C++ 八级

2026 年 03 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	D	B	B	C	B	C	B	B	B	C	C	B	A	C	A

第 1 题 某班级有8名男生和6名女生，现要选出3人组成学习小组，要求小组中至少有1名男生和1名女生，则不同的选法共有（ ）种。

- A. 112
- B. 168
- C. 224
- D. 288

第 2 题 在杨辉三角中，从第0行开始计数，第10行的所有数之和为（ ）。

- A. 512
- B. 1024
- C. 2048
- D. 4096

第 3 题 下列代码实现了快速幂算法，其时间复杂度为（ ）。

```
long long fastPow(long long b, long long e, long long mod) {
    long long result = 1;
    while (e > 0) {
        if (e & 1)
            result = result * b % mod;
        b = b * b % mod;
        e >>= 1;
    }
    return result;
}
```

- A. $O(\log b)$
- B. $O(\log e)$
- C. $O(\log mod)$
- D. $O(e)$

第 4 题 从5本不同的数学书和4本不同的物理书中选取3本书，要求至少包含1本数学书，则不同的选法有（ ）种。

- A. 60

- [ ] **B.** 74
- [ ] **C.** 80
- [ ] **D.** 84

第 5 题 在二叉搜索树（BST）中，若中序遍历的序列为 `{1, 2, 3, 4, 5}`，且先序遍历的第一个序列元素为 `3`，则下列说法正确的是（ ）。

- [ ] **A.** 该树一定是一棵完全二叉树
- [ ] **B.** 元素4和5不可能是兄弟节点
- [ ] **C.** 元素1所在节点的深度可能大于3（根节点深度为1）
- [ ] **D.** 元素2一定是元素1的父节点

第 6 题 在一个有向带权图中，使用Dijkstra算法求单源最短路时，若使用优先队列（小根堆）优化，其时间复杂度为（ ）。

- [ ] **A.** $O(V^2)$
- [ ] **B.** $O(V \cdot E)$
- [ ] **C.** $O((V + E) \log V)$
- [ ] **D.** $O(V^2 \log V)$

第 7 题 对于含 n 个顶点（$n \geq 2$）的连通加权有向图，若图中不存在负权环，则任意两点之间的最短路径（简单路径）最多包含（ ）条边。

- [ ] **A.** n
- [ ] **B.** $n - 1$
- [ ] **C.** $n + 1$
- [ ] **D.** 无法确定，取决于图的具体边数

第 8 题 在使用Floyd算法求任意两点间最短路径时，时间复杂度为 $O(V^3)$。若在某次算法执行前，已经用 Dijkstra 算法正确求出了所有点对的最短路并存入了 `dist` 数组。如果此时继续对该 `dist` 数组执行一次完整的 Floyd 算法过程（无任何提前终止），执行完毕后 `dist` 数组内的值（ ）。

- [ ] **A.** 会发生改变，因为 Floyd 又做了一次松弛
- [ ] **B.** 不会发生改变
- [ ] **C.** 可能变大，因为未针对已有最短路优化
- [ ] **D.** 可能在某些负权图中陷入死循环

第 9 题 关于图论中的最短路径算法，下列说法中严格正确的是（ ）。

- [ ] **A.** Dijkstra 算法能够高效处理包含负权边的有向图。
- [ ] **B.** Floyd 算法可以求出任意两点间的最短路径，且允许图中存在负权边（但不能有负权环）。
- [ ] **C.** 单源最短路径算法无法用于无向图，无向图只能通过 BFS 求解。
- [ ] **D.** Dijkstra 算法的每一步必定从当前未访问的节点中，选取距离起始点最远的节点进行松弛操作。

第 10 题 有6个人排成一排照相，其中甲、乙两人必须相邻，且丙不能站在排头的不同排法有（ ）种。

- [ ] **A.** 120
- [ ] **B.** 144

- [ ] **C.** 192
- [ ] **D.** 240

第 11 题 下列代码试图实现Floyd算法求所有点对之间的最短路径，横线处应填入（ ）。

```
void floyd(int n, int dist[][MAXN]) {
    for (int k = 0; k < n; k++)
        for (int i = 0; i < n; i++)
            for (int j = 0; j < n; j++)
                if (__________) // 在此处填入选项
                    dist[i][j] = dist[i][k] + dist[k][j];
}
```

- [ ] **A.** `dist[i][k] + dist[k][j] < dist[i][j]`
- [ ] **B.** `dist[i][k] != INF && dist[k][j] != INF`
- [ ] **C.** `dist[i][k] != INF && dist[k][j] != INF && dist[i][k] + dist[k][j] < dist[i][j]`
- [ ] **D.** `dist[i][j] == INF`

第 12 题 用数字 0、1、2、3、4 组成无重复数字的五位偶数，共有（ ）个。

- [ ] **A.** 48
- [ ] **B.** 60
- [ ] **C.** 72
- [ ] **D.** 96

第 13 题 在一个无向带权图中，若使用 Prim 算法从顶点 0 开始构造最小生成树（边权均为正整数，且 `graph[u][v] == 0` 表示无边），下列代码中横线处应填入（ ）。

```
int prim(vector<vector<int>>& graph, int n) {
    vector<bool> inMST(n, false);
    vector<int> minEdge(n, INT_MAX);
    minEdge[0] = 0;
    int result = 0;
    for (int i = 0; i < n; i++) {
        int u = -1;
        for (int j = 0; j < n; j++)
            if (!inMST[j] && (u == -1 || minEdge[j] < minEdge[u]))
                u = j;
        inMST[u] = true;
        result += minEdge[u];
        for (int v = 0; v < n; v++)
            if (__________) // 在此处填入选项
                minEdge[v] = graph[u][v];
    }
    return result;
}
```

- [ ] **A.** `graph[u][v] && !inMST[v] && graph[u][v] < minEdge[v]`
- [ ] **B.** `!inMST[v] && graph[u][v] < minEdge[v]`
- [ ] **C.** `graph[u][v] > 0 && !inMST[v]`
- [ ] **D.** `!inMST[v] && minEdge[v] > 0`

第 14 题 已知三个点 $A(x_1,y_1), B(x_2,y_2), C(x_3,y_3)$ 在平面直角坐标系中的坐标。下列 C++ 表达式中，在**精度误差范围1e-8内**能正确计算判断这三个点是三点共线的表达式是（ ）。

- [ ] **A.** `(x2-x1)/(y2-y1) == (x3-x1)/(y3-y1)`
- [ ] **B.** `(x2-x1)*(y3-y1)-(x3-x1)*(y2-y1) == 0`
- [ ] **C.** `fabs((x2-x1)*(y3-y1)-(x3-x1)*(y2-y1)) < 1e-8`
- [ ] **D.** `fabs((x2-x1)/(y2-y1)-(x3-x1)/(y3-y1)) < 1e-8`

第 15 题 在64位操作系统下（LP64 / LLP64 模型），下面代码的输出结果是（）。

```cpp
#include <iostream>
using namespace std;

int main() {
    int a[4] = {1, 2, 3, 4};
    int (*p)[4] = &a;
    int *q = a;

    cout << sizeof(a) << " ";
    cout << sizeof(p) << " ";
    cout << sizeof(p + 1) << " ";
    cout << sizeof(q + 1) << " ";
    cout << (p + 1) - p << " ";
    cout << (q + 1) - q << endl;
}
```

- [ ] **A.** `16 8 8 8 1 1`
- [ ] **B.** `16 8 16 8 1 1`
- [ ] **C.** `16 8 8 4 4 1`
- [ ] **D.** `16 8 8 8 4 1`

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 在C++中，若结构体中包含一个 `static` 成员变量，则该变量的存储空间属于结构体对象的一部分。（）

第 2 题 对于任意正整数n，二项式$(a+b)^n$展开式中各项的二项式系数之和等于2^n。（）

第 3 题 在C++中，若函数参数类型为 `const int &`，则该参数既可以绑定左值，也可以绑定右值。（）

第 4 题 若一个无向图的最小生成树唯一，则图中所有边权必定各不相同。（）

第 5 题 使用快速排序对n个元素进行排序时，无论最好、最坏还是平均情况，时间复杂度均为$O(n\log n)$。（）

第 6 题 若一个图中所有顶点的度数为偶数，则一定存在欧拉回路。（）

第 7 题 使用倍增法预处理区间最值问题时，预处理的时间复杂度为$O(n\log n)$，查询的时间复杂度为$O(1)$。（）

第 8 题 如果将一个连通无向图G_1中所有边的权值都统一增加同一个正整数常数C，形成图G_2。则G_1的最小生成树中每条边在G_2中对应的边组成的树，一定是G_2的最小生成树。（）

第 9 题 在图论算法中，Kruskal算法和Prim算法都可以用来求解最小生成树，且这两者的贪心策略无论在任何连通无向图上求得的最小生成树总边权和必定相同。（）

第 10 题 在动态规划问题中，“状态转移方程+递推”和“递归+记忆化搜索”通常是解决同一问题的两种不同实现方式，它们的时间复杂度总是相同的。（）

3 编程题（每题 25 分，共 50 分）

3.1 编程题 1

- **试题名称**：消息查找
- **时间限制**：1.0 s
- **内存限制**：512.0 MB

3.1.1 题目描述

小 A 的消息记录中有 n 条消息，依次以 $1, 2, \ldots, n$ 编号。编号小的消息发送时间早于编号大的消息。

一条消息可以引用一条编号小于它的消息，也可以不引用消息。小 A 注意到消息记录里有引用的消息数量不会非常多。消息记录的一个例子是：

- 【消息 1】小 A：有人做了今天的第一题吗？
- 【消息 2】小 A：我第一题 WA 了，可能是什么原因？
- 【消息 3：引用消息 1】小 B：我我我
- 【消息 4：引用消息 2】小 C：我也 WA 了
- 【消息 5：引用消息 2】小 B：是不是没开 `long long`？
- 【消息 6：引用消息 5】小 A：改了就 AC 了，太厉害了！

对于消息 i（$1 \le i \le n$），小 A 以 r_i 标记消息 i 是否有引用，以及所引用的消息编号。如果 $r_i > 0$，则消息 i 为引用了消息 r_i；如果 $r_i = 0$，则消息 i 没有引用消息。

消息记录里有非常多条消息。为了快速查找所需要的消息，小 A 准备实现一个简单的消息查找工具。消息查找工具任意时刻只能定位恰好一条消息，如果当前位于消息 i（$1 < i \le n$），那么接下来可以选择以下两种操作之一：

- 定位到消息 $i - 1$；
- 如果消息 i 引用了消息 r_i，定位到消息 r_i。

以上操作可以执行任意次（包括零次）。

小 A 有 q 次询问。在第 k（$1 \le k \le q$）次询问中，小 A 给出消息编号 x_k, y_k（$y_k < x_k$）。小 A 想知道，如果当前消息查找工具位于 x_k，至少需要多少次操作才能定位到消息 y_k。

3.1.2 输入格式

第一行，两个正整数 n, q，分别表示消息条数与询问次数。

第二行，n 个非负整数 $r_1, r_2, \ldots, r_n$，表示消息的引用关系，具体含义见题目描述。

接下来 q 行中的第 k 行（$1 \le k \le q$）包含两个正整数 x_k, y_k，表示一次询问。

保证至多只有 1000 条引用消息。

3.1.3 输出格式

输出 q 行，每行一个整数，表示将界面从消息 x_k 切换到消息 y_k 所需的最少操作次数。

3.1.4 样例

3.1.4.1 输入样例 1

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

3.1.4.2 输出样例 1

```
2
2
3
```

3.1.4.3 输入样例 2

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

3.1.4.4 输出样例 2

```
1
2
2
2
1
```

3.1.5 数据范围

对于 40% 的测试点，保证 $1 \leq n \leq 2000$，$1 \leq q \leq 2000$。

对于所有测试点，保证 $1 \leq n \leq 10^5$，$1 \leq q \leq 10^5$，$0 \leq r_i < i$，$1 \leq y_k < x_k \leq n$，保证至多有 1000 条引用消息。

3.1.6 参考程序

```cpp
#include <cstdio>
#include <algorithm>

using namespace std;

const int N = 1e5 + 5;
const int C = 2e3 + 5;
const int oo = 1e9;

int n, q;
int r[N], mark[N], pos[N];
int p[C], cnt;
int d[C][C];
int pre[N], suf[N];

int main() {
    scanf("%d%d", &n, &q);
    for (int i = 1; i <= n; i++) {
        scanf("%d", &r[i]);
        if (r[i])
            mark[i] = mark[r[i]] = 1;
    }
    for (int i = 1; i <= n; i++)
        if (mark[i]) {
            p[++cnt] = i;
            pos[i] = cnt;
        }
    for (int i = 1; i <= cnt; i++) {
        for (int j = 1; j <= i; j++)
            d[i][j] = oo;
        d[i][i] = 0;
        for (int j = i; j > 1; j--) {
            d[i][j - 1] = min(d[i][j - 1], d[i][j] + p[j] - p[j - 1]);
            if (r[p[j]]) {
                int k = pos[r[p[j]]];
                d[i][k] = min(d[i][k], d[i][j] + 1);
            }
        }
    }
    for (int i = 1; i <= n; i++) {
        pre[i] = pre[i - 1];
        if (mark[i])
            pre[i] = i;
    }
    suf[n + 1] = n + 1;
    for (int i = n; i; i--) {
        suf[i] = suf[i + 1];
        if (mark[i])
            suf[i] = i;
    }
    while (q--) {
        int x, y;
        scanf("%d%d", &x, &y);
        if (pre[x] < suf[y])
            printf("%d\n", x - y);
        else
            printf("%d\n", x - pre[x] + d[pos[pre[x]]][pos[suf[y]]] + suf[y] - y);
    }
    return 0;
}
```

3.2 编程题 2

- **试题名称**：子图最短路
- **时间限制**：1.0 s
- **内存限制**：512.0 MB

3.2.1 题目描述

给定包含 n 个结点 m 条边的**带权无向图** G，结点依次以 $1, 2, \ldots, n$ 编号。第 i（$1 \le i \le m$）条边连接编号为 u_i 与 v_i 的两个结点，权值为 w_i。

对于指定的 $1 \le \ell \le r \le n$，按以下方式构造图 G 的子图 $G(\ell, r)$：

- 保留 G 中编号在区间 $[\ell, r]$ 中的结点。删去其它编号不在 $[\ell, r]$ 中的结点以及与之相连的边。剩余的结点和边构成子图 $G(\ell, r)$。

对于 $G(\ell, r)$ 中的任意结点 u, v 应有 $\ell \le u, v \le r$。记 u, v 在子图 $G(\ell, r)$ 上的最短距离为 $d(\ell, r, u, v)$。特殊地，若 u, v 在子图 $G(\ell, r)$ 上不连通，则认为 $d(\ell, r, u, v) = 0$。

你需要求出 $\sum_{\ell=1}^{n} \sum_{r=\ell}^{n} \sum_{u=\ell}^{r} \sum_{v=u}^{r} d(\ell, r, u, v)$ 对 10^9 取模的结果。

- 题目中的英文字母 l 使用了特殊写法 ℓ，以避免英文字母 l 与数字 1 混淆。

3.2.2 输入格式

第一行，两个正整数 n, m，表示结点数与边数。

接下来 m 行，第 i（$1 \le i \le m$）行包含三个正整数 u_i, v_i, w_i，表示一条连接结点 u_i, v_i 的权值为 w_i 的边。

3.2.3 输出格式

输出一行，一个整数，表示 $\sum_{\ell=1}^{n} \sum_{r=\ell}^{n} \sum_{u=\ell}^{r} \sum_{v=u}^{r} d(\ell, r, u, v)$ 对 10^9 取模的结果。

3.2.4 样例

3.2.4.1 输入样例 1

```
3 2
1 2 1
2 3 2
```

3.2.4.2 输出样例 1

```
9
```

3.2.4.3 输入样例 2

```
4 6
1 2 100
2 3 100
3 4 100
1 3 10
2 4 10
1 4 1
```

3.2.4.4 输出样例 2

```
784
```

3.2.5 数据范围

对于 40% 的测试点，保证 $2 \leq n \leq 20$。

对于所有测试点，保证 $2 \leq n \leq 100$，$2 \leq m \leq \frac{n(n-1)}{2}$，$1 \leq u_i, v_i \leq n$，$1 \leq w_i \leq 10^6$。图中可能存在重边。

3.2.6 参考程序

```cpp
#include <cstdio>
#include <algorithm>

using namespace std;

const int N = 105;
const int mod = 1e9;

int n, m;
int f[N][N];
int g[N][N];
int ans;

int main() {
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++)
            f[i][j] = mod;
        f[i][i] = 0;
    }
    for (int i = 1; i <= m; i++) {
        int u, v, w;
        scanf("%d%d%d", &u, &v, &w);
        f[u][v] = min(f[u][v], w);
        f[v][u] = min(f[v][u], w);
    }
    for (int l = 1; l <= n; l++) {
        for (int i = 1; i <= n; i++)
            for (int j = 1; j <= n; j++)
                g[i][j] = f[i][j];
        for (int r = l; r <= n; r++) {
            for (int i = 1; i <= n; i++)
                for (int j = 1; j <= n; j++)
                    g[i][j] = min(g[i][j], g[i][r] + g[r][j]);
            for (int i = l; i <= r; i++)
                for (int j = i; j <= r; j++)
                    ans = (ans + g[i][j]) % mod;
        }
    }
    printf("%d\n", ans);
    return 0;
}
```