清华大学本科生考试试题专用纸

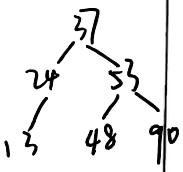
考试课程 数据结构 （ A 卷） 2018 年 1 月 17 日

姓名 陈栗凡 班级 自74 学号 2017011621

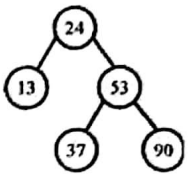
一、填空题（45 分，每题 3 分）

1、 已知一棵有 2018 个节点的树，其叶子节点的个数为 115，该树对应的二叉树中无右孩子的节点个数为 1904 。 中间结点个数：2018-115=1903，加上根结点=1904

2、 设高度为 h（单个顶点的二叉树认为高度为 1，空树的高度认为是 0）的二叉树上只有度为 0 和度为 2 的结点，则此类二叉树中所包含的结点数至少为 2h-1 。



3、 在右图所示的平衡二叉树中，插入关键字 48 后得到一棵新平衡二叉树。在新平衡二叉树中，关键字 37 所在节点的左、右子节点中保存的关键字分别是 24,53 。

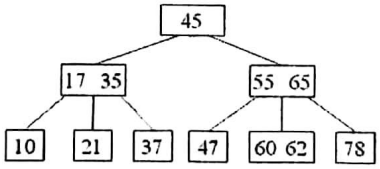


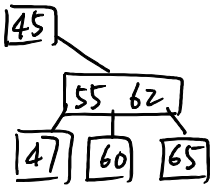
4、 设哈希表长 m=14，哈希函数为 H(key)=key MOD 11。表中已有 4 个节点 H(15)=4，H(38)=5，H(61)=6，H(84)=7，其余地址为空。如用线性探测法处理冲突，则关键字为 49 的节点地址是 8 。

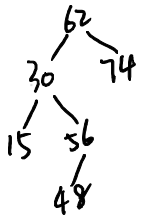
5. 6个顶点完全图边数：$C_6^2=15$ 加一个顶点连通 15+1

5、 若无向图中含 7 个顶点，则保证图 G 在任何情况下都是连通的，则需要的边数最少是 16 。

6、 如下图所示的三阶 B-树，删除关键字 78 得到一棵新 B-树，其最右叶节点中所含的关键字是 65 。

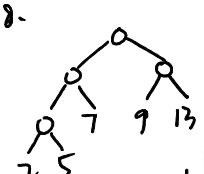




7. 

7、 按照序列（62，30，74，15，56，48）构造的二叉排序树，在等概率情况下，其查找成功的平均查找长度是 2.5 。

8、 以数据集（2，5，7，9，13）为权值构造哈夫曼树，其带权路径长度是 79 。

8. 

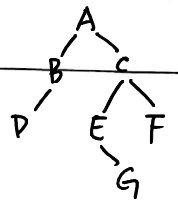
(7+9+13)×2+(2+5)×3

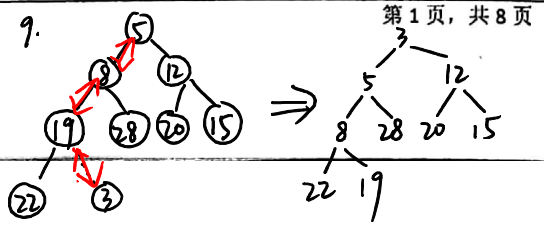
9、 已知关键字序列 5，8，12，19，28，20，15，22 是小根堆（最小堆），插入关键字 3，调整好后得到的小根堆是 3,5,12,8,28,20,15,22,19 。

10、 字符串 ababaababcabb 的 next 数组是 011234234512 3 。

a b a b a a b a b c a b b
next 0 1 1 2 3 4 2 3 4 5 1 2 3
nextval 0 1 0 1 0 4 1 0 1 5 0 1 3

11、 已知二叉树中序遍历结果为 DBAEGCF，后序遍历结果是 DBGEFCA，则其前序遍历结果是 ABDCEGF 。

11. 

9. 

(注:后缀表达式中不含括号)

12、 表达式((A+B)*C*D-E+F)/G 的后缀表达式为 AB+C*D*E-F+G/。

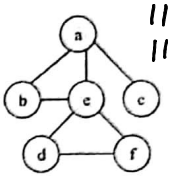
13、 对关键字序列（51，28，39，80，70，96，10，35，40）进行 2 趟希尔排序的结果是：10,28,39,35,40,80,51,96,70。（希尔排序增量序列为4，2，1）

第1次：40,28,10,35,51,96,39,80,70

14、 对给定的关键字序列 110，119，007，911，114，120，122 进行基数排序，则第 2 趟分配收集后得到的关键字序列是 007,110,911,114,119,120,122。

第1次：
110,120,911,122,
114,007,119

15、 如右图所示，在 aebdfc、acfdeb、eacdfb、aedfcb、aefdcb、aefdbc、edfcab 7 个序列中，符合深度优先遍历的序列有 2 个。



二、分析题（25 分，每题 5 分）

1、试分析函数 ABC 中常数 C1、C2 有何功能。

```
typedef struct node
{int data ; struct node *lchild, *rchild; } btnode;
void ABC(btnode *BT, int &c1, int &c2)
{
   if (BT != NULL ){
     ABC(BT->lchild, c1, c2);
     c1++;
     if(BT->lchild ==NULL && BT->rchild ==NULL)
       c2++;
     ABC(BT->rchild, c1, c2);
   }
}
```

结果请写在以下空格内：

C1: 统计树中总结点个数

C2: 统计树中叶结点个数

2、将二叉树 bt 中每一个结点的左右子树互换的 C 语言算法如下，其中 ADDQ(Q, bt)，DELQ(Q)，EMPTY(Q)分别为进队、出队和判别队列是否为空的函数，请补充算法，完成其功能。

```
typedef struct node
{int data ; struct node *lchild, *rchild; } btnode;
void  EXCHANGE(btnode *bt)
{
  btnode *p, *q;
  if (bt)
  {
     ADDQ(Q, bt);
     while(!EMPTY(Q))
       {
```

```
            p=DELQ(Q);
            q=___(1)___;
            p->rchild=___(2)___;
            ___(3)___=q;
            if(p->lchild)___(4)___;
            if(p->rchild)___(5)___;
        }
    }
}
```

结果请写在以下空格内：

(1)_____

(2)_____

(3)_____

(4)_____

(5)_____

3、请补充二叉搜索树中删除结点的代码（结点值为 x，删除后需保持二叉搜索树性质）

```
typedef struct node
{int data ; struct node *lchild, *rchild; } btnode;
bool Remove(int x, btnode *&p)
{
   if (___(1)___)
   {
      if (x < p->data) Remove(x, p->lchild);
      else if (x > p->data) Remove(x, p->rchild);
      else if (___(2)___)
      {
         Btnode *temp = p;
         if (p->lchild == NULL) p = p->rchild;
         else p = p->lchild;
         delete temp;
      }
      else
      {
          Btnode *temp = p->rchild;
          while (temp->lchild!=NULL)___(3)___;
          ___(4)___;
          ___(5)___;
      }
      return true;
   }
   return false;
}
```

结果请写在以下空格内：

(1)_____

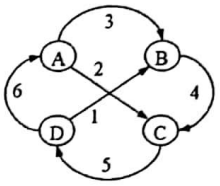
(2)_____

(3)_____

(4)_____

(5)_____

4、已知带权有向图如右图所示，请用 Floyd 算法计算该图中每两点间的最短路径及长度。



(1) 写出计算过程（依次计算 $A^{(0)}$，$A^{(1)}$，...，$A^{(4)}$。$A^{(0)}$为邻接矩阵，顶点次序为 ABCD，矩阵方格中为两点间当前的最短路径长度及路径上的顶点）。

(2) 补充 Floyd 算法。

（1）计算过程

$A^{(0)}$：

0	3 AB	2 AC	∞
∞	0	4 BC	∞
∞	∞	0	5 CD
6 DA	1 DB	∞	0

$A^{(1)}$：

0	3 AB	2 AC	∞
∞	0	4 BC	∞
∞	∞	0	5 CD
6 DA	1 DB	8 DAC	0

$A^{(2)}$：

0	3 AB	2 AC	∞
∞	0	4 BC	∞
∞	∞	0	5 CD
6 DA	1 DB	5 DBC	0

$A^{(3)}$：

0	3 AB	2 AC	7 ACD
∞	0	4 BC	9 BCD
∞	∞	0	5 CD
6 DA	1 DB	5 DBC	0

$A^{(4)}$：

0	3 AB	2 AC	7 ACD
15 BCDA	0	4 BC	9 BCD
11 CDA	6 CDB	0	5 CD
6 DA	1 DB	5 DBC	0

（2）补充 Floyd 算法。

```
typedef struct
 {
   int adj;
   char *info; /* 该弧相关信息的指针(可无) */
 }ArcCell,AdjMatrix[MAX_VERTEX_NUM][MAX_VERTEX_NUM];
 typedef struct
 {
   char vexs[MAX_VERTEX_NUM]; /* 顶点向量 */
   AdjMatrix arcs; /* 邻接矩阵 */
   int vexnum,arcnum; /* 图的当前顶点数和弧数 */
   GraphKind kind; /* 图的种类标志 */
 }MGraph;
typedef int PathMatrix[MAX_VERTEX_NUM][MAX_VERTEX_NUM][MAX_VERTEX_NUM];
typedef int DistancMatrix[MAX_VERTEX_NUM][MAX_VERTEX_NUM];
void ShortestPath_FLOYD(MGraph G,PathMatrix &P[],DistancMatrix &D)
 { /* 用 Floyd 算法求有向网 G 中各对顶点 v 和 w 之间的最短路径 P[v][w]及其 */
   /* 带权长度 D[v][w]。若 P[v][w][u]为 TRUE,则 u 是从 v 到 w 当前求得最短 */
   /* 路径上的顶点。*/
   int u,v,w,i;
   for(v=0;v<G.vexnum;v++) /* 各对结点之间初始已知路径及距离 */
     for(w=0;w<G.vexnum;w++)
     {
       ____(1)____;
       for(u=0;u<G.vexnum;u++)
         ____(2)____;
       if(D[v][w]<INFINITY) /* 从 v 到 w 有直接路径 */
       {
         P[v][w][v]=TRUE;
         P[v][w][w]=TRUE;
       }
     }
   for(u=0;u<G.vexnum;u++)
     for(v=0;v<G.vexnum;v++)
       for(w=0;w<G.vexnum;w++)
         if(____(3)____) /* 从 v 经 u 到 w 的一条路径更短 */
         {
           ____(4)____;
           for(i=0;i<G.vexnum;i++)
             ____(5)____;
         }
 }
```

(1) D[v][w]:= G.arcs[v][w].adj //D矩阵初始化

(2) P[v][w][u]=FALSE //P矩阵初始化.

(3) D[v][w] > D[v][u]+D[u][w]

(4) D[v][w]=D[v][u]+D[u][w]

(5) P[v][w][i]:= P[v][u][i]||P[u][w][i]

//将沿途经过的路径置为TRUE.

结果请写在以下空格内：

(1)_____

(2)_____

(3)_____

(4)_____

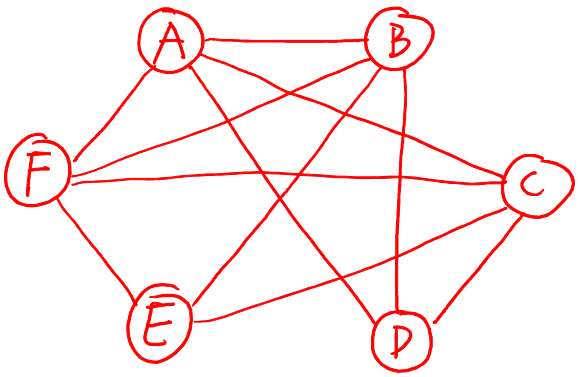
(5)_____

5、某学院 10 名博士生(B1～B10)选修 6 门课程(A～F)的情况如下表(用 √ 表示选修)所示。

	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10
A	√	√	√		√				√	√
B	√			√				√	√	
C		√			√	√	√			√
D	√				√			√		
E				√		√	√			
F			√	√			√		√	√

现需要安排这 6 门课程的考试，要求是：(1) 每天上、下午各安排一门课程考试，计划连续 3 天考完；(2) 每个博士生每天只能参加一门课程考试，在这 3 天内考完全部选修课；(3) 在遵循上述两条的基础上，各课程的考试时间应尽量按字母升序做先后顺序安排(字母升序意味着课程难度逐步增加)。 请为该学院安排各门课程的考试顺序（需采用图论方法）。

将不能在同一天考试的课程相连：



故考试安排为 AE. BC. DF.

三、算法设计题（30 分，每题 15 分）

1、农夫约翰为了修理栅栏，要将一块很长的木板切割成 N 块。准备切成的木板的长度为 L_1、L_2、...、L_N，未切割前木板的长度恰好为切割后木板长度的总和。每次切断木板时，需要的开销为这块木板的长度。例如长度为 21 的木板要切成长度为 5、8、8 的三块木板。长 21 的木板切成长为 13 和 8 的板时，开销为 21。再将长度为 13 的板切成长度为 5 和 8 的板时，开销是 13。于是合计开销是 34。请求出按照目标要求将木板切割完最小的开销是多少。

限制条件

1：1<=N<=20000

2：0<=Li<=50000

样例输入（对应题目中的例子）

3

8 5 8

样例输出

34

```c
#include"stdio.h"
#include"limits.h"
#include"malloc.h"
 //找出L[]数组中最小的元素并返回地址
int min(int N,int L[])
{
    int loc,min=INT_MAX;
    for(int i=1;i<=N;i++)
        if(L[i]<min)
        {min=L[i];loc=i;}
    return loc;
}
 //采用Huffman树的形式解决
int cut(int N,int L[])//L为题目中给出的L[1]-L[N]
{
    int total=0;
    for(int i=1;i<N;i++)
    {
        int num1,num2;
        int min1=min(N,L),len1=L[min1];
        L[min1]=INT_MAX;
        int min2=min(N,L),len2=L[min2];
        L[min2]=len1+len2;
        total+=len1+len2;
    }
    return total;
}
int main()
{
    int N;
    scanf("%d",&N);
    int *L=(int *)malloc(sizeof(int)*(N+1));
    for(int i=1;i<=N;i++)
        scanf("%d",&L[i]);
    printf("%d\n",cut(N,L));
}
```

2、给定一个具有 n 个顶点的无向图。要给图上每个顶点染色，并且要使相邻的顶点颜色不同。判定该图是否能最多用 2 种颜色进行染色？题目保证没有重边和自环。

限制条件：1<=n<=1000

输入：

第一行为顶点数 n，边数 m

后面各行为每条边起点和终点的顶点序号 i,j

样例输入：

3 3

0 1

0 2

1 2

样例输出：

NO

```
#include"stdio.h"
#include"malloc.h"
 #define MAX_VEX_NUM 999
struct ArcNode  //采用图的邻接表来储存
{
    int adjvex;
    ArcNode *next;
};
struct Graph
{
    int vexnum,edgenum;
    ArcNode *vexs[MAX_VEX_NUM];
};
bool dfs(Graph G,int *color,int i,int c)
{
    //如果新上色与已有颜色冲突
    if(color[i]+c==0)return false;
    //如果该节点已被上色
    if(color[i]!=0)return true;
    color[i]=c;
    for(ArcNode *p=G.vexs[i];p!=NULL;p=p->next)
        if(!dfs(G,color,p->adjvex,-c)) return false;
    return true;
}
bool judge(Graph G)
{
    //color=0未上色，color=1和-1代表两种颜色
    int *color=(int*)malloc(sizeof(int)*G.vexnum);
    for(int i=0;i<G.vexnum;i++) color[i]=0;
    for(int i=0;i<G.vexnum;i++)
        if(color[i]==0)
            if(!dfs(G,color,i,1)) return false;
    return true;
}
int main()
{
    Graph G;
    int i;
    int vex1,vex2;
    scanf("%d%d",&G.vexnum,&G.edgenum);
    for(i=0;i<G.vexnum;i++)
        G.vexs[i]=NULL;
    for(i=0;i<G.edgenum;i++)//图的储存
    {
        scanf("%d%d",&vex1,&vex2);
        ArcNode *arc1=(ArcNode*)malloc(sizeof(ArcNode));
        ArcNode *arc2=(ArcNode*)malloc(sizeof(ArcNode));
        arc1->adjvex=vex1;arc2->adjvex=vex2;
        arc1->next=arc2->next=NULL;
        ArcNode *p=G.vexs[vex1];
        while(p&&p->next) p=p->next;
        if(!p) G.vexs[vex1]=arc2;
        else p->next=arc2;
        p=G.vexs[vex2];
        while(p&&p->next) p=p->next;
        if(!p) G.vexs[vex2]=arc1;
        else p->next=arc1;
    }
    if(judge(G))printf("TRUE");
    else printf("FALSE");
}
```