提高组模拟题第四套试题及答案

1．在以下各项中，（　D　）不是 CPU 的组成部分。

A.控制器　　B.运算器　　C.寄存器　　D.主板

解析：控制器、运算器、寄存器均为 CPU 组成部分。

2．（2017）8＋（1234）10 的结果是（　B　）。

A.(8E2)16　　B.(100011100001)2

C.(8E0)16　　D.(100011100011)2

解析：

2017 转十进制 2*8^3+1*8^1+7*8^0=1024+8+7=1039

1039+1234=2273

8E2=8*16^2+14*16^1+2*16^0=2048+224+32=2304

8E0=8*16^2+14*16^1=2048+224=2272

100011100001=1*2^11+1*2^7+1*2^6+1*2^5+1*2^0=2048+128+64+32+1=2273

所以选 B

3．设 A=B=True,C=D=False，逻辑运算表达式值为假的是（　D　）。

A.（¬A∧B)∨(C∧D∨A)　　B. ¬(((A∧B)∨C)∧D)

C.A∧(B∨C∨D)∨D　　D.(A∧(D∨C))∧B

解析：逻辑运算符，非(¬)，与(∧)，或(∨)，运算后只有 D 的值为假。

4．若已知一个栈的入栈序列是 1,2,3，・・・,其输出序列为 p1,p2,p3,・・・,pn，若 p1=n，则 pi 为（　C　）。

A.i　　B.n-i　　C.n-i+1　　D.不确定

解析：n 最后入栈，最先出栈，故出栈序为 n,n-1,・・・,1,出栈顺序和入栈顺序是反的，所以出栈的第 i 个元素就是从 n 开始倒着数的第 i 个元素。1~n 编号，逆序就是 n-i+1。

5．设 S=“abbcce”，不同的非空子串个数有（　A　）个。

A.19　　B.17　　C.16　　D.18

解析：

非空子串数量 n(n+1)/2,即 6*7/2=21。有两个相同字符 b 和 c 再-2=19。

6．C＋＋中，125^2 的值是（　A　）。

A.127　　B.128　　C.15625　　D.126

解析：

^按位异或，0^0=0 0^1=1 1^0=1 1^1=0

125=62*2……1

62=31*2……0

31=15*2……1

15=7*2……1

7=3*2……1

3=1*2……1

1=0*2……1
1111101
0000010
1111111=1*2^6+1*2^5+1*2^$+1*2^3+1*2^2+1*2^1+1*2^0=64+32+16+8+4+2+1=127
选 A

7．设某算法的计算时间表示为递推关系式 T(n)=2T(n/2)+n(n 为正整数）及 T(0)=1，则该算法的时间复杂度为（　C　　）。

A.O(n)　　B.O(n2)　　C.(nlogn)　　D.(logn)

解析：

主定理：T(n)=2T(n/2)+O(n)可得到 T(n)=O(nlogn)。

8．下列有关二叉树的叙述，不正确的是（　　C　）

A.二叉树的深度为 k，那么最多有 2k-1 个节点　　(k>=1)

B.在二叉树的第 i 层上，最多有 2i-1 个节点 (i>=1)

C.完全二叉树一定是满二叉树

D.堆是完全二叉树

解析：

完全二叉树是设点数为 n，让根的编号为 1，每个点 i 的左右儿子标号分别为 2i 和 2i＋1，之后所有点的编号形成一个 1 到 n 的排列的树；满二叉树是一种特殊的完全二叉树，其 n 符合 2^k-1 的形式。

9．下列排序算法中，平均时间复杂度是 O(n2)的是（　　B　　）。

A.快速排序　　B.插入排序　　C.归并排序　　D.堆排序

解析：

插入排序是 O(n^2).其他是 O(nlogn)。

10．下列哪些图一定可以进行黑白染色，使得相邻节点的颜色不同（　A　　）。

A.树　　B.基环树　　C.连通图　　D.欧拉图

解析：

把树的深度为奇数的点染成黑色，深度为偶数的点染成白色即可。

11．下列不是操作系统的有（　　D　）。

A.Linux　　B.Windows　　C.Android　　D.Wps

解析：WPS 是办公软件。

12．下列关于算法的叙述中，错误的是（　B　　）。

A.算法代表着用系统的方法描绘解决问题的策略机制。

B.一个算法的好坏，只需要从时间复杂度这一方面进行考虑

C.一个算法必须具有有穷性、可行性、正确性、输入和输出

D.对于一些 np 完全问题，现在未能找到有效的算法解决

解析：

一个算法的好坏，需要从时空复杂度和正确性进行考虑。

13．下列叙述中正确的是（　A　）。

A.线性表是线性结构　　B.栈与队列是非线性结构

C.线性链表是非线性结构　　D.二叉树是线性结构

解析：

栈、队列是线性结构，树是非线性结构。

14．链表的（　B　）操作需要 O（n）的时间复杂度实现。

A.插入　　B.定位　　C.删除　　D.合并

解析：

链表的定位是 O(n)，插入、删除、合并是 O(1)。

15．NOI 比赛，下列选项中不可以带入考场的是（　C　）。

A.键盘　　B.空白纸张　　C.U 盘　　D.铅笔

解析：

u 盘不能带入。

阅读程序

```
1   #include <iostream>
2   using namespace std;
3   int n,k,ans;
4   int main()
5   {
6       cin>>n>>k;
7       int ans=0;
8       while(n)
9       {
10          if(n%k>0)ans++;
11          n/=k;
12      }
13      cout<<ans<<endl;
14      return 0;
15  }
```

（1）（1 分）把第 10 行 n%k>0 改成 n%k 不影响程序运行结果。（）

答案　√

解析：　n%k 的值为真，执行循环所以结果不影响。

（2）输入的 k 需要大于 1。（）

答案　√

解析：　k>1 时，k 进制才有意义。

（3）输入的 n 需要大于 0。（ ）

解析：答案　×

n 可以为 0。

（4）该算法的时间复杂度为 O（logn）。（ ）

答案 √

解析：该算法的复杂度为 n 在 k 进制意义下的位数，即 O（logn）。

（5）输入 125 5 的输出结果为（ A ）。

A.1 B.2 C.3 D.4

解析：125%5=0 25%5=0 5%5=0 1%5=1，所以输出 ans 值为 1。

（6）当 n 在 int 范围内时，输出的位数最大值为（ C ）。

A.29 B.30 C.31 D.32

解析：n=2^31-1,k=2 时输出位数最大为 31。2^31 是 1 后面 31 个 0，32 位，减去 1，就剩 31 位数。

(2)

```
1   # include<iostream>
2   using namespace std;
3   int equationCount(int n,int m)
4   {
5       if(n==1||m==1)
6           return 1;
7       else if(n<m)
8           return equationCount(n,n);
9       else if(n==m)
10          return 1+equationCount(n,n-1);
11      else
12          return equationCount(n,m-1)+equationCount(n-m,m);
13  }
14  int main( )
15  {
16      int n;
17      cin>>n;
18      cout<<equationCount(n,n)<<endl;
19      return 0;
20  }
```

（1）（1 分）输入的 n 必须为正整数。（ ）

答案 √

解析： 整数的划分，n<1 时会无限递归。

（2）把第 9 行的“else if(n==m)”和第 10 行的“return 1＋equationCount（n，n-1）；”去掉，不影响程序运行结果。（）

答案 ×

解析：若不判 n==m，执行 equationCount(n-m,m);，则会导致递归到 equationCount(0,m)而无限递归。

（3）把第 18 行的“n,n’”改成“n,n+1””不影响程序运行结果。（ ）

答案 √

解析：显然 n 的正整数无序拆分中所有的数都不超过 n。

（4）把第 7 行的“else if(n<m)”和第 8 行的 return equationCount（n，n）；”去掉，不影响程序运行结果。（）

答案 ×

解析：若不判 n<m，则会导致递归到负数而无限递归。

（5）输入 7 的输出结果为（ D ）。

A.12 B.13 C.14 D.15

解析：

有 15 种拆分方案。

（6）该算法的时间复杂度为（ D ）。

A.O(log n) B.O(n) C.O(n^2) D.以上都不是

解析：

由于没有使用记忆化，故该算法会把 n 的所有拆分方案都枚举一遍，时间复杂度为指数级。

(3)

```
#include <iostream>
using namespace std;
const int maxn=100000;
int a[maxn],b[maxn],n;
int Search(int num,int low,int high)
{
    int mid;
    while(low<=high)
    {
        mid=(low+high)/2;
        if(num>=b[mid])low=mid+1;
        else high=mid-1;
    }
    return low;
}
int main()
{
    int len,pos;
    cin>>n;
    for(int i=1;i<=n;i++)
        cin>>a[i];
    b[1]=a[1];
    len=1;
    for(int i=2;i<=n;i++)
    {
        if(a[i]>=b[len])
        {
            len++;
            b[len]=a[i];
        }
        else
        {
```

```
33                pos=Search(a[i],1,len);
34                b[pos]=a[i];
35            }
36        }
37        cout<<len<<endl;
38        return 0;
39}
```

●

程序解析：

程序实现了利用贪心加二分求给定数组的 LIS（最长不下降子序列），算法流程为保存 b[i]，表示当前长度为 i 的上升子序列的末尾最小值，每次二分一个最大的 x 满足 b[x]<=a[i]，可得以 a[i]为结尾的最长不下降子序列长度为 x＋1，把 b[x+1]设为 a[i]。

（1）（1 分）输入的 a[i]必须在 [1，n] 范围内。（）

答案 ×

解析： a[i]可以是 int 范围内任意数。

（2）（1 分）把第 11 行的“mid+1”改成“mid”不影响程序运行结果。（）

答案 ×

解析： mid＋1 改成 mid 之后会死循环。

（3）当数组 a 单调不降时输出为 1。（）

答案 ×

解析：当数组 a 单调不降时 LIS 长度为 n。

（4）数组 b 内的元素始终单调不降。（）

答案 √

解析：a 数组大于 b 数组最后一位数，就把 a[i]增加到 b 数组，如果小于就找到 b 数组中第一个大于等于 a[i]的数替换掉，所以 b 数组是单调不下降的。

（5）当输入第一行为 20，第二行为 1 20 2 19 3 18 4 17・・・9 12 10 11 时，输出为（ C ）。

A.1 B.10 C.11 D.20

解析：1 20 2 19 3 18 4 17 5 16 6 15 7 14 8 13 9 12 10 11 这个序列的 LIS 为 1 2 3 4 5 6 7 8 9 10 11，长度为 11。

（6）该算法的时间复杂度为（ B ）。

A.O(n) B.O(n log n) C.O(n2) D.O(n2log n)

解析：每次都二分时间复杂度 O(nlogn)。

完善程序

1．给一个矩阵 N×M，给你第 i 行 1 的个数和位置，让你选一些行精确覆盖 M 列（精确覆盖：每列有且只有 1 个 1）。

例如：如下的矩阵：

11100001

10001110

10010110

00010010
00001100
就包含了这样一个集合（第 1、4、5 行）。
Input
多组数据，对于每组数据：
第一行两个整数 N，M；
接下来 N 行，每行开头一个整数 x，表示该行上 1 的个数，接下来 x 个整数，每个 1 的位置。
Output
如果有解随意输出一组集合，否则输出 NO，中间空格隔开。

```
#include<bits/stdc++.h>
using namespace std;
# define ll long long
# define inf 1000000000
# define N 2005
# define M 2000005
int read()
{
    int x=0,f=1;char ch=getchar();
    while(ch<'0'||ch>'9'){if(ch=='-')f=-1;ch=getchar();}
    while(ch>='0'&&ch<='9'){x=x*10+ch-'0';ch=getchar();}
    return x*f;
}
int n,m;
int h[N],s[N],q[N];
int u[M],d[M],L[M],R[M],C[M],X[M];
void del(int c) //delete ROW C
{
    _____①_____
    ______②_____
    for(int i=d[c];i!=c;i=d[i])
    for(int j=R[i];j!=i;j=R[j])
        u[d[j]]=d[u[j]]=j,s[C[j]]--;
}
void add(int c)
{
    L[R[c]]=R[L[c]]=c;
    for(int i=u[c];i!=c;i=u[i])
    for(int j=L[i];j!=i;j=L[j])
        u[d[j]]=d[u[j]]=j,s[C[j]]++;
}
Void link(int r,int c)
{
    static int size=0;size++;
    X[size]=r;C[size]=c;
    s[c]++;
    d[size]=d[c];
    u[size]=c;u[d[size]]=size;
    d[u[size]]=size;
    if(h[r]==-1)
        h[r]=L[size]=R[size]=size;
    else
```

```
    {
        R[size]=R[h[r]];
        L[size]=h[r];
        L[R[size]]=size;
        R[L[size]]=size;
    }
}
bool dance(int k)
{
    if(R[0]==0)
    {
        printf("%d",k);
        for(int i=1;i<=k;i++)
        printf(" %d",X[q[i]]);
        puts("");
        return 1;
    }
    int mn=inf,c;
    for(int i=R[0];i;i=R[i])
    if(s[i]<mn)mn=s[i],c=i;
    ___③___
    for(int i=d[c];i!=c;i=d[i])
    {
        q[k+1]=i;
        for(int j=R[i];j!=i;j=R[j]) ___④_____
        if(dance(k+1))  return 1;
        for(int j=L[i];j!=i;j=L[j]) ____⑤_____
    }
    add(c);
    Return 0;
}
int main(   )
{
    while(scanf("%d%d",&n,&m)!=EOF)
    {
        for(int i=0;i<=m;i++)
        {
            d[i]=u[i]=i;
            L[i+1]=i;R[i]=i+1;
            s[i]=0;
        }
        R[m]=0;size=m;
        int x,y;
        for(int i=1;i<=n;i++)
        {
            h[i]=-1;
            x=read();
        while(x--)
            {
                y=read();
                link(i,y);
```

```
94              }
95          }
96      if(! dance(0))puts("NO");
97      }
98      return 0;
99  }
```

（1）①处应填（ B ）。

A.L[L[c]]=R[c];　　B.L[R[c]]=L[c];

C.R[L[c]]=R[c];　　D.R[R[c]]=L[c];

解析：del 函数删除 c 列上所有 1 元素所在的行，链表删节点 c，答案 B 或者 C，如果 1 选择 B，2 就选择 C。

（2）②处应填（ C ）。

A.L[L[c]]=R[c];　　B.L[R[c]]=L[c];

C.R[L[c]]=R[c];　　D.R[R[c]]=L[c];

解析：答案 C 或者 B

①处和②处链表删节点 c，即为把 L[c]的右指针设为 R[c]，R[c]的左指针设为 L

（3）③处应填（ A ）。

A.del(c);　　B.add(c);　　C.del(mn);　　D.add(mn);

解析：上面 for 循环，选择元素最少的列 c，删掉第 c 列，因此调用 del。

（4）④处应填（ D ）。

A.add(j);　　B.del(j);　　C.add(C[j]);　　D.del(C[j]);

【解析】删除 j 对应的列。

（5）⑤处应填（ C ）。

A.add(j);　　B.del(j);　　C.add(C[j]);　　D.del(C[j]);

解析：

恢复 j 对应的列，C[j]是列号。

2.（树的直径）给定一颗 n 个节点的树，每条边有个长度 wi，求这棵树直径的长度。树的直径是指树的最长简单路。

做法：两次 BFS。开始任选一点 u 作为起点进行 BFS，找到最远的一点 s，再从 s 再次 BFS 找到最远的一点 t。s-t 两点的路径长度就是直径的长度。

```
1   #include <iostream>
2   #include <cstring>
3   using namespace std;
4   const int inf＝0x3f3f3f3f; //假设的无穷大值，具体数值为 1061109567
5   const int maxn=1005;
6   struct Node{
7       int to, w, next; //临接表节点，to 表示这条边的终点，w 表示这条边的长度，next 表示下一条边的编号
8   }edge[maxn*2];
```

```
int head [maxn], tot; //head [u] 表示 u 的临接表头节点的标号
int n; //节点树
int dis [maxn]; //离起点的距离
bool vis [maxn]; //某个点是否已经拜访过
int que [maxn], first, last; //队列
void init()
{
    memset(head,-1,sizeof(head));
    tot=0;
}
void addedge(int u,int v,int w)
{
    edge[tot].to=v;
    edge[tot].w=w;
    edge[tot].next=head[u];
    head[u]=tot++;
}
int BFS(int u)
{
    first=last=0;
    memset(dis,inf,sizeof(dis));
    memset(vis,0,sizeof(vis));
    dis[u]=0;
    vis[u]=1;
    que[last++]=u;
    while(____①___)
    {
        u=que[first++];
        for(int i=head[u];i!=-1;i=edge[i].next)
        {
            int v=edge[i].to;
            if(___②___)
            {
                vis[v]=1;
                que[last++]=v;
                ____③__;
            }
        }
    }
    int tmp=1;
    for(int i=2;i<=n;i++)
        if( ____④____)tmp=i;
    return tmp;
}
int main()
{
    int u,v,w,s,t;
    cin>>n;
    init();
    for(int i=1;i<n;i++)
    {
```

```
60          cin>>u>>v>>w;
61          addedge(u,v,w);
62          addedge(v,u,w);
63      }
64      s=BFS(1);
65      ____⑤___;
66      cout<<dis[t]<<endl;
67      return 0;
68  }
```

（1）①处应填（　A　）。

A.first<last　　B.first<=last　　C.first<last-1　　D.last==n

解析：

队列不为空。

（2）②处应填（　A　　）。

A.!vis[v]　　B.vis[v]　　C.vis[u]　　D.dis[v]

解析：

!vis[v] 未访问过

（3）③处应填（　B　）。

A.dis[v]=dis[u]+1　　B.dis[v]=dis[u]+edge[i].w

C.dis[u]=dis[v]+1　　D.dis[u]=dis[v]+edge[i].w

解析：

计算 dis[v]为 dis[u]加上（u，v）边权。

（4）④处应填（　C　）。

A.dis[i]<dis[tmp]　　B.dis[i]<tmp

C.dis[i]>dis[tmp]　　D.dis[i]==tmp

解析：

打擂台，找到一个比 dist[temp]更大的值，就更新 temp。

（5）⑤处应填（　A　）。

A.t=BFS(s)　　B.t=BFS(1)

C.t=BFS(t)　　D.s=BFS(t)

解析：

看前面题目描述就能选出正确答案，“做法：两次 BFS。开始任选一点 u 作为起点进行 BFS，找到最远的一点 s，再从 s 再次 BFS 找到最远的一点 t。s-t 两点的路径长度就是直径的长度。”