15. 基础4——例题练习与讲解

授课教师：游伟 副教授、孙亚辉 副教授、王文轩 讲师

授课时间：周二14:00 – 15:30，周四14:00 – 15:30（教学一楼1603）

上机时间：周四18:00 – 21:00（明德楼地下机房F）

课程主页：https://www.youwei.site/course/programming

YOJ-147. 教室排课

题目描述

信息学院有四个专业A、B、C、D，各专业入学新生人数分别是Na, Nb, Nc，Nd人。新学期开始有一门公共课，按专业划分成四个教学班，四个班在某个相同的时间段上课。已知该时间段还剩余8间教室可用，编号从1到8，每个教室能容纳的人数分别为120，40，85，50，100，140，70，100。试编一个程序，为上述四个教学班分配教室。　　找出所有可行的分配方案，对于每个方案依次输出为专业A、B、C、D分配的教室编号，按照字典顺序输出所有方案。

输入格式

一行，包含4个整数Na, Nb, Nc，Nd (20≤Na, Nb, Nc，Nd≤120)，每2个整数之间用一个空格隔开。

输出格式

如果存在分配方案，输出若干行，每行表示一种教室分配方案，包含4个整数，依次表示A、B、C、D四个专业分配的教室编号。
注意：按照字典序输出所有方案。
如果不存在分配方案，输出-1。

输入样例

109 87 120 81

输出样例

1 5 6 3
1 5 6 8
1 8 6 3
1 8 6 5
6 5 1 3
6 5 1 8
6 8 1 3
6 8 1 5

YOJ-147. 教室排课

- 基本思路：
  - 遍历所有可能：因为教室数量少（8 间），可以通过四层循环遍历所有可能的 4 间教室组合（i、j、k、l 分别代表 A、B、C、D 的教室编号）。
  - 筛选有效方案：对每个组合，检查：
  - 四个编号是否互不相同（教室不重复）。
  - 每个教室的容量是否≥对应专业的人数。
  - 按字典序输出：由于循环是从 1 到 8 依次遍历，自然会按字典序（先比较第一个编号，再第二个，以此类推）输出方案。
  - 判断无方案情况：若没有符合条件的组合，输出 -1

YOJ-147. 教室排课

```
int main() {
    bool f = false;  // 标记是否有可行方案，初始为false
    int a, b, c, d;  // 存储四个专业的人数
    // 教室容量数组：e[1]是1号教室容量，e[2]是2号教室容量，以此类推
    int e[] = {0, 120, 40, 85, 50, 100, 140, 70, 100};
    // 输入四个专业的人数
    cin >> a >> b >> c >> d;
    // 四层循环遍历所有可能的教室分配（i:A, j:B, k:C, l:D）
    for (int i = 1; i <= 8; i++) {  // A专业的教室编号i
        for (int j = 1; j <= 8; j++) {  // B专业的教室编号j
            for (int k = 1; k <= 8; k++) {  // C专业的教室编号k
                for (int l = 1; l <= 8; l++) {  // D专业的教室编号l
                    // 检查四个教室编号是否互不相同（不能重复使用教室）
                    if (i == j || i == k || i == l || j == k || j == l || k == l)
                        continue;  // 有重复则跳过当前组合
                    // 检查每个教室容量是否足够（教室容量≥对应专业人数）
                    if (a <= e[i] && b <= e[j] && c <= e[k] && d <= e[l]) {
                        // 输出可行方案
                        cout << i << " " << j << " " << k << " " << l << endl;
                        f = true;  // 标记存在可行方案
                    }
                }
            }
        }
    }
    // 如果没有找到任何可行方案，输出-1
    if (!f)
        cout << "-1" << endl;
    return 0;
}
```

YOJ-134：猜数字

题目描述

猜数字游戏是文曲星上的一款打发时间的小游戏。游戏的规则是这样的：计算机随机产生一个四位数，然后玩家猜这个四位数是什么。每猜一个数，计算机都会告诉玩家猜对几个数字，其中有几个数字在正确的位置上。　　比如计算机随机产生的数字为1122。如果玩家猜1234,因为1,2这两个数字同时存在于这两个数中，而且1在这两个数中的位置是相同的，所以计算机会告诉玩家猜对了2个数字，其中一个在正确的位置。如果玩家猜1111,那么计算机会告诉他猜对2个数字，有2个在正确的位置。　　现在给你一段猜数字的过程，你的任务是根据这段对话确定这个四位数是什么。

输入格式

　　输入第一行为一个正整数N(1<=N<=10)，表示在这段对话中共有N次问答（不允许出现重复问答）。在接下来的N行中，每行三个整数A,B,C。游戏者猜这个四位数为A，然后计算机回答猜对了B个数字，其中C个在正确的位置上。

输出格式

　　一行，如果根据这段对话能确定这个四位数，则输出这个四位数，若不能，则输出"Not sure"。

输入样例

```
6
4815 2 1
5716 1 0
7842 1 0
4901 0 0
8585 3 3
8555 3 2
```

输出样例

```
3585
```

YOJ-134：猜数字

- 基本思路：
  - 枚举+验证
  - 枚举 1000-9999，依次验证是否满足所给 n 个条件
- 注意事项：
  - 预处理：将输入的和待验证的四位数处理成四位单独的数字
  - 最终输出：答案唯一时输出答案，0个或多个时均需要输出“Not sure”
  - 如何计算猜对的数字个数（不要求位置相同）：

    例：两数 a = 1123 , b = 1113

    猜对的 1 的个数由 1 较少的 1123 决定

    所以 猜对的数字个数 = 1 较少的数中1的个数 + 2较少的数中2的个数 + 3较少的数中3的个数 + …

    = 2 + 0 + 1 + …

    = 3

YOJ-134：猜数字

方法1-主函数部分-读入和预处理

```
// 全局变量定义
int n,                  // 问答次数N（1<=N<=10）
    a[20][20],          // 存储每个猜测数字的各位数字，a[i][0]-a[i][3]表示第i个猜测的个位到千位
    b[20],              // 存储每个猜测的B值（猜对的数字个数）
    c[20],              // 存储每个猜测的C值（位置正确的数字个数）
    cnt[20][20];        // 统计每个猜测中数字0-9的出现次数，cnt[i][j]表示第i个猜测中数字j出现的次数
int main()
{
    int t,              // 临时存储输入的猜测数字
        ans = -1;       // 存储最终答案，-1表示未找到
    cin >> n;           // 读取问答次数N

    // 读取并处理N次问答数据
    for(int i = 1; i <= n; i++)
    {
        scanf("%d %d %d", &t, &b[i], &c[i]);  // 读取猜测数字A、B值、C值

        // 将猜测数字x分解为各位数字，并统计各数字出现次数
        for(int j = 0; j <= 3; j++){
            a[i][j] = t % 10;           // 存储各位数字（从个位开始存储）
            cnt[i][a[i][j]]++;          // 对应数字计数加1
            t /= 10;                    // 去掉最低位，处理下一位
        }
    }
```

YOJ-134：猜数字

方法1-主函数部分-枚举+输出

```
39      bool book = 0;  // 标记是否已找到有效解（0未找到，1已找到）
40      int x[4];       // 存储候选数字t的各位数字（个位、十位、百位、千位）
41
42      // 枚举所有可能的四位数 1000-9999
43      for(x[0] = 0; x[0] <= 9; x[0]++)
44      {
45          for(x[1] = 0; x[1] <= 9; x[1]++)
46          {
47              for(x[2] = 0; x[2] <= 9; x[2]++)
48              {
49                  for(x[3] = 1; x[3] <= 9; x[3]++)
50                  {
51                      if(check(x))
52                      {  // 检查当前数字i是否满足所有条件
53                          if(book)
54                          {
55                              // 如果已找到过一个解，又找到第二个解，则解不唯一
56                              printf("Not sure");
57                              return 0;
58                          }
59                          book = 1;  // 标记已找到解
60                          ans = x[3] * 1000 + x[2] * 100 + x[1] * 10 + x[0];// 记录当前解
61                      }
62                  }
63              }
64          }
65      }
66      // 输出结果
67      if(ans == -1)
68          printf("Not sure");  // 未找到任何解
69      else
70          printf("%d", ans);   // 输出唯一解
71      return 0;
```

YOJ-134：猜数字

方法1-验证函数部分-预处理

```
77  bool check(int x[])// 存储候选数字t的各位数字（个位、十位、百位、千位）
78  {
79      int cnt1[20],               // 存储候选数字t中各数字(0-9)的出现次数
80          b1, c1;                 // 计算出的B值（猜对数字个数）和C值（位置正确个数）
81
82      for(int i = 0; i < 20; i++)  // 初始化cnt1数组为0
83          cnt1[i] = 0;
84
85      // 统计候选数字各数字出现次数，用于计算B值
86      for(int i = 0; i <= 3; i++)cnt1[x[i]]++;       // 对应数字计数加1
```

YOJ-134：猜数字

方法1-验证函数部分-检查是否满足条件

```
86      // 检查候选数字是否满足所有N次问答的条件
87      for(int i = 1; i <= n; i++){
88          b1 = c1 = 0;  // 初始化B值和C值计算器
89
90          // 计算B值：猜对的数字个数（不考虑位置）
91          // 方法：对于每个数字0-9，取其在猜测和候选数字中出现次数的较小值，然后求和
92          for(int j = 0; j <= 9; j++)
93              b1 += min(cnt[i][j], cnt1[j]);
94
95          // 计算C值：位置正确的数字个数
96          // 方法：逐位比较猜测数字与候选数字的对应位置
97          for(int j = 0; j <= 3; j++)
98              c1 += (a[i][j] == x[j]);  // 位置相同则计数加1
99
100         // if(x[0]==0&&x[1]==1&&x[2]==9&&x[3]==9){
101         //     printf("%d %d\n",b1,c1);
102         // }
103
104         // 如果与任一问答条件不符，则该候选数字无效
105         if(b1 != b[i] || c1 != c[i])
106             return false;
107     }
108
109     return true;  // 通过所有检查，候选数字有效
```

YOJ-134：猜数字

方法2-主函数部分-读入和预处理（同方法1）

```
// 全局变量定义
int n,                  // 问答次数N（1<=N<=10）
    a[20][20],          // 存储每个猜测数字的各位数字，a[i][0]-a[i][3]表示第i个猜测的个位到千位
    b[20],              // 存储每个猜测的B值（猜对的数字个数）
    c[20],              // 存储每个猜测的C值（位置正确的数字个数）
    cnt[20][20];        // 统计每个猜测中数字0-9的出现次数，cnt[i][j]表示第i个猜测中数字j出现的次数
int main()
{
    int x,              // 临时存储输入的猜测数字
        ans = -1;       // 存储最终答案，-1表示未找到
    cin >> n;           // 读取问答次数N

    // 读取并处理N次问答数据
    for(int i = 1; i <= n; i++)
    {
        scanf("%d %d %d", &x, &b[i], &c[i]);  // 读取猜测数字A、B值、C值

        // 将猜测数字x分解为各位数字，并统计各数字出现次数
        for(int j = 0; j <= 3; j++){
            a[i][j] = x % 10;           // 存储各位数字（从个位开始存储）
            cnt[i][a[i][j]]++;          // 对应数字计数加1
            x /= 10;                    // 去掉最低位，处理下一位
        }
    }
```

YOJ-134：猜数字

方法2-主函数部分-枚举+输出（枚举方式不同于方法1）

```
    bool book = 0;  // 标记是否已找到有效解（0未找到，1已找到）

    // 枚举所有可能的四位数 1000-9999
    for(int i = 1000; i <= 9999; i++)
    {
        if(check(i))
        {  // 检查当前数字i是否满足所有条件
            if(book)
            {
                // 如果已找到过一个解，又找到第二个解，则解不唯一
                printf("Not sure");
                return 0;
            }
            book = 1;  // 标记已找到解
            ans = i;   // 记录当前解
        }
    }

    // 输出结果
    if(ans == -1)
        printf("Not sure");  // 未找到任何解
    else
        printf("%d", ans);   // 输出唯一解

    return 0;
}
```

YOJ-134：猜数字

方法2-验证函数部分-预处理（需要对t进行拆分）

```
67  bool check(int t)
68  {
69      int cnt1[20],               // 存储候选数字t中各数字(0-9)的出现次数
70          x[4],                   // 存储候选数字t的各位数字（个位、十位、百位、千位）
71          b1, c1;                 // 计算出的B值（猜对数字个数）和C值（位置正确个数）
72
73      for(int i = 0; i < 20; i++)  // 初始化cnt1数组为0
74          cnt1[i] = 0;
75
76      // 将候选数字t分解为各位数字，并统计各数字出现次数
77      for(int i = 0; i <= 3; i++){
78          x[i] = t % 10;          // 获取当前最低位数字
79          cnt1[x[i]]++;           // 对应数字计数加1
80          t /= 10;                // 去掉最低位，处理下一位
81      }
```

YOJ-134：猜数字

方法2-验证函数部分-检查是否满足条件（同方法1）

```
83      // 检查候选数字t是否满足所有N次问答的条件
84      for(int i = 1; i <= n; i++){
85          b1 = c1 = 0;  // 初始化B值和C值计算器
86
87          // 计算B值：猜对的数字个数（不考虑位置）
88          // 方法：对于每个数字0-9，取其在猜测和候选数字中出现次数的较小值，然后求和
89          for(int j = 0; j <= 9; j++)
90              b1 += min(cnt[i][j], cnt1[j]);
91
92          // 计算C值：位置正确的数字个数
93          // 方法：逐位比较猜测数字与候选数字的对应位置
94          for(int j = 0; j <= 3; j++)
95              c1 += (a[i][j] == x[j]);  // 位置相同则计数加1
96
97          // 如果与任一问答条件不符，则该候选数字无效
98          if(b1 != b[i] || c1 != c[i])
99              return false;
100     }
101
102     return true;  // 通过所有检查，候选数字有效
103 }
```