一、选择题

1.计算机访问一次内存、SSD 硬盘、SATA 硬盘的时间大概分别是多少？

A、几微秒，几毫秒，几十毫秒

B、几十纳秒，几十微秒，几十毫秒

C、几十纳秒，几十微秒，几十毫秒

D、几微秒，几十微秒，几十毫秒

2.八进制的 256 用七进制表示是多少？

A、356　　B、336　　C、338　　D、346

3.若进程在内存中占 3 页（开始时内存为空），若采用先进先出（LRU）页面淘汰算法，当执行如下访问页号序列后 0，1，7，8，6，2，3，7，2，9，8，1，0，2 会发生多少缺页？

A、8,32　　B、32,8　　C、32,6　　D、8,30

4.以下关于链式存储结构说法错误的是（）

A、查找节点时链式存储比顺序存储快

B、每个节点是由数据域和指针域组成

C、比顺序存储结构的存储密度小

D、逻辑上不相邻的节点物理上可能相邻

5.假定一个二维数组的定义语句为

“int

a[3][4]={{3,4},{2,8,6}

};”

，则元素 a[1][2]的值为（）

A、6　　B、4　　C、2　　D、8

6.下面函数的功能是（）

```
int fun (char *s)
{
    char *p=s;
    while(*p++);
    return p-s-1;
}
```

A、计算字符串的位(bit)数

B、复制一个字符串

C、求字符串的长度

D、求字符串存放的位置

7.判断有向图是否存在回路，利用（）方法最佳

A、拓扑排序

B、求最短路径

C、求关键路径

D、广度优先遍历

8.依次读入数据元素序列{a,b,c,d,e,f,g}进栈，元素进栈或出栈顺序是未知的，下列序列中，不可能成为栈空时弹出的元素构成序列的有（）

A、{d,e,c,f,b,g,a}

B、{c,d,b,e,f,a,g}

C、{e,f,d,g,c,b,a}

D、{f,e,g,d,a,c,b}

9.下列有关图的遍历说法中，不正确的是（）

A、有向图和无向图都可以进行遍历操作

B、基本遍历算法两种：深度遍历和广度遍历

C、图的遍历必须用递归实现

D、图的遍历算法可以执行在有回路的图中

10.在 16 位机器上跑下列 foo 函数的结果是（）

```
void foo()
{
   int i = 65536;
  cout << i <<'','';
   i = 65535;
   cout << i;
}
```

A、-1,65535　B、0,-1　C、-1,-1　D、0,65535

11.有一段年代久远的 C++代码，内部逻辑复杂，现在需要利用其实现一个新的需求，假定有以下可行的方案，应当优先选择（）

A、修改老代码的接口，满足新的需求

B、将老代码抛弃，自己重新实现类似的逻辑

C、修改老代码的内部逻辑，满足新的需求

D、在这段代码之外写一段代码，调用该代码的一些模块，完成新功能需求

12.在 5 个页框上使用 LRU 页面替换算法，当页框初始为空时，引用序列为 0、1、7、8、6、2、3、7、2、9、8、1、0、2，系统将发生（）次缺页

A、13　　B、12　　C、11　　D、8

13.阿里巴巴有相距 1500km 的机房 A 和 B，现有 100GB 数据需要通过一条 FTP 连接在 100s 的时间内从 A 传输到 B。已知 FTP 连接建立在 TCP 协议之上，而 TCP 协议通过 ACK 来确认每个数据包是否正确传送。网络信号传输速度 2*108m/s，假设机房间带宽足够高，那么 A 节点的发送缓冲区可以设置为最小（）

A、18M　　B、12M　　C、6M　　D、24M

14.有三个结点的，可以构成多少个种叉树？

A、5　　B、13　　C、12　　D、15

15.一副牌 52 张(去掉大小王)，从中抽取两张牌，一红一黑的概率是多少？

A、25/51　　B、1/3　　C、1/2　　D、26/51

16.设某文件经内排序后得到 100 个初始归并段（初始顺串），若使用多路归并排序算法，且要求三趟归并完成排序，问归并路数最少为（）

A、8　　B、7　　C、6　　D、5

17.一个优化的程序可以生成一 n 个元素集合的所有子集，那么该程序的时间复杂度是（）

A、O(n!)　　B、O(2n)　　C、O(n2)　　D、O(n log n)

18.快速排序在已经有序的情况下效率最差，复杂度为（）

A、O(n log n)　　B、O(n2)　　C、O(n1.5)　　D、O(n2 log n)

19.有一堆石子共 100 枚，甲乙轮流从该堆中取石子，每次可取 2、4 或 6 枚，若取得最后的石子的玩家为赢，若甲先取，则（）

A、谁都无法取胜　　B、乙必胜　　C、甲必胜　　D、不确定

20.现有一完全的 P2P 共享协议，每次两个节点通讯后都能获取对方已经获取的全部信息，现在使得系统中每个节点都知道所有节点的文件信息，共 17 个节点，假设只能通过多次两个对等节点之间通讯的方式，则最少需要（）次通讯

A、32　　B、31　　C、30　　D、29

二、解答题

21.设计一个最优算法，查找 n 个元素数组的最大值和最小值，要比较 2n 次；请写一个最高效的算法，并说明他要比较的次数。请注意复杂度的常数(不用写代码，说明步骤和过程即可，要定出比较的次数，没写不给分)

22.已知三个升序整数数组 a[l], b[m]和 c[n]。请在三个数组中各找一个元素，是的组成的三元组距离最小。三元组的距离定义是：假设 a[i]、b[j]和 c[k]是一个三元组，那么距离为:

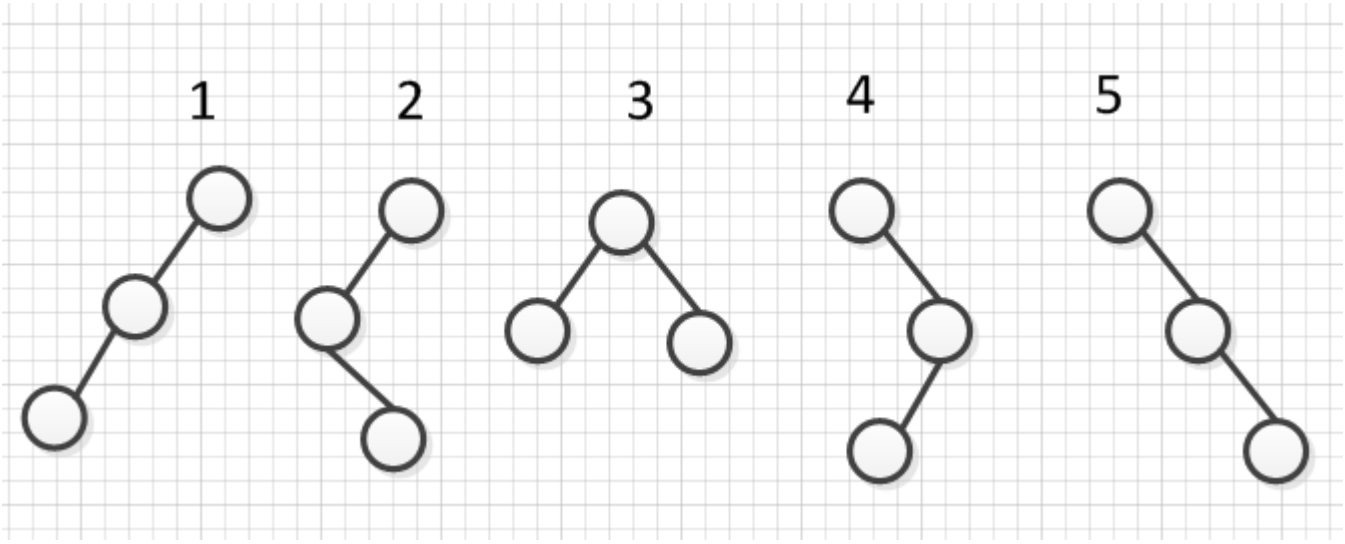
$$Distance = max(|a[i] - b[j]|, |a[i] - c[k]|, |b[j] - c[k]|)$$

请设计一个求最小三元组距离的最优算法，并分析时间复杂度。

23.请设计一个算法，在满足质因数仅为 3,5,7 或其组合的数中，找出第 K 大的数。比如 K=1,2,3 时，分别应返回 3,5,7。要求算法时间复杂度最优。

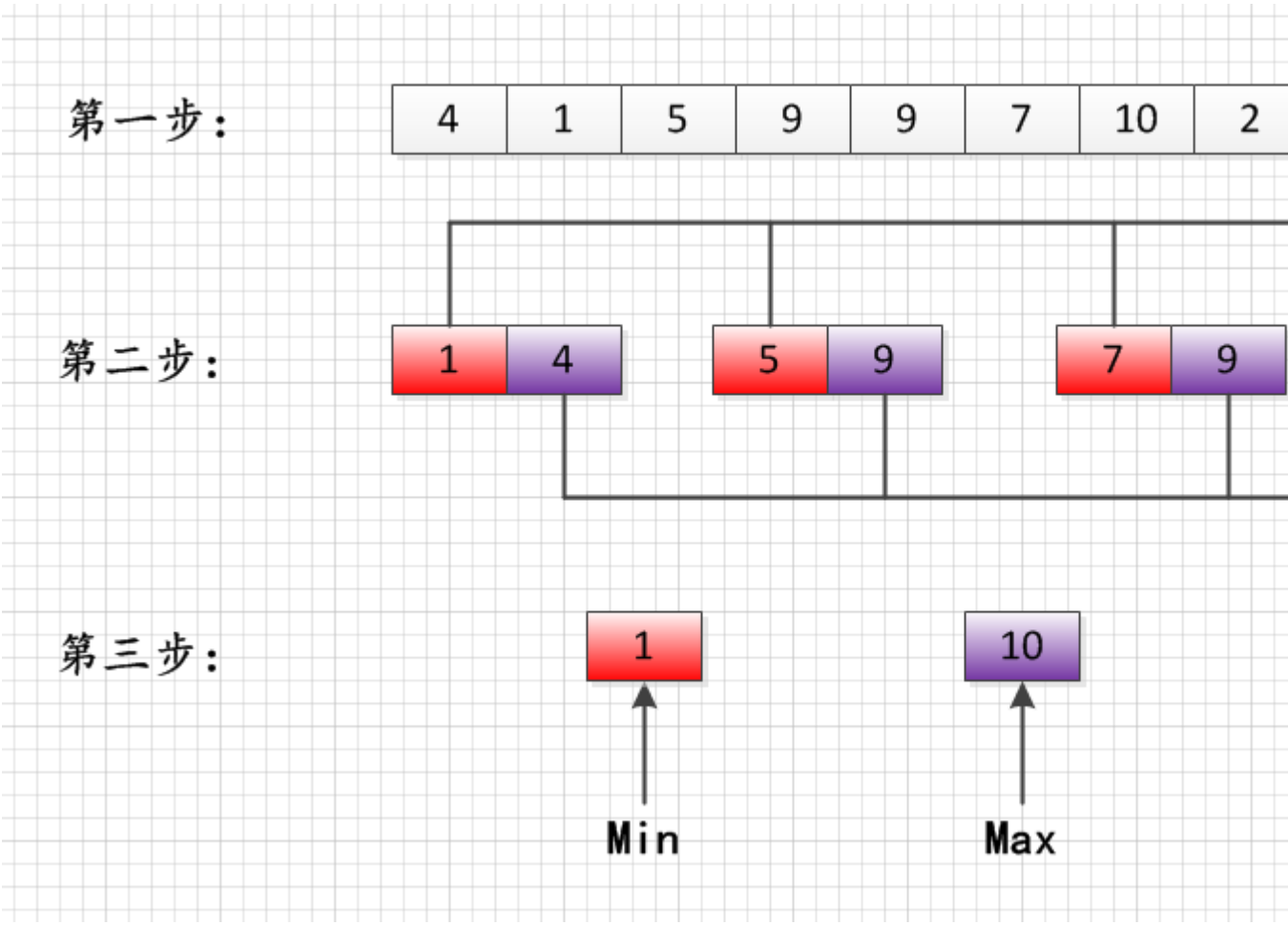
24.在黑板上写下 50 个数字：1 至 50。在接下来的 49 轮操作中，每次做如下动作：选取两个黑板上的数字 a 和 b 檫去，在黑板上写$|b-a|$。请问最后一次动作之后剩下数字可能是什么？为什么？（不用写代码，不写原因不得分）

三、答案解析（部分题目）14 解：5 种



21 解：两两一对分组，如果数组元素个数为奇数，就最后单独分一个，然后分别对每一组的两个数比较，把小的放在左边，大的放在右边，这样遍历下来，总共比较的次数是 N/2 次；在前面分组的基础上，那么可以得到结论，最小值一定在每一组的左边部分找，最大值一定在数组的右边部分找，最大值和最小值的查找分别需要比较 N/2 次和 N/2 次；这样就可以找到最大值和最小值了，比较的次数为：

N/2 * 3 = (3N)/2 次



代码实现：

```
#include <stdio.h>

#include <stdlib.h>

#define N 7

int main()

{

    int arr[N] = {4, 1, 5, 9, 9, 7, 10};

    int iter = 0;    int cnt = 0;
```

```
    for(iter = 0; iter <= N / 2 + 1  iter += 2)
    {
        if(++cnt && arr[iter] > arr[iter + 1] )
        {
            int temp = arr[iter];
            arr[iter] = arr[iter + 1];
            arr[iter + 1] = temp;
        }
    }
    int myMin = arr[0];
    for(iter = 2; iter < N  iter += 2)
    {
        if(++cnt && arr[iter] < myMin)
        {
            myMin = arr[iter];
        }
    }
int myMax = arr[1];
    for(iter = 3; iter < N; iter += 2)
```

```
    {
        if(++cnt && arr[iter] > myMax)
        {
            myMax = arr[iter];
        }
    }
    if(N % 2 != 0 && ++cnt && myMax < arr[N - 1]) myMax = arr[
N - 1];
    printf("min is %d\n", myMin);
    printf("max is %d\n", myMax);
    printf("compare times is %d", cnt);
    return 0;
}
```

22 解：第一个关键点： $\max\{|x1-x2|,|y1-y2|\} = (|x1+y1-x2-y2|+|x1-y1-(x2-y2)|)/2$ —公式（1）我们假设 x1=a[i]，x2=b[j]，x3=c[k]，则

$Distance = \max(|x1 - x2|, |x1 - x3|, |x2 - x3|) = \max(\max(|x1 - x2|, |x1 - x3|), |x2 - x3|)$ —公式（2）

根据公式（1），

max(|x1 – x2|, |x1 – x3|) = 1/2 (|2x1 – x2– x3| + |x2 – x3|)，带入公式（2），得到：

Distance = max(1/2 (|2x1 – x2– x3| + |x2 – x3|) , |x2 – x3|)

=1/2 * max(|2x1 – x2– x3| , |x2 – x3|) + 1/2*|x2 – x3| //把相同部分 1/2*|x2 – x3|分离出来

=1/2 * max(|2x1 – (x2 + x3)| , |x2 – x3|) + 1/2*|x2 – x3| //把(x2 + x3)看成一个整体，使用公式（1）

=1/2 * 1/2 *((|2x1 – 2x2| + |2x1 – 2x3|) + 1/2*|x2 – x3|

=1/2 *|x1 – x2| + 1/2 * |x1 – x3| + 1/2*|x2 – x3|

=1/2 *(|x1 – x2| + |x1 – x3| + |x2 – x3|) //求出来了等价公式，完毕！ **第二个关键点：**如何找到(|x1 – x2| + |x1 – x3| + |x2 – x3|) 的最小值，x1，x2，x3，分别是三个数组中的任意一个数，算法思想是：用三个指针分别指向 a,b,c 中最小的数，计算一次他们最大距离的 Distance ，然后在移动三个数中较小的数组指针，再计算一次，每次移动一个，直到其中一个数组结束为止，最慢(l+ m + n)次，复杂度为 O(l+ m + n)**代码如下：**

```
#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#define l 3
```

```
#define m 4

#define n 6

int Mymin(int a, int b, int c)

{

    int Min = a < b ? a : b;

    Min = Min < c ? Min : c;

    return Min;

}

int Solvingviolence(int a[], int b[], int c[])

{

    //暴力解法，大家都会，不用过多介绍了！

    int i = 0, j = 0, k = 0;     int MinSum = (abs(a[i] - b[j]) + abs(a[i]

 - c[k]) + abs(b[j] - c[k])) / 2;

//    int store[3] = {0};

    int Sum = 0;

    for(i = 0; i < l; i++)

    {

        for(j = 0; j < m; j++)

        {
```

```
            for(k = 0; k < n; k++)
            {
                Sum = (abs(a[i] - b[j]) + abs(a[i] - c[k]) + abs(b[j] - c[k
])) / 2;
                if(MinSum > Sum)
                {
                    MinSum = Sum;//
                    store[0] = i;//
                    store[1] = j;//
                    store[2] = k;
                }
            }
        }
    }
//    printf("the min is %d\n", minABC);
//    printf("the three number is %-3d%-3d%-3d\n", a[store[0]],b[
store[1]], c[store[2]]);
    return MinSum;
}
```

```
int MinDistance(int a[], int b[], int c[])
{
    int MinSum = 0; //最小的绝对值和
    int Sum = 0;  //计算三个绝对值的和，与最小值做比较
    int MinOFabc = 0; // a[i] , b[j] ,c[k]的最小值
    int cnt = 0;  //循环次数统计，最多是 l + m + n 次
    int i = 0, j = 0, k = 0;  //a,b,c 三个数组的下标索引
    MinSum = (abs(a[i] - b[j]) + abs(a[i] - c[k]) + abs(b[j] - c[k])) /
2;
    for(cnt = 0; cnt <= l + m + n; cnt++)
    {
        Sum = (abs(a[i] - b[j]) + abs(a[i] - c[k]) + abs(b[j] - c[k])) / 2
;
       MinSum = MinSum < Sum ? MinSum : Sum;
        MinOFabc = Mymin(a[i] ,b[j] ,c[k]);//找到 a[i] ,b[j] ,c[k]的最小
值
        //判断哪个是最小值，做相应的索引移动
        if(MinOFabc == a[i])
      {
                if(++i >= l) break;
```

```
        }
//a[i]最小,移动 i
        if(MinOFabc == b[j])
        {
            if(++j >= m) break;
        }//b[j]最小,移动 j
        if(MinOFabc == c[k])
        {
            if(++k >= n) break;
        }//c[k]最小,移动 k
    }
    return MinSum;}int main(void)
{
    int a[l] = {5, 6, 7};
   int b[m] = {13, 14, 15, 17};
    int c[n] = {19, 22, 24, 29, 32, 42};
    printf("\nBy violent solution ,the min is %d\n", Solvingviolence(a,
 b, c));
```

```
    printf("\nBy Optimal solution ,the min is %d\n", MinDistance(a,
b, c));
    return 0;
}
```