CCF 编程能力等级认证
Grade Examination of Software Programming

Python 八级

2026 年 09 月

1 单选题（每题 2 分，共 30 分）

题号	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
答案	B	C	C	A	B	B	C	D	A	B	A	C	A	D	D

第 1 题 用数字 0、1、2、3、4、5 组成没有重复数字的三位数，且该三位数能被 3 整除，共有（ ）个。

- **A.** 36
- **B.** 40
- **C.** 44
- **D.** 48

第 2 题 6 个人围成一圈就座，座位没有区分，但区分时针方向，且甲、乙两人必须相邻，则共有（ ）种不同坐法。

- **A.** 24
- **B.** 36
- **C.** 48
- **D.** 120

第 3 题 将 6 个完全相同的小球放入 4 个不同的盒子中，每个盒子均可以为空，共有（ ）种不同的放法。

- **A.** 10
- **B.** 60
- **C.** 84
- **D.** 4096

第 4 题 某二叉树的先序遍历序列为 `A B D E C F G`，中序遍历序列为 `D B E A F C G`，则其后序遍历序列为（ ）。

- **A.** `D E B F G C A`
- **B.** `D B E F G C A`
- **C.** `D E B G F C A`
- **D.** `D E B F C G A`

第 5 题 关于快速幂算法，下列说法正确的是（ ）。

- **A.** 快速幂可以处理任意整数负指数的情况
- **B.** 快速幂将指数按二进制拆分，能够把乘法次数从朴素乘幂时的 $O(b)$ 优化为 $O(\log b)$

- [ ] **C.** 快速幂只能在模数为质数时使用
- [ ] **D.** 快速幂的空间复杂度通常为 $O(b)$

第 6 题 杨辉三角中，第 7 行第 3 个数（行、列均从 0 开始计数）是（　　）。

- [ ] **A.** 21
- [ ] **B.** 35
- [ ] **C.** 42
- [ ] **D.** 56

第 7 题 一个长方形的长是宽的 3 倍，周长为 48，则该长方形的面积为（　　）。

- [ ] **A.** 72
- [ ] **B.** 96
- [ ] **C.** 108
- [ ] **D.** 144

第 8 题 若 $x+y=7$，$x-y=1$，则 $x\times y$ 的值为（　　）。

- [ ] **A.** 7
- [ ] **B.** 8
- [ ] **C.** 10
- [ ] **D.** 12

第 9 题 关于最小生成树（MST）算法，下列说法正确的是（　　）。

- [ ] **A.** Prim 算法适用于稠密图，Kruskal 算法适用于稀疏图
- [ ] **B.** Prim 算法和 Kruskal 算法得到的最小生成树边集一定完全相同
- [ ] **C.** Kruskal 算法必须使用邻接矩阵存储图
- [ ] **D.** Prim 算法只能处理有向图

第 10 题 某连通带权无向简单图的边集合为 `{(1, 2, 5), (1, 3, 1), (2, 3, 3), (2, 4, 4), (3, 4, 2), (3, 5, 6), (4, 5, 7)}`，其中，每条边的三元组 `(u, v, w)` 表示结点 `u` 和结点 `v` 之间有一条权值为 `w` 的无向边。使用 Kruskal 算法按边权从小到大扫描，第 3 条被选入最小生成树的边是（　　）。

- [ ] **A.** `(1, 2, 5)`
- [ ] **B.** `(2, 3, 3)`
- [ ] **C.** `(3, 4, 2)`
- [ ] **D.** `(4, 5, 7)`

第 11 题 Floyd 算法经典三重循环 `for k for i for j`，最外层变量 `k` 代表（　　）

- [ ] **A.** 当前允许作为中间顶点的最大编号（即只允许编号不超过 `k` 的顶点作为中间点）
- [ ] **B.** 当前起点
- [ ] **C.** 当前终点
- [ ] **D.** 当前最短路径的长度

第 12 题 下列函数按渐近增长速度从慢到快排列，正确的是（　　）。

- [ ] **A.** $O(n^2)$、$O(n\log n)$、$O(n)$、$O(\log n)$
- [ ] **B.** $O(\log n)$、$O(n)$、$O(n^2)$、$O(n\log n)$
- [ ] **C.** $O(\log n)$、$O(n)$、$O(n\log n)$、$O(n^2)$
- [ ] **D.** $O(n\log n)$、$O(\log n)$、$O(n)$、$O(n^2)$

第 13 题 在使用小根堆（优先队列）优化的 Dijkstra 算法中，堆中每个元素通常存储的是（　　）。

- [ ] **A.** 顶点编号和该顶点当前的最短距离
- [ ] **B.** 边的权值和边的终点
- [ ] **C.** 顶点的入度
- [ ] **D.** 父结点编号和边权

第 14 题 对长度为 `n` 的数组使用差分数组支持 `m` 次区间加操作，最后通过一次前缀和还原每个位置的最终值，整个过程的渐进时间复杂度为（　　）。

- [ ] **A.** $O(mn)$
- [ ] **B.** $O((n+m)\log n)$
- [ ] **C.** $O(n\log m)$
- [ ] **D.** $O(n+m)$

第 15 题 假设在 Python 标准实现下运行下面程序运行打印输出为（　　）

```python
class A:
    def __init__(self):
        print("A",end="")
    def __del__(self):
        print("~A",end="")

class B(A):
    def __init__(self):
        super().__init__()
        print("B",end="")
    def __del__(self):
        print("~B",end="")

def main():
    b = B()

main()
```

- [ ] **A.** `BA~A~B`
- [ ] **B.** `BA~B~A`
- [ ] **C.** `AB~A~B`
- [ ] **D.** `AB~B`

2　判断题（每题 2 分，共 20 分）

题号	1	2	3	4	5	6	7	8	9	10
答案	√	√	×	√	×	√	√	×	√	√

第 1 题 从 4 本不同的书中选出 3 本，分别分给甲、乙、丙 3 人，每人至多 1 本，共有 $P(4,3)=24$ 种不同分法。

第 2 题 对任意正整数 n，二项式 $(a+b)^n$ 的展开式中，按项序从第 0 项起计数，奇数项系数之和等于偶数项系数之和。

第 3 题 若一个连通无向图的最小生成树中存在权值相同的边，则最小生成树一定不唯一。

第 4 题 使用邻接表存储图时，Dijkstra 算法的朴素实现（不使用堆优化）的时间复杂度为 $O(V^2)$，其中 V 为结点数。

第 5 题 堆排序是一种稳定的排序算法。

第 6 题 每个大于 1 的整数都可以唯一地分解为若干个质因数的乘积（不考虑因子顺序）。

第 7 题 循环队列通过牺牲一个存储单元，可以区分队空和队满两种状态。

第 8 题 Python 子类继承父类后，父类中定义的公开方法，在子类实例对象上一定可以直接对外访问调用。

第 9 题 使用滚动数组优化动态规划时，通常只能降低空间复杂度，不能降低时间复杂度。

第 10 题 一个三角形的三条边的边长分别为 5、12、13，则它的面积为 30。

3 编程题（每题 25 分，共 50 分）

3.1 编程题 1

- **试题名称**：生成树计数
- **时间限制**：1.0 s
- **内存限制**：512.0 MB

3.1.1 题目描述

给定一张有 n 个顶点 m 条边的无向连通图 G，顶点依次以 $1, 2, \ldots, n$ 编号。G 有以下特殊的性质：

- G 中的每条边至多属于一个简单环。
- G 中没有重边与自环。

简单环是指环中顶点互不相同，且不经过重复边的回路。

请你求出 G 的不同生成树的数量。两棵生成树不同，当且仅当存在一条边在其中一棵生成树中出现，而不在另一棵生成树中出现。

由于答案可能很大，你只要求出答案对 998244353 取模的结果。

3.1.2 输入格式

第一行，两个正整数 n, m，分别表示 G 的顶点数与边数。

接下来 m 行，每行两个整数 u_i, v_i，表示一条连接顶点 u_i, v_i 的无向边。

3.1.3 输出格式

输出一行，一个整数，表示 G 的不同生成树的数量对 998244353 取模的结果。

3.1.4 样例

3.1.5 输入样例 1

```
7 8
1 2
2 3
3 1
3 4
4 5
5 6
6 7
7 4
```

3.1.6 输出样例 1

```
12
```

3.1.7 输入样例 2

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

3.1.8 输出样例 2

```
1
```

3.1.9 数据范围

对于 40% 的测试点，保证 $1 \leq n \leq 8$，$1 \leq m \leq 10$。

对于 60% 的测试点，保证 $1 \leq n \leq 2000$，$1 \leq m \leq 2000$。

对于所有测试点，保证 $1 \leq n \leq 10^5$，$1 \leq m \leq 10^5$，$1 \leq u_i, v_i \leq n$。

3.1.10 参考程序

```python
import sys
sys.setrecursionlimit(10 ** 5)
# 顶点数与边的数目
n, m = map(int, input().split())
# 邻接表
e = [[] for _ in range(n + 1)]
# 记录无向图
for i in range(m):
    u, v = map(int, input().split())
    e[u].append(v)
    e[v].append(u)
mod = 998244353
# d[i]: 从节点1出发到节点i有多少跳
# dfs过程中，如果发现某一个节点曾经被访问过（值不为0），那么就表明发现了一个环
# 环的大小是当前节点的d值与曾经访问过的节点的d值之差+1
d = [0] * (n + 1)
# 从节点p出发，进行dfs，抵达了u节点，对于与u相邻的节点：
# 如果节点曾经被访问过，则表示发现了新的环，将这个环上的不同生成树的数量乘到总值上
# 如果该节点没被访问过，则继续dfs
def dfs(u, p=0):
    # 节点u到最初的节点1的距离，需要在d[p]基础上+1
    d[u] = d[p] + 1
    # 总生成树数量
    r = 1
    for v in e[u]:
        if v == p:
            continue
        # 这个节点没有被访问过，不论这个节点和当前节点是否在同一个环上，都进行下一步dfs
        # 如果这个节点在一个环上，那么等我们绕一圈回来会触发下一个分支，把环上的生成树数量计入
        # 如果这个节点不在一个环上，我们会把这个节点连接的其它环上可能的生成树计入
        if not d[v]:
            r = r * dfs(v, u) % mod
        # 这个节点曾经被访问过，这意味着绕着环走了一圈，这个时候需要乘上这个环的节点数量
        if d[v] < d[u]:
            r = r * (d[u] - d[v] + 1) % mod
        # 如果什么相邻节点都没有，那么返回的就是1，表示后面没有新的节点
    return r
print(dfs(1))
```

3.2 编程题 2

- **试题名称**：末班车
- **时间限制**：5.0 s
- **内存限制**：512.0 MB

3.2.1 题目描述

城市里有 n 个地铁站以及 m 条地铁线路，地铁站依次以 $1, 2, \ldots, n$ 编号。

第 i 条地铁线路（$1 \le i \le m$）的列车从地铁站 u_i **单向**驶向地铁站 v_i，最晚发车时间为第 l_i 分钟，途中行驶需要 t_i 分钟。从第 0 分钟到第 l_i 分钟，每分钟都会有一班列车从地铁站 u_i 发出。第 x 分钟（$0 \le x \le l_i$）发出的列车会在第 $x + t_i$ 分钟到达地铁站 v_i，乘坐这班列车的乘客可以换乘第 $x + t_i$ 分钟以及之后的所有从地铁站 v_i 发出的任意线路的列车。

现在有 q 组询问。第 i 组询问（$1 \le i \le q$）给出起点地铁站编号 x_i，终点地铁站编号 y_i 以及出发时间 s_i，你需要判断第 s_i 分钟从地铁站 x_i 出发是否能到达地铁站 y_i。第 s_i 分钟从地铁站 x_i 出发意味着你可以乘坐第 s_i 分钟以及之后的所有从地铁站 x_i 发出的任意线路的列车。

3.2.2 输入格式

第一行，三个正整数 n, m, q，分别表示地铁站数量，地铁线路数量，询问数量。

接下来 m 行，每行四个整数 u_i, v_i, l_i, t_i，分别表示地铁线路的起点，终点，最晚发车时间，行驶所需时间。

接下来 q 行，每行三个整数 x_i, y_i, s_i，分别表示行程起点，行程终点，出发时间。

3.2.3 输出格式

输出共 q 行。对于每组询问，如果第 s_i 分钟从地铁站 x_i 出发能到达地铁站 y_i 则输出一行 `Yes`，否则输出一行 `No`。请注意输出区分大小写。

3.2.4 样例

3.2.5 输入样例

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

3.2.6 输出样例

```
Yes
Yes
No
Yes
No
```

3.2.7 数据范围

对于 40% 的测试点，保证 $q \leq 100$。

对于所有测试点，保证：

- $1 \leq n \leq 500$
- $1 \leq m \leq 1000$
- $1 \leq q \leq 5 \times 10^5$
- $1 \leq u_i, v_i, x_i, y_i \leq n$
- $0 \leq l_i, s_i \leq 10^5$
- $1 \leq t_i \leq 10^4$

3.2.8 参考程序

```python
n, m, q = map(int, input().split())
e = [[] for _ in range(n + 1)]
for _ in range(m):
    u, v, l, t = map(int, input().split())
    e[v].append([u, l, t])
oo = 10 ** 9

# 计算各个节点到节点y的最晚出发时间
def calc(y):
    # 各个节点的最晚出发时间，初始都是负无穷
    d = [-oo] * (n + 1)
    # 目标节点可以任意晚出发
    d[y] = oo
    # 各个节点的最晚出发时间是否已经确认
    vis = [False] * (n + 1)

    # 遍历当前的所有节点u，寻找一个这样一个节点
    # 1. 其最晚出发时间d[u]被更新过
    # 2. 我们没确认d[u]就是其最晚出发时间
    # 3. 在符合1,2的节点中，d[u]最大
    for _ in range(n):
        u = 0
        for i in range(1, n + 1):
            # 判断是否符合条件1,2，记录d[u]最大的一个
            if not vis[i] and d[i] > d[u]:
                u = i
        # 如果所有节点都不符合条件1,2则结束算法
        if u == 0:
            break
        # 符合条件1,2,3的节点，确定d[u]为其最晚出发时间
        vis[u] = True
        # 对于每一个和u相邻的节点，更新其最晚出发时间
        for v, l, t in e[u]:
            # 如果这个节点先到u再到目标节点y，那么其最晚出发时间:
            # 1. 该节点最晚发车时间
            # 2. 比d[u]早t，使得其出发后能从u坐末班车到y
            w = min(d[u] - t, l)
            # w<0表示不可能通过u到y
            if w < 0:
                continue
            # 如果w更大，则更新d[v]
            if w > d[v]:
                d[v] = w
    return d
d = [[]]
# 提前存储好各个节点抵达节点i的最晚出发时间
# d[a][b]表示从节点b去往节点a的最晚出发时间
for i in range(1, n + 1):
    d.append(calc(i))
for _ in range(q):
    x, y, s = map(int, input().split())
    if s <= d[y][x]:
        print('Yes')
    else:
        print('No')
```