一．选择题（30分）

1. G 是一个非连通简单无向图，共有 36 条边，则该图至少有（ ）个顶点。

A. 9　　B. 10　　C. 11　　D. 12

2. 下列排序算法中，（ ）算法是不稳定的。

A. 插入排序　　B. 冒泡排序　　C. 归并排序　　D. 快速排序

3. 关于卡塔兰数$C_n = \frac{(2n)!}{(n+1)!n!}$，下列说法错误的是（ ）。

A. C_n表示含n对括号的合法括号序列的个数。

B. C_n表示长度为n的入栈序列对应的合法出栈序列个数。

C. C_n表示有$n+1$个节点的不同形态的二叉树的个数。

D. C_n表示通过连接顶点而将$n+2$边的凸多边形分成三角形的方法个数。

4. 前序遍历序列与后序遍历序列相同的二叉树为（ ）。

A. 只有根节点的二叉树。

B. 根节点无右子树的二叉树。

C. 非叶子结点只有左子树的二叉树。

D. 非叶子节点只有右子树的二叉树。

5. 若 AVL 树插入元素的过程中发生了旋转操作，则树高（ ）。

   A. 增加 1 或减少 1
   B. 可能变化也可能不变
   C. 一定不改变
   D. 可能变化超过 2

6. 在二叉树查找中，以下查找序列中可能出现的是（ ）。

   A. 930, 207, 922, 265, 260, 200
   B. 87, 768, 456, 372, 326, 378, 365
   C. 726, 521, 201, 328, 384, 319, 365
   D. 48, 260, 570, 307, 340, 380, 361, 365

7. 下列哪种方法是针对闭散列策略的冲突排解方法（ ）。

   A. 公共溢出区法
   B. 线性试探法
   C. 独立链法
   D. 多槽位法

8. 若某算法的计算时间表示为递推关系式$T(N) = 2T\left(\frac{N}{2}\right) + N\log N$，$T(1) = 1$，则该算法的时间复杂度为 ( ) 。

   A. N^2
   B. $N^2\log N$

C. $N\log N$

D. $N\log^2 N$

9. 以下哪个数字最适合应用于双向平方试探的开放定址散列表长（）。

A. 13

B. 17

C. 23

D. 29

10. 下列排序算法种，最坏情况下的渐进时间复杂度最小的是（ ）。

A. 冒泡排序

B. 选择排序

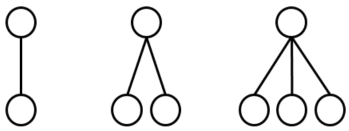
C. 归并排序

D. 快速排序

二. 填空题（35分）

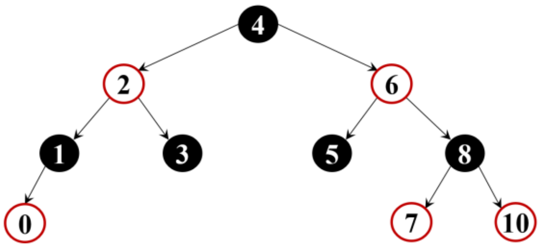
1. 一棵二叉树的中序遍历为BHEADJGCIKF，后序遍历为BEJDAHKIFCG，则前序遍历为________（3分）

2. 以下三棵树形成的森林，转化为二叉树后，树中属于右孩子的节点数为_____个。（3分）

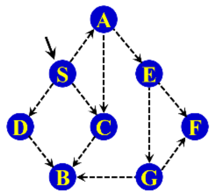


3. 使用RPN及栈的方式计算表达式(2+3!)×(4+6-7)的值，计算过程中，操作符栈最多的元素个数为_____个，操作数栈中元素个数最多为______个。（4分）

4. 以下红黑树，插入关键码9之后，红黑树中黑节点个数为_____个。该红黑树所对应的4阶B树，删除关键码4之后，根节点关键码为______(如根节点关键码从小到大为1,2,3，则填入123；如关键码从小到大为1,2，则填入12)；继续删除关键码2，则根节点关键码为______。（注：删除非叶子节点使用直接前驱替代；节点合并时若左右皆可合并选取左边合并）（6分）



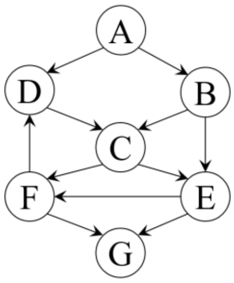
5. 对下图进行以下代码进行深度优先遍历，假设各节点被DISCOVERED的顺序为：SCBDAEGF，则各节点被VISITED的顺序为___________。（3分）




```
template <typename Tv, typename Te>
void Graph<Tv, Te>::DFS ( int v) {
```

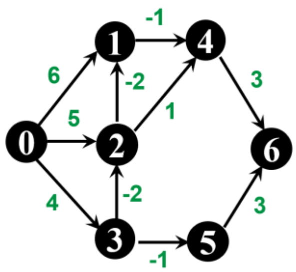
```
    status (v) = DISCOVERED;    //设置v已被发现
    for (int u = firstNbr ( v ); -1 < u; u = nextNbr ( v, u ))
        if (UNDISCOVERED == status ( u ) ) DFS ( u ); //深搜v所有未访问过
邻居u
    status (v) = VISITED; //设置v已被访问完毕
}
```

6. 对下面有向图采用基于递归的深度优先遍历，则满足遍历序列的包括_______。（假设a），b），d）满足，则填入abd）（4分）



a) ABEGCFD
b) ADCEFGB
c) ADCFGBE
d) ADCEGFB
e) ABCEFGD
f) ABCFDEG
g) ADCFGEB

7. 对下面带负值边的有向图进行Bellman&Ford算法求取从0号节点到其他节点的最短路，则在求解的过程中，第三步迭代，即求解到从源点0出发最多经过3条边到达各点的最短路，此时各点最短路径长度的总和为___________。使用Bellman&Ford算法最终得到从源点0出发到其他各点的最短路，此时能否形成一棵最短路径树？________(请在空格内填入“能”，“不能”或“不确定”)。(6分)



8. 对序列5,0,2,9,6,7,4,1,8,3进行快速排序，每次使用首个元素作为轴点，当递归进行完第一趟排序后，首元素5作为轴点放置到正确的位置，则在递归第二趟（第二层，包括左和右）排序后，形成的排序序列为__________。（若结果为0,1,2,3,4,5,6,7,8,9,则填入0123456789）(3分)

9. 由2022个结点组成的完全二叉树做层次遍历，辅助队列的容量至少为__________。(3分)

三、算法设计与分析（35分）

1. 两稀疏矩阵A与B进行乘法，请描述快速算法的核心思想并给出复杂度分析。（10分）
2. 代码写作与分析（25分）疫情下的物流运输

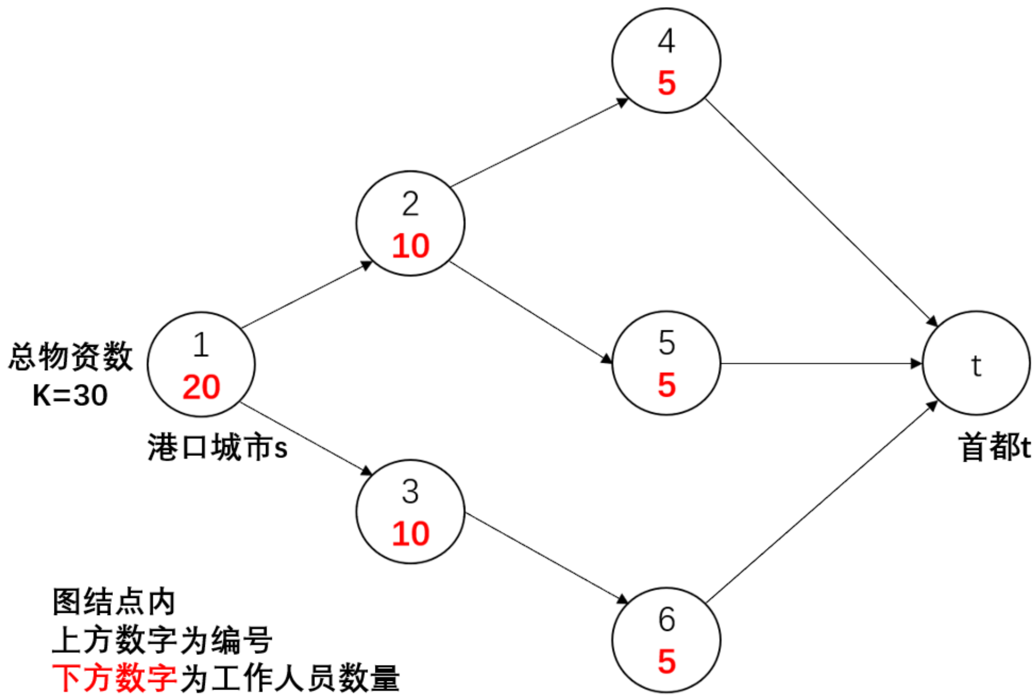
国家 A 总共由 n 个城市和 1 个首都 t 构成，这 n 个城镇之间由 n-1 条单向道路连接且构成一个二叉树，这个树的根节点为港口城市 s，每个在叶子节点的城市和首都之间由 1 条单向道路连接。现在港口城市 s 收到了总共 K 个防疫物资，他们需要把这些防疫物资运送到首

都城市 t。然而每个城市的快递工作人员是有限的，第 i 个城市的工作人员数目为 ci，且 1 个快递工作人员只能处理 1 个防疫物资。也就是说，当一个城市想要转运 a 个防疫物资到其他的邻近城市时，它就需要 a 个快递工作人员。

问题 1. 在 s 收到总共 K 个防疫物资的条件下，最多能运送多少物资到首都城市 t。请写出算法设计思路，并给出复杂度分析，写出代码（算法设计：7 分，复杂度分析：3 分，代码：5 分）

问题 2. 由于疫情的蔓延，这 n 个城市当中随机会有 T 个城市受到疫情影响，这些城市能工作的快递人员数量会减半（整数下取整）。请给出随机哪些城市受到影响时，最终能运送到 t 的物资最多以及哪些城市受到影响时，能运送到的物资最少。 （10 分，讨论当 T=1 时的情况 5 分，T>1 的情况 5 分）（算法设计：7 分，复杂度分析：3 分，不需写代码）

样例
第一行 4 个数字 n，s，K，T
第二行 n 个数字，代表第 i 个城市有多少工作人员
接下来 m 行每行两个数字 x 和 y，代表连接 x 和 y 的单向道路



6 1 30 2
20 10 10 5 5 5
1 2
1 3
2 4
2 5
3 6

问题 1 答案 15
问题 2 答案

最好情况：3 和 4 受到疫情影响，结果为 12
最坏情况：2 和 6 受到疫情影响，结果为 7

数据范围 n<=10^4, T <= 10, K <= 10^6
每一问正确解法的最坏时间复杂度均小于等于 10^6

文字解答

第一问 设 c_i 为第 i 个城市的工作人员数目，则其实际最大每日运输量为

$$d_i = \begin{cases} c_i & , \quad i \text{ 为叶节点} \\ \min(c_i, \sum_{j \in \mathrm{Child}(i)} d_j) & , \quad i \text{ 非叶节点} \end{cases} \tag{1}$$

自底向上计算 d_i 即可. 最后能运送的最大防疫物资数目为 $\min(d_1, K)$.

第二问 ($T = 1$) 先计算 $e_i = \lfloor c_i \rfloor$, 即第 i 个城市受影响后的每日最大运输量. 假设它受影响，则总运输量减少恰为 $\delta_i = \max(d_i - e_i, 0)$. 对于最好最坏情况，选使得 δ_i 最小/最大的 i 即可.

第二问 ($T = 2$) 思路：动态规划. 设 $F(i, T)$ 为以 i 为根节点的子树中有 T 节点个受到影响后的最大运输能力. 则有以下情况 ($\mathrm{lc}(i)$ 和 $\mathrm{lc}(i)$ 代表 i 的两个孩子 (如果存在的话)):

1. i 受到影响，左子树有 t 个节点受影响，右子树有 $T-1-t$ 个节点受影响. 此时 i 为根的子树最大运输量为 $\min(e_i, F(\mathrm{lc}(i), t) + F(\mathrm{rc}(i), T-1-t))$. 其中 t 的可能取值为 $0, \cdots, T-1$.
2. i 不受影响，左子树有 t 个节点受影响，右子树有 $T-t$ 个节点受影响. 此时 i 为根的子树最大运输量为 $\min(c_i, F(\mathrm{lc}(i), t) + F(\mathrm{rc}(i), T-t))$. 其中 t 的可能取值为 $0, \cdots, T$.

若要最好情况，则 $F(i, T)$ 取上述所有情况中最大值即可，最坏情况取最小值即可.

注意边界情况:

1. $T = 0$: 同第一问;
2. $T = 1$: 同第二问 $T = 1$ 的情况;
3. T 应当小于等于 i 为根的子树的节点数，上述情况讨论中 t 的取值应当考虑到这一点.

伪代码解答
第一问伪代码，时间复杂度 O(n)

```
void dfs(TreeNode* x){
    if(x == NULL) return;
    dfs(x->left);
    dfs(x->right);
    int left_n = 0
    int right_n = 0;
    if(x->left != NULL)
        left_n = x->left->max_num;
    if(x->right != NULL)
        right_n = x->right->max_num;
    x->max_num = x->ci;
    if(x->left == NULL && x->right == NULL){
```

```
        return;
    }
    x->max_num = min(x->max_num, left_n + right_n);
}

void build_tree(){
    root = new TreeNode(s, cap[s]);
    for(auto edge : edges){
        int x = edge.from, y = edge.to;
        if(x->left == NULL){
            x->left = new TreeNode(y, cap[y]);
        } else {
            x->right = new TreeNode(y, cap[y]);
        }
    }
}

int main(){
    read_data();
    build_tree();
    dfs(root);
    cout<<s->max_num<<endl;
}
```

第二问伪代码，时间复杂度 O(nT^2)

```
// 以最好情况为例子，最坏情况把 mymax 变成 mymin 即可
void dfs_mymax(TreeNode* x){
    if(x == NULL) return;
    dfs_mymax(x->left);
    dfs_mymax(x->right);
    int left_n = 0
    int right_n = 0;
    x->max_num[0] = x->ci;
    x->max_num[1] = x->ci / 2;
    for(int t = 2; t <= T; t++) x->max_num[t] = -INT_MAX; // 求最坏情况为正无穷
    if(x->left == NULL && x->right == NULL){
        return;
    }

    for(int t = 0; t <= T; t++){
        max_num_child[t] = -INT_MAX;
        for(int t2 = 0; t2 <= t; t2++){
```

```
            int left_n = 0, right_n = 0;
            if(x->left != NULL)
                left_n = x->left->max_num[t2];
            if(x->right != NULL)
                right_n = x->right->max_num[t-t2];
            max_num_child[t] = mymax(max_num_child[t], left_n + 
right_n);
        }
    }
    x->max_num[0] = min(x->max_num[0], max_num_child[0]);
    for(int t = 1; t <= T; t++){
        int self0 = min(x->max_num[0], max_num_child[t]);
        int self1 = min(x->ci / 2, max_num_child[t-1]);
        x->max_num[t] = mymax(self0, self1);
    }
}
```
最后答案为 root->max_num[T]