分。

2. 简要描述类 P、NP 和 NP 完全复杂度之间关系的两种可能性，并使用维恩图（根据集合包含）显示这两种可能性。

$$(1)\ P = NP = NPC,\ (2)\ P \subseteq NP \text{ but } NP \not\subseteq P, \text{ so } P \neq NP$$



P≠NP　　　　P=NP

三、计算与算法应用题（本题 48 分，每小题 12 分）

1. 查找已排序数组的中位数很容易：返回中间元素。 但是，如果给定两个大小分别为 **m** 和 **n** 的排序数组 **A** 和 **B**，并且你想找到 **A** 和 **B** 中所有数字的中位数有下述多种算法，你可以假设 **A** 和 **B** 是不相交的。

(a) 合并两个排序数组并使用线性时间选择找到中值。 [4 分]

合并需要 O(m + n) 时间， Θ(m + n) 时间内运行

(b) 如果 m＝n，为 A 选择中位数 m1，为 B 选择中位数 m2。如果 m1＝m2 ，则返回 m1 。如果 m1＞m2 ，删除 A 的后半部分和 B 的前半部分。然后我们得到两个大小为 n/2 的子数组。 重复直到两个数组都小于一个常数。 m1 ＜ m2 是对称的。[4 分]

Θ(lg n) 时间内运行的算法。

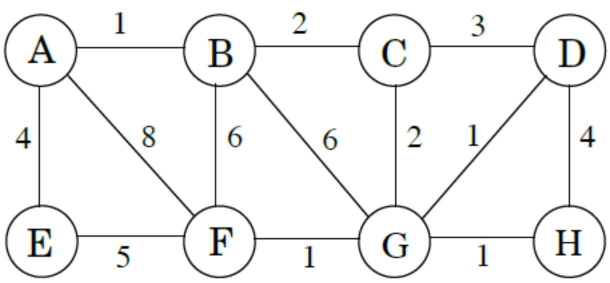
(c) 对于任何 m 和 n。假设|A| = m > n = |B|。 则可以安全地删除元素 A[0 : (m−n)

/2] 和 A[(m+n)/2 : m - 1]，因为这些元素都不能是 A + B 的中值。经过这个过程，

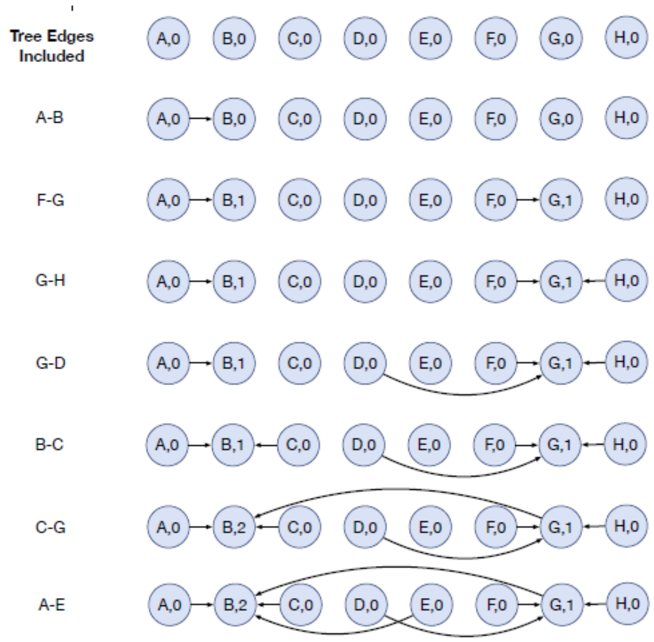
我们得到两个大小约为 n 的数组 . 然后我们使用 (b) 部分算法。 [4 分]

复杂度为 $\Theta(\lg(\min(m, n)))$ 时间内运行的算法

2. 利用 **Kruskal** 算法计算下图的最小生成树。要求显示每个步骤中连通分量（并集和）的状态。



解答：该算法将首先对边进行排序，得到以下顺序：**A-B**、**F-G**、**G-H**、**G-D**、**B-C**、**C-G**、**C-D**、**A-E**、**D-H**、**E-F**、**B-F**、**B-G**、**A-F**。只有在连接两个新连接的分量时，每条边才会添加到树中。下图显示了添加的树边以及每次添加边后并集数据结构的状态。



根据过程和结果，按正确情况酌情给分。

3. 利用动态规划方法求 **A = 'bacdca'**、**B = 'adbcda'**编辑距离的子问题存储表。

解答：

A B 两字符串最小编辑距离为 4.

首先，我们创建一个大小为(len(A) + 1) × (len(B) + 1)的二维数组，初始时所有元素都为 0。其中，len(A)表示字符串 A 的长度，len(B)表示字符串 B 的长度。

下面是根据输入的字符串 A 和字符串 B 的编辑距离子问题存储表的示意图：

	Ø	a	d	b	c	d	a
Ø	0	1	2	3	4	5	6
b	1	1	2	2	3	4	5
a	2	1	2	3	3	4	5
c	3	2	3	3	2	3	4
d	4	3	3	4	3	3	4
c	5	4	4	4	3	4	4
a	6	5	5	5	4	4	4

在这个表中，每个格子(i，j)表示将字符串 A 的前 i 个字符转换为字符串 B 的前 j 个字符所需的最小编辑距离。表的第一行和第一列分别对应空字符串与字符串 A、B 之间的编辑距离。最右下角的格子的值即为字符串 A 和字符串 B 之间的编辑距离。

根据表的正确程度，酌情给分。

4.

最优旅行的顺序为 1 3 2 4 1

旅行售货员问题的解空间可以组织成一棵树，从树的根结点到任一叶结点的路径定义了图的一条周游路线。旅行售货员问题要在图 G 中找出费用最小的周游路线。路线是一个带权图。图中各边的费用（权）为正数。图的一条周游路线是包括 V 中的每个顶点在内的一条回路。周游路线的费用是这条路线上所有边的费用之和。

算法开始时创建一个最小堆，用于表示活结点优先队列。堆中每个结点的子树费用的下界 lcost 值是优先队列的优先级。接着算法计算出图中每个顶点的最小费用出边并用 minout 记录。如果所给的有向图中某个顶点没有出边，则该图不可能有回路，算法即告结束。如果每个顶点都有出边，则根据计算出的 minout 作算法初始化。

算法的 while 循环体完成对排列树内部结点的扩展。对于当前扩展结点，算法分 2 种情况进行处理：

1、首先考虑 s=n-2 的情形，此时当前扩展结点是排列树中某个叶结点的父结点。如果该叶结点相应一条可行回路且费用小于当前最小费用，则将该叶结点插入到优先队列中，否则舍去该叶结点。

2、当 s<n-2 时，算法依次产生当前扩展结点的所有儿子结点。由于当前扩展结点所相应的路径是 x[0:s]，其可行儿子结点是从剩余顶点 x[s+1:n-1]中选取的顶点 x[i]，且(x[s]，x[i])是所给有向图 G 中的一条边。对于当前扩展结点的每一个可行儿子结点，计算出其前缀

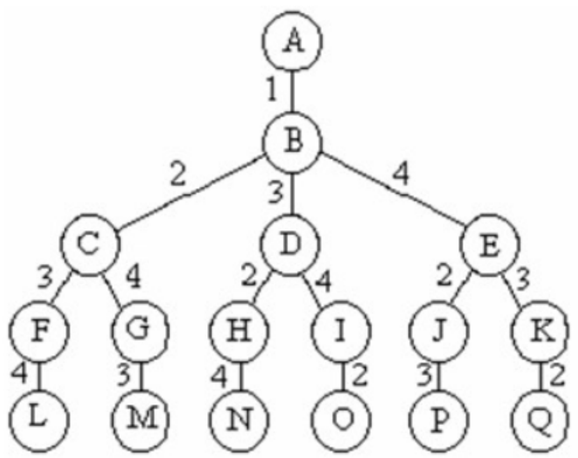
(x[0:s]，x[i])的费用 cc 和相应的下界 lcost。当 lcost<bestc 时，将这个可行儿子结点插入到活结点优先队列中。

算法中 while 循环的终止条件是排列树的一个叶结点成为当前扩展结点。当 s=n-1 时，已找到的回路前缀是 x[0:n-1]，它已包含图 G 的所有 n 个顶点。因此，当 s=n-1 时，相应的扩展结点表示一个叶结点。此时该叶结点所相应的回路的费用等于 cc 和 lcost 的值。剩余的活结点的 lcost 值不小于已找到的回路的费用。它们都不可能导致费用更小的回路。因此已找到的叶结点所相应的回路是一个最小费用旅行售货员回路，算法可以结束。

算法结束时返回找到的最小费用，相应的最优解由数组 v 给出。

算法执行过程最小堆中元素变化过程如下：

{ }—{B}—{C，D，E}—{C，D，J，K}—{C，J，K，H，I}—{C，J，K，I，N}—{C，K，I，N，P}—{C，I，N，P，Q}—{C，N，P，Q，O}—{C，P，Q，O}—{C，Q，O}—{Q，O，F，G}—{Q，O，G，L}—{Q，O，L，M}—{O，L，M}—{O，M}—{M}—{ }



也有其他画法。根据正确程度，酌情给分。

四、算法设计题（本题 30 分，每小题 15 分，第 1，2 题选做 1 题，第 3 题必做）

```
for (i = 0; i < n; ++i)
        {
            scanf("%d", arr + i);//键入人的重量
        }
        bubble(arr,n);
        printf("\n");
        for (i = 0, j = n - 1;i<=j ;)
        {
            if (arr[i] + arr[j] < max)
            {
                i++;
                j--;
                ++count;
            }
            else
            {
                j--;
                ++count;
            }
        }
```

.应用 SPFA 算法判断负环，SPFA 的复杂度是 O(|V||E|)。其他算法，根据正确程度，酌情给分。

```
bool SPFA(int acioi)
{
    queue<int>q;
    for(register int i = 1;i <= n;++ i)
        d[i] = 99999999;
    d[acioi] = 0;
    q.push(acioi);
    while(!q.empty())
    {
        int x = q.front();
        q.pop();use[x] = false;
        for(register int i = head[x];i != 0;i = a[i].ne)
        {
            int y = a[i].y;
            if(d[y] > d[x] + a[i].z)
            {
                d[y] = d[x] + a[i].z;
                cnt[y] = cnt[x] + 1;
                if(cnt[y] > n)
                    return false;
                if(use[y] == false)
                {
                    use[y] = true;
                    q.push(y);
                }
            }
        }
    }
    return true;
}
```

4.　该问题为多重背包问题，在 01 背包的问题上增加条件:第 i 个物品可以选 n[i]个.

状态转移方程为 f[i][v]=max{f[i-1][v-k*c[i]]+k*w[i]|0<=k<=n[i]}，参考代码如下

```
#include <iostream>
#include <algorithm>
#define N 1002
using namespace std;

int f[N];
int w[N];
int v[N];
int s[N];

int main() {
    int n,W; cin >> n >> W;
    for(int i=1;i<=n;i++) {
        cin >> w[i] >> v[i] >> s[i];
    }
    for(int i=1;i<=n;i++) {
        for(int j=W;j>=w[i];j--) {
            for(int k=0;k<=s[i] && k*w[i] <=j ;k++) {
                f[j] = max(f[j],f[j-k*w[i]] + k*v[i]);
            }
        }
    }
    cout << f[W] <<endl;
    return 0;
}
```

其他算法，按正确与否，酌情给分。