暨南大學
JINAN UNIVERSITY

2020 年全国硕士研究生统一入学考试自命题试题 B 卷

****

学科、专业名称：网络空间安全

研究方向：网络空间安全 083900

考试科目名称及代码：数据结构 830

考生注意：所有答案必须写在答题纸（卷）上，写在本试题上一律不给分。

一、 单项选择题(每题 2 分，共 30 分)

1. 下述关于顺序存储结构优点的说法，哪个是正确的（ **C** ）

   A. 插入运算方便　　B. 可方便地用于各种逻辑结构的存储表示

   C. 存储密度大　　D. 删除运算方便

2. 假设根结点为第 1 层，深度为 h 层的二叉树至少有(**B**) 个结点(h>1)；

   A. 2^h　　B. 2^{h-1}　　C. 2^h+1　　D. 2^h-1

3. 用单向链表来实现容量为 n 的堆栈时，链表头指针指向堆栈顶部元素，链表尾指针指向堆栈底部元素，则以下说法错误的是(**C**)

   A. 入栈操作的复杂度为 O(1)　　B. 出栈操作的复杂度为 O(1)

   C. 删除底部元素的复杂度为 O(1)　　D. 插入一个新的堆栈底部元素复杂度为 O(1)

4. 以下关于递归算法的论述，不正确的是(**B**)

   A. 递归算法的代码可读性好　　B. 递归算法可以提高程序运行效率

   C. 递归调用层次太深有可能造成堆栈溢出　　D. 递归调用层次太深会占用大量内存

5. 设有字符集合{4,6,3,W,S}，将字符序列 6W43S 中的字符按顺序进入堆栈，出栈可发生在任何时刻。则以下的出栈序列错误的是（ **D** ）。

   A. 64WS3　　B. 4W36S　　C. 6W34S　　D. WS436

6. 在管理城市道路交通网络据时，最适合采用（ **A** ）数据结构来对其进行存储。

   A．有向图　　B．无向图　　C．树　　D．矩阵

7. 具有 k 个顶点的完全有向图的边数为(**A**)。

   A. k(k-1)　　B. k(k-1)/2　　C. k^2-1　　D. k^2+1

8. 若线性表最常用的操作是增加或者删除某个元素, 则采用(**B**)存储方式节省时间.

   A. 单链表　　B. 双链表　　C. 单循环链表　　D. 顺序表

9. 由权为 6,3,2,8 的四个叶子结点构造一个哈夫曼树，该树的带权路径长度为（ **B** ）。

   A. 36　　B. 35　　C. 34　　D. 33

10. 为了提高哈希表的查找效率，以下方法说法不正确的是(**B**)。

    A. 设计好的哈希函数　　B. 增加哈希函数的个数

    C. 增大存储空间　　D. 采用更好的地址冲突解决方法

11. 以下数据结构中哪一个是非线性结构？(**D**)

    A. 队列　　B. 栈　　C. 线性表　　D. 二叉树

12. 对于一个整数集合{11，37，29，55，80，46，73，17}进行散列存储时，若选用函数 H（K）＝K %9 作为散列(哈希)函数，则散列地址为 1 的元素有（ **B** ）个。

    A．3　　B．4　　C．5　　D．6

13. 有一个 100*90 的整数稀疏矩阵，其中非 0 元素个数为 10；设每个整数占用 3 个字节，则用三元组表示该矩阵时，总共需要的存储空间为(**C**)字节。

A．30　　B．33　　C．90　　D．99

14. 在一个双向链表中，当删除结点 p 时，错误的操作序列为 (**A**)。

A. p=p->prev; p->next->prev=p; p->next=p->next->next;

B. p=p->next; p->prev=p->prev->prev; p->prev->next=p;

C. p->prev->next=p->next; p->next->prev=p->prev;

D. p=p->prev; p->next=p->next->next; p->next->prev=p;

15. 在一个具有 V 个顶点的有向连通图中，若所有顶点的入度数之和为 N，所有顶点的出度之和为 M，则以下说法正确的是（ **C** ）。

A．V=(M+N)/2　　B．M>V　　C．M=N　　D．N>V

二、填空题(每空 2 分，共 20 分)

1. 对 n 个不同的排序码进行冒泡排序，在元素无序的情况下比较的次数为 **n(n-1)/2** 。

2. 在单链表中，要将 m 所指结点插入到 n 所指结点之后，其语句表示为 **m->next=n->next;n->next=m** 。

3. 设有数组 A[i][j]，数组的每个元素长度为 3 字节，i 的值为 1 到 8，j 的值为 1 到 10，数组从内存首地址 BA 开始顺序存放，当用以列为主存放时，元素 A[5][8]的存储首地址为 **BA+142** 。

4. 设哈夫曼树中有 199 个结点，则该哈夫曼树中有 100 个叶子结点。

5. 对 22 个记录的有序表作折半查找，当查找失败时候，至多需要比较 5 次关键字，至少需要比较 4 次关键字。

6. 由 3 个结点可以构造出 5 种不同的二叉树。

7. 最大容量为 s 的循环队列，队尾指针是 rear，队头是 front，则队满的条件是 front =(rear+1)%s 。

8. G 是一个非连通无向图，共有 28 条边，则该图至少有 9 个顶点。

9. 数据表中有 10000 个元素，如果仅要求求出其中最大的 10 个元素，则采用 **堆** 排序算法最节省时间。

三．判断题（每题 1 分，共 10 分，正确的选 T，错误的选 F）

1. 对 n 个记录进行插入排序，最多只需要 O(nlog(n))次两两比较。(**错**)

2. 用链表(Linked list)来实现的堆栈，单次入栈/出栈操作的时间复杂度可达到 O(1)。(**对**)

3. 当图 G 中存在权重为负数的边时，Dijkstra 算法不能用于计算 G 中两结点间最短路径。(**对**)

4. 贪婪算法有时候能够求出全局最优解，有时候无法求出全局最优解。(**对**)

5. 对含有 n 个记录的集合进行快速排序，在最坏情况下的时间复杂度是 O(nlog(n))。(**错**)

6. 哈希表查找效率高，当查找主键在范围[a,b]内所有的记录时，也应该优先选择哈希表。(**错**)

7. 无向图的邻接矩阵是对称的，因此可只存储矩阵的下三角阵以节省存储空间。 (**对**)

8. 用 Prime 算法和 Kruskal 算法求得的图的最小生成树一定相同。(**错**)

9. 在一个有向图的邻接表或逆邻接表中，若某顶点的链表为空，则该顶点的度为零。(**错**)

10. 在有 n 个顶点的无向图中，若边数>n-1，则该图必是连通图。(**错**)

四、简答题（40 分）

1. 简述逻辑结构的四种基本关系并画出它们的关系图。（10 分）

2. 设二维数组 num[1….m, 1…n]含有 m*n 个整数，请分析判断数组中元素是否互不相同的算法的时间复杂度。（8 分）

算法分析：维护一个初始节点个数为 0 的大根堆，对二维数组进行枚举。将每次枚举到的元素在大根堆中进行查找，若查找成功则表示数组中存在相同元素，若查找失败则将该元素插入大根堆中。该算法所需要使用的时间复杂度为 O(nmlog(nm))

3. 设待排序的关键字序列为{12，2，16，30，28，10，16*，20，6，18} ，请写出二路归并排序的方法下，每趟排序结束后关键字序列的状态。（6 分）

第一趟：2，12，16，30，10，28，16，20，6，18
第二趟：2，12，16，30，10，16，20，28，6，18
第三趟：2，10，12，16，16，20，28，30，6，18
第四趟：2，6，10，12，16，16，18，20，28，30

4. 已知图 1 所示的有向图，请给出（1）每个顶点的入度与出度；（2）邻接矩阵。（10 分）

(1)顶点 1 无出度，入度：2，5，6
顶点 2 出度：1，4，入度：3，6
顶点 3 出度：2，6，入度：4
顶点 4 出度：3，5，6，入度：2
顶点 5 出度：1，入度：4，6
顶点 6 出度：1，2，5，入度：3，4

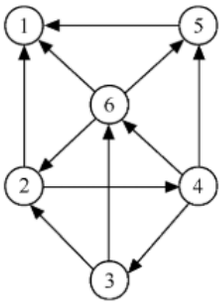
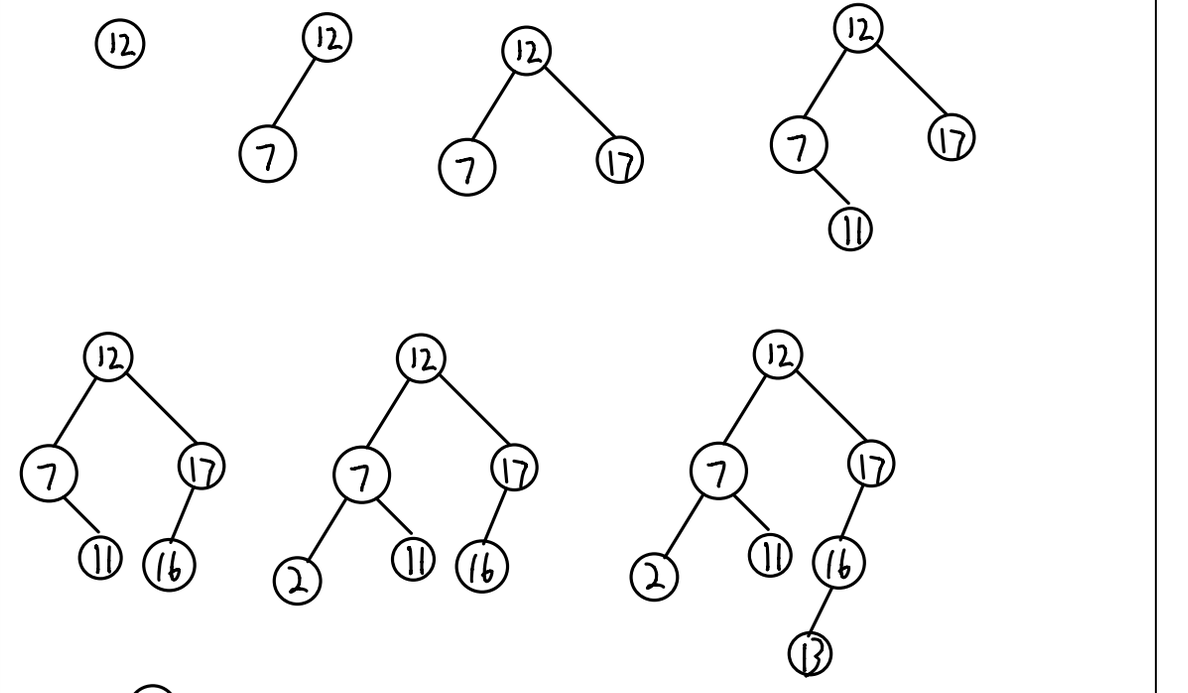


图 1. 有向图

(2)

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

5. 在一棵空的二叉排列树中依次插入关键字序列为 12，7，17，11，16，2，13，9，21，4，请画出所得到的二叉排列树。（6 分）



五、算法填空（共 2 小题，每空 2 分，共 20 分）

1. 在汉诺塔(hanoi tower)游戏中，总共有 3 根柱子和 n 个大小不一样的盘子。初始状态时 n 个盘子从小到大堆叠在 1 号柱子，下面的递归算法伪代码能够将这 n 个盘子从 1 号柱子移动到 3 号柱子。其中，该递归算法满足以下条件：(1)每次只移动 1 个盘子，(2)任何一个盘子只有当它上面没有堆放盘子时才能移动，(3)任何时刻在任何一个柱子上永远不能出现大盘子堆在小盘子之上的情况。请在_____处填上适当内容，使其成为一个完整的算法。

```
hanoi(from, tmp, to, n)
{    if(n==1){
          move(  (1)from  ,  (2)to  );
          return;
      }
      hanoi( (3)from, to,tmp,n-1      );
      printf ( "(%d,%d)", from, to );
      hanoi( (4)tmp,from,to,n-1      );
      return;
}
```

2. 假设一维数组 A 保存有 N 个整数，以下快速排序算法代码能够对数组 A 进行排序。请在 _____处填上适当内容，使其成为一个完整的算法。

```
int partition(int* A, int N, int p, int r)
{       int x = A[r];
        int i =   (5)p-1    ;
         for (int j = p; j<=r-1; j++){
             if (   (6)A[j]>=x    ){
                  i = i + 1;
                  int temp = A[i];
               A[i] = A[j];
               A[j] = temp;
            }
         }
         int temp = A[i+1];
          A[i+1] = A[r];
          A[r] = temp;
         return  (7)i+1  ;
}
void QuickSort(int* A, int N, int p, int r)
{
      int q;
      if (   (8)p<r   ){
          q = partition(A, N, p, r);
          QuickSort(     (9)A,N,p,q-1      );
          QuickSort(     (10)A,N,p+1,r    );
       }
       return;
}
```

```
void main()
{       QuickSort(A, N, 0,N-1);
        return 0;
}
```

六．编写算法（30 分）

1. 写一个算法统计在输入字符串中各个不同字符出现的频度并将结果存入文件（字符串中的合法字符为 A-Z 这 26 个字母与 0-9 这 10 个数字）。（10 分）

```cpp
#include<iostream>
#include<cstdio>
#include<cstring>
#include<fstream>
using namespace std;
const int maxn = 36;
int cnt[maxn];
void solve(char *s){
    int len = strlen(s);
    for(int i = 0; i < len; i++){
        if('0' <= s[i] && s[i] <= '9'){
            cnt[s[i]-'0'] += 1;
        }
        else if('A' <= s[i] && s[i] <= 'Z'){
            cnt[s[i]-'A'+10] +=1;
        }
    }
    ofstream ofile;
    ofile.open("./myfile.txt");
    for(int i = 0; i < 10; i++){
        ofile << char(i+'0') << ":" << cnt[i] << endl;
    }
    for(int i = 10; i < 36; i++){
        ofile << char(i+'A'-10) << ":" << cnt[i] << endl;
    }
    ofile.close();
}
int main(){
    char s[] = "IVONDSVNOEIBNNCDWIOPGIENCAOINIWOEBGWJBVOEWOF1805Y0392U50HF130HH280F02CH0G2HV0J2H08GVH02EJ0";
    solve(s);
    return 0;
}
```

2. 已知 f 为单链表的表头指针，链表中存储的都是整型数据，请写出实现下列运算的递归算法，求（1）链表中最大整数；（2）所有整数的平均值。（10 分）

```
#include<iostream>
#include<cstdio>
using namespace std;
const int inf = 0x3f3f3f3f;
struct node{
    int data;
    struct node *next;
};
struct node *m = new node();

int getMax(struct node *p){
    if(p->next==NULL) return p->data;
    else return max(getMax(p->next), p->data);
}

int solve1(struct node *m){
    if(m->next==NULL) return -inf; //-inf表示链表为空
    else return getMax(m->next);
}

int cnt = 0;

int getSum(struct node *p){
    if(p==NULL) return 0;
    cnt+=1;
    return getSum(p->next)+p->data;
}

double solve2(struct node *m){
    if(m->next==NULL) return -inf; //-inf表示链表为空
    else return 1.0*getSum(m->next)/cnt;
}
```

3. 采用邻接表存储结构，编写一个算法，判别无向图中任意给定的两个顶点之间是否存在一条长度为 k 的简单路径。（10 分）

```cpp
#include<iostream>
#include<cstdio>
#include<vector>
#include<cstring>
using namespace std;
const int maxn = 1e5+10;
vector<int> vec[maxn];
bool vis[maxn], flag;
void init(){
    memset(vis, false, sizeof(vis));
    flag = false;
}
void solve(int s, int e, int n, int k){
    if(n > k) return;
    else if(flag) return;
    else if(s == e && n != k) return;
    else if(s == e && n == k){
        flag = true;
        return;
    }
    int len = vec[s].size();
    for(int i = 0; i < len; i++){
        if(vis[vec[s][i]]) continue;
        vis[vec[s][i]] = true;
        solve(vec[s][i], e, n+1, k);
        vis[vec[s][i]] = false;
    }
    return;
}


int main(){
    int n, m, k, s, e;
    cin >> n >> m >> k;
    for(int i = 0; i < m; i++){
        int u, v;
        cin >> u >> v;
        vec[u].push_back(v);
        vec[v].push_back(u);
    }

    while(cin >> s >> e){
        init();
        vis[s] = true;
        solve(s, e, 0, k);
        vis[s] = false;
        if(flag) cout << "yes!" << endl;
        else cout << "no!" << endl;
    }
    return 0;
}
```