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

Python 八级（样题）

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	D	B	C	B	C	C	C	D	B	A	C	C	C

第 1 题 从A城到C城需要经过B城，其中A到B可选高铁和飞机，B到C可以自驾或打的，请问A到C有几种交通选择（ ）。

- **A.** 2
- **B.** 4
- **C.** 8
- **D.** 不知道

第 2 题 下面程序输出的n的值是（ ）。

```
n = 0

for a in range(1, 5):
    for b in range(1, 5):
        for c in range(1, 5):
            for d in range(1, 5):
                if a != b and a != c and a != d and b != c and b != d and c != d:
                    if a != 4:
                        print(a * 100 + b * 100 + c * 10 + d, " ", sep = "",end = "")
                        n += 1
print("\n", n)
```

- **A.** 4
- **B.** 12
- **C.** 18
- **D.** 24

第 3 题 对$(a+b)^5$,想求出 a^3b^2 的系数可以使用（ ）

- **A.** 杨氏三角
- **B.** 祖冲之角
- **C.** 勾股三角
- **D.** 杨辉三角

第 4 题 对于 4 个结点的简单有向图，最少（ ）条边可以形成一条覆盖所有顶点的环。

- **A.** 5
- **B.** 4

☐ **C.** 3

☐ **D.** 2

第 5 题 对正整数 a 和 n（n 为 2 的正整数次幂），下面求 a^n 值的方法是（ ）

```
def fan(a, n):
    n = int(n)

    if n == 0:
        return 1
    if n == 1:
        return a

    s = fan(a, n/2)

    return s * s
```

☐ **A.** 折半

☐ **B.** 二分

☐ **C.** 倍增

☐ **D.** 迭代

第 6 题 平面内，通过一点可以作( )条平行于给定直线的直线?

☐ **A.** 0

☐ **B.** 1

☐ **C.** 2

☐ **D.** 无限多

第 7 题 已知Python变量pi被赋值3.14159。如果一个等边三角形的边长为4，下列（ ）表达式可以求其面积 。

☐ **A.** 16*math.sin(pi/3)

☐ **B.** 16*math.cos(pi/3)

☐ **C.** 8*math.sin(pi/3)

☐ **D.** 8*math.cos(pi/3)

第 8 题 新，重新截图 8. 下面程序使用BFS遍历一个有n个顶点、边权都为1的无向图G，下面说法正确的是（ ）。

```
N = 2023

n = 10
G = [[] for i in range(N)]
q, dis = [0] * N, [0] * N


def BFS(st):
    hd, tl = 1,0
    for i in range(1,n+1):
        dis[i] = -1

    tl += 1
    q[tl] = st
    dis[st] = 0

    while hd <= tl:
        u = q[hd]
        hd += 1
        for i in range(0, len(G[u])):
            v = G[u][i]
            if(dis[v] != -1):
                continue

            dis[v] = dis[u] + 1
            tl += 1
            q[tl] = v
```

- [ ] **A.** tl记录遍历的结点数
- [ ] **B.** dis按照贪心法变化
- [ ] **C.** dis存储st到其他顶点的距离
- [ ] **D.** 算法复杂度是O(n^2)

第 9 题 下面的冒泡排序中尝试提前结束比较过程，横线处应该填写的代码是（ ）。

```
def BubbleSort(lst) :
    i = len(lst)
    while i > 1:
        lastExchangeIndex = 1
        for j in range(i-1):
            if lst[j + 1] < lst[j]:
                lst[j + 1], lst[j] = lst[j], lst[j + 1]
                lastExchangeIndex = j
        ____________________________
```

- [ ] **A.** i = lastExchangeIndex + 1
- [ ] **B.** i = lastExchangeIndex -1
- [ ] **C.** i = lastExchangeIndex
- [ ] **D.** i = lastExchangeIndex - j

第 10 题 对数列3、4、7、12、19、28、39求和，除简单累加外，还可以用下面（ ）来直接计算。

- [ ] **A.** 等差数列求和
- [ ] **B.** 等比数列求和
- [ ] **C.** 斐波拉契数列
- [ ] **D.** 其他某种有规律序列

第 11 题 约定杨辉三角形第0行只有1个元素是1，第1行有2个元素都是1，第四行的所有元素之和是（ ）。

- [ ] **A.** 8

- [ ] **B.** 16
- [ ] **C.** 24
- [ ] **D.** 32

第 12 题 下列程序的输出是 ( ) 。

```
sum = 0
for x in range(-10, 10+1):
    if (3 * x + 5 >= -11) and (3 * x + 5 <= 11):
        sum += x
print(sum)
```

- [ ] **A.** -12
- [ ] **B.** 0
- [ ] **C.** 55
- [ ] **D.** -55

第 13 题 对于具有n个元素的二叉排序树（又名二分查找树）进行前序遍历，其时间复杂度是( )。

- [ ] **A.** O(1)
- [ ] **B.** O(log n)
- [ ] **C.** O(n)
- [ ] **D.** O(n^2)

第 14 题 Dijkstra的算法在实现时一般可以选用（ ）来提高效率?

- [ ] **A.** 数组
- [ ] **B.** 链表
- [ ] **C.** 堆
- [ ] **D.** 栈

第 15 题 有关下面代码的说法正确的是（ ）。

```
class Node:
    def __init__(self,Val,Nxt = None):
        self.Val = Val
        self.Next = Nxt

firstNode = Node(10)
firstNode.Next = Node(100)
firstNode.Next.Next = Node(111, firstNode)
```

- [ ] **A.** 上述代码构成单向链表
- [ ] **B.** 上述代码构成双向链表
- [ ] **C.** 上述代码构成循环链表
- [ ] **D.** 上述代码构成指针链表

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 学校组织班际排球赛，每个班级可以派男女各一个参赛队伍，每队5人。班级A的每位同学都可以报名，那可以用加法原理来计算A班有多少支候选队伍。( )

第 2 题 若a,x,b都是float类型，则对方程a*x+b=0求解的程序中可以直接用x=-b/a来求解。( )

第 3 题 从15本不同的书中选3本，总共有455种方法。（ ）

第 4 题 连通图G有n个顶点m条边，须删除m-n+1条边后才能变成一棵生成树。（ ）

第 5 题 在Python语言中，所有int类型的值，经过若干次左移操作（<<）后，它们的值总会变为0。（ ）

第 6 题 如果一个四边形的对角线互相平分，并且2条对角线的长度都为8，那么这个四边形的面积一定是32。（ ）

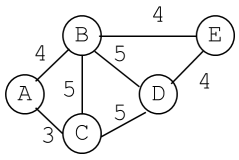
第 7 题 最小生成树的权值是指生成树所有边的权值之和最小。（ ）

第 8 题 如果一个图中所有边的权重都为正数，则Floyd算法可以求出该图中任意两个顶点间的最短路径。（ ）

第 9 题 下面是图的深度遍历的代码，则横线处可以填入：if(vis[x]) return。（ ）

```
def dfs(x):
    __________ #补充代码
    vis[x] = 1   #访问当前节点
    print(x)
    i = vFirst[x]
    while i != -1: #遍历每一个相邻的顶点
        dfs(e[i].v)
        i = e[i].next
```

第 10 题 下图中A到E的Dijkstra单源最短路可以在第2次探索中找到。（ ）



3　编程题（每题 25 分，共 50 分）

3.1　编程题 1

- **试题名称**：区间
- **时间限制**：1.0 s
- **内存限制**：128.0 MB

3.1.1　问题描述

小杨有一个长度为 n 的正整数序列 A。

小杨有 q 次询问。第 i 次（$1 \leq i \leq q$）询问时，小杨会给出 l_i, r_i, x_i，请你求出 x_i 在 $A_{l_i}, A_{l_i+1}, \ldots, A_{r_i}$ 中出现的次数。

3.1.2 输入描述

第一行包含一个正整数 T，表示数据组数。

对于每组数据：第一行包含一个正整数 n，表示序列 A 的长度。第二行包含n 个正整数 $A_1, A_2, \ldots, A_n$，表示序列 A 。第三行包含一个正整数 q，表示询问次数。接下来 q 行，每行三个正整数 l_i, r_i, x_i，表示一组询问。

3.1.3 输出描述

对于每组数据，输出 q 行。第 i 行（$1 \leq i \leq q$）输出一个非负整数，表示第 i 次询问的答案。

3.1.4 样例输入1

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

3.1.5 样例输出1

```
0
2
0
1
```

3.1.6 子任务

子任务编号	分值	n	q	$\max A_i$
1	30	≤ 100	≤ 100	≤ 10
2	30	$\leq 10^5$	$\leq 10^5$	$\leq 10^5$
3	40	$\leq 10^5$	$\leq 10^5$	$\leq 10^9$

对于全部数据，保证有 $1 \leq T \leq 5$，$1 \leq n \leq 10^5$，$1 \leq q \leq 10^5$，$1 \leq A_i \leq 10^9$。

3.1.7 参考程序

```cpp
#include <bits/stdc++.h>
const int maxn=100086;
using namespace std;
int t, n, q;
map<int,vector<int> >mp;
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
    cin>>t;
    while(t--){
        cin>>n;
```

```
        mp.clear();
        for(int i = 1;i <= n;i++)
        {
            int x;
            cin>>x;
            mp[x].push_back(i);
        }
        cin>>q;
        while(q--)
        {
            int l, r, x;
            cin>>l>>r>>x;
            cout<<upper_bound(mp[x].begin(), mp[x].end(), r) -
lower_bound(mp[x].begin(), mp[x].end(), l)<<endl;
        }
    }

}
```

3.2 编程题 2

- **试题名称**：小杨的旅游
- **时间限制**：1.0 s
- **内存限制**：128.0 MB

3.2.1 问题描述

小杨准备前往 B 国旅游。

B 国有 n 座城市，这 n 座城市依次以 1 至 n 编号。城市之间由 $n-1$ 条双向道路连接，任意两座城市之间均可达（即任意两座城市之间存在可达的路径）。

小杨可以通过双向道路在城市之间移动，通过一条双向道路需要 1 单位时间。

B 国城市中有 k 座城市设有传送门。设有传送门的城市的编号依次为 $b_1, b_2, \ldots, b_k$。小杨可以从任意一座设有传送门的城市花费 0 单位时间前往另一座设有传送门的城市。

注：如果两座设有传送门的城市之间存在双向道路，那么小杨可以选择通过双向道路移动，也可以选择通过传送门传送。

小杨计划在 B 国旅游 q 次。第 i 次旅游（$1 \le i \le q$），小杨计划从编号为 u_i 的城市前往编号为 v_i 的城市，小杨希望你能求出所需要的最短时间。

3.2.2 输入描述

第一行包含三个正整数 n, k, q，分别表示 B 国的城市数量，设有传送门的城市数量，以及小杨计划在 B 国旅游的次数。

接下来 $n-1$ 行，每行包含两个正整数 x_i, y_i，表示一条双向道路连接的两座城市的编号。

第 $n+1$ 行包含 k 个正整数 $b_1, b_2, \ldots, b_k$，表示设有传送门的城市的编号。

接下来 q 行，每行包含两个正整数 u_i, v_i，表示小杨第 i 次旅游行程的起点城市编号与终点城市编号。

3.2.3 输出描述

输出共 q 行。第 i 行（$1 \le i \le q$）输出一个非负整数，表示小杨计划第 i 次旅游从编号为 u_i 的城市前往编号为 v_i 的城市所需要的最短时间。

3.2.4 样例输入1

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

3.2.5 样例输出1

```
4
```

3.2.6 样例输入2

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

3.2.7 样例输出2

```
2
1
4
```

3.2.8 子任务

子任务编号	分值	n	k	q
1	30	≤ 500	≤ 500	$=1$
2	30	$\le 2\times 10^5$	$=0$	$\le 2\times 10^5$
3	40	$\le 2\times 10^5$	$\le 2\times 10^5$	$\le 2\times 10^5$

对于全部数据，保证有 $1 \le k \le n \le 2\times 10^5$，$1 \le q \le 2\times 10^5$，$1 \le x_i, y_i \le n$，$1 \le u_i, v_i \le n$。对于所有 $1 \le i \le n-1$，有 $x_i \ne y_i$。

3.2.9 参考程序

```
#include<bits/stdc++.h>
using namespace std;
#define int long long
#define Getchar() p1==p2 and (p2=(p1=Inf)+fread(Inf,1,1<<21,stdin),p1==p2)?EOF:*p1++
#define Putchar(c) p3==p4 and (fwrite(Ouf,1,1<<21,stdout),p3=Ouf),*p3++=c
```

```
char Inf[1<<21],Ouf[1<<21],*p1,*p2,*p3=Ouf,*p4=Ouf+(1<<21);
inline void read(int &x,char c=Getchar())
{
    bool f=c!='-';
    x=0;
    while(c<48 or c>57) c=Getchar(),f&=c!='-';
    while(c>=48 and c<=57) x=(x<<3)+(x<<1)+(c^48),c=Getchar();
    x=f?x:-x;
}
inline void write(int x)
{
    if(x<0) Putchar('-'),x=-x;
    if(x>=10) write(x/10),x%=10;
    Putchar(x^48);
}
int n,k,q,head[200010],mini[200010],dep[200010],siz[200010],hvson[200010],fa[200010],start[200010],pos;
struct edge
{
    int to,next;
};
edge e[400010];
inline void add(const int &x,const int &y)
{
    static int cnt=0;
    e[++cnt].to=y,e[cnt].next=head[x],head[x]=cnt;
}
inline void dfs1(int pos,int fath,int de,int maxi=-0x3f3f3f3f)
{
    fa[pos]=fath,dep[pos]=de,siz[pos]=1;
    for(int i=head[pos];i;i=e[i].next)
        if(e[i].to!=fath)
        {
            dfs1(e[i].to,pos,de+1),siz[pos]+=siz[e[i].to];
            if(siz[e[i].to]>maxi) hvson[pos]=e[i].to,maxi=siz[e[i].to];
        }
}
inline void dfs2(int pos,int Start)
{
    start[pos]=Start;
    if(hvson[pos])
    {
        dfs2(hvson[pos],Start);
        for(int i=head[pos];i;i=e[i].next) if(e[i].to!=fa[pos] && e[i].to!=hvson[pos]) dfs2(e[i].to,e[i].to);
    }
}
inline int lca(int x,int y)
{
    while(start[x]!=start[y])
        if(dep[start[x]]>=dep[start[y]]) x=fa[start[x]];
        else y=fa[start[y]];
    if(dep[x]>=dep[y]) return y;
    return x;
}
```

```cpp
queue<int> qu;
inline void bfs()
{
    while(!qu.empty())
    {
        pos=qu.front(),qu.pop();
        for(int i=head[pos];i;i=e[i].next) if(mini[e[i].to]>mini[pos]+1) mini[e[i].to]=mini[pos]+1,qu.push(e[i].to);
    }
}
signed main()
{
    read(n),read(k),read(q),memset(mini,0x3f,sizeof(mini));
    for(int i=1,x,y;i<n;i++) read(x),read(y),add(x,y),add(y,x);
    dfs1(1,0,1),dfs2(1,1);
    for(int i=1,x;i<=k;i++) read(x),mini[x]=0,qu.push(x);
    bfs();
    for(int i=1,x,y;i<=q;i++)
    {
        read(x),read(y);
        write(min(dep[x]+dep[y]-dep[lca(x,y)]*2,mini[x]+mini[y])),Putchar('\n');
    }
    fwrite(Ouf,1,p3-Ouf,stdout),fflush(stdout);
    return 0;
}
```