软件质量分析期末考试

课程名称:	软件质量分析	指导教师:	陈仪香
姓　　名:	王海生	学　　号:	10235101559
年　　级:	2023 级	主　　题:	期末考试

目录

1 第一题

1.1 相关知识点

对于给定的正互反判断矩阵 A，可以通过下面三种方法来估算权重向量：

1. 右特征向量法（EigenVector Method, EV）
2. 对数最小二乘法（Logarithmic Least Squares Method, LLSM）
3. 卡方最小二乘法（Chi-Square Minimization Method, CSM）。

最终应当选择 TD 值最小的方法。

1.2 答案

1.2.1 代码运行结果

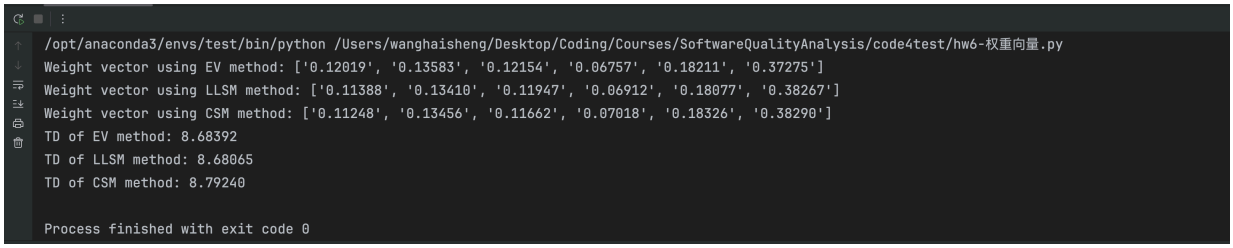


图 1: 运行结果

1.2.2 题目答案

权重向量：

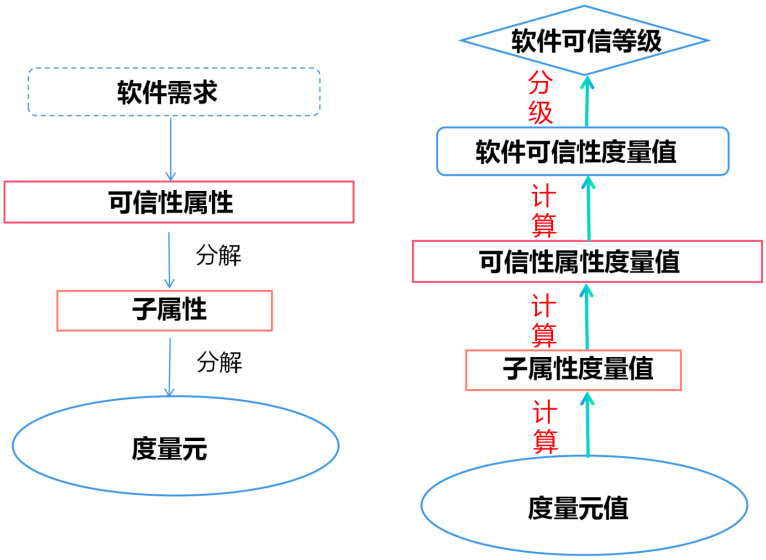
Weight vector using EV method: ['0.12019', '0.13583', '0.12154', '0.06757', '0.18211', '0.37275']
Weight vector using LLSM method: ['0.11388', '0.13410', '0.11947', '0.06912', '0.18077', '0.38267']
Weight vector using CSM method: ['0.11248', '0.13456', '0.11662', '0.07018', '0.18326', '0.38290']

“强度”TD 计算：

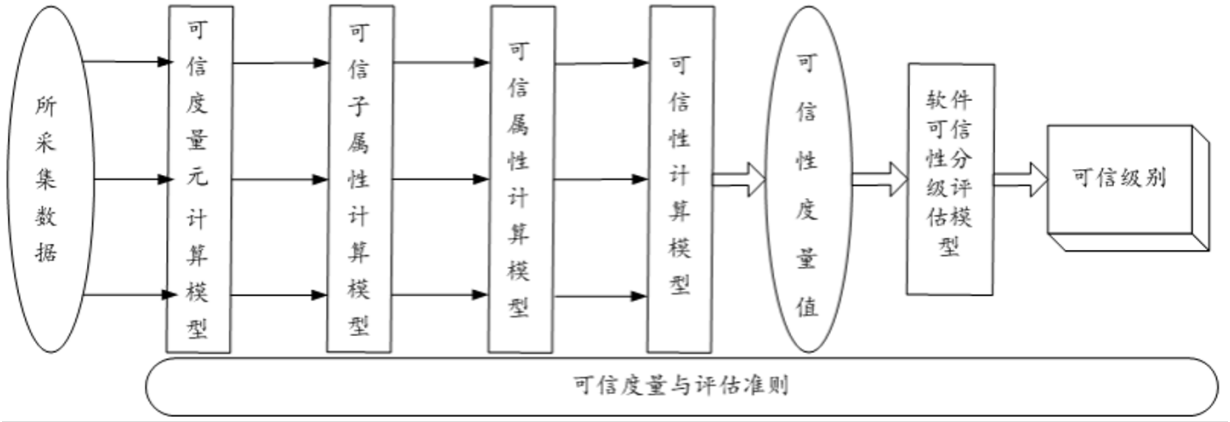
TD of EV method: 8.68392
TD of LLSM method: 8.68065
TD of CSM method: 8.79240
最终选择 TD 值最小的方法，即 LLSM 对应的 8.68065。

2 第二题

2.1 相关知识点



由课堂内容可知，分析一个软件的可信等级，需要先自上而下地将软件需求解构为可信性属性、子属性、度量元；在计算时，要自下而上地计算每一个子类，然后得到最终的可信等级。更细节的流程图如下所示。



在最后一步，通过下述分级模型来判定软件可信等级。

软件可信度量值要求（黄金分割）	可信属性要求（多数原则）	可信等级
$9.5 \leq T$	1. 低于 9.5 分的关键属性个数不超过 $n - \lceil n \cdot \frac{2}{3} \rceil$ 个 2. 没有低于 8.5 分的可信属性	V
$8.5 \leq T < 9.5$ 或者 $T > 9.5$ 且不能评为 V 级别	1. 低于 8.5 分的关键属性个数不超过 $n - \lceil n \cdot \frac{2}{3} \rceil$ 个 2. 没有低于 7.0 分的可信属性	IV
$7.0 \leq T < 8.5$ 或者 $T > 8.5$ 且不能评为 IV 级别及以上者	1. 低于 7.0 分的关键属性个数不超过 $n - \lceil n \cdot \frac{2}{3} \rceil$ 个 2. 没有低于 4.5 分的可信属性	III
$4.5 \leq T < 7.0$ 或者 $T > 7.0$ 且不能评为 III 级别及以上者	1. 低于 4.5 分的关键属性个数不超过 $n - \lceil n \cdot \frac{2}{3} \rceil$ 个	II
$T < 4.5$ 或者 $T > 4.5$ 且不能评为 II 级别及以上者	1. 无要求	I

2.2 答案

2.2.1 代码运行结果

图 2: 运行结果

2.2.2 题目答案

ID1：8.50652 8.27245 8.07854 8.49687 8.15320 8.94651 8.93288 9.24878 9.64811 软件信任值为 8.561 信任等级为 III

ID2：7.83745 8.51402 8.38193 8.44011 8.42742 8.37854 8.55674 8.68562 8.90258 软件信任值为 8.468 信任等级为 III

ID3：8.51307 8.99790 7.82639 8.39207 8.50961 8.52526 8.39207 7.64984 9.12939 软件信任值为 8.428 信任等级为 III

ID4：8.87679 9.15556 8.80373 8.73551 8.92864 9.04323 8.99919 8.99500 9.32376 软件信任值为 8.966 信任等级为 IV

2.2.3 可视化效果

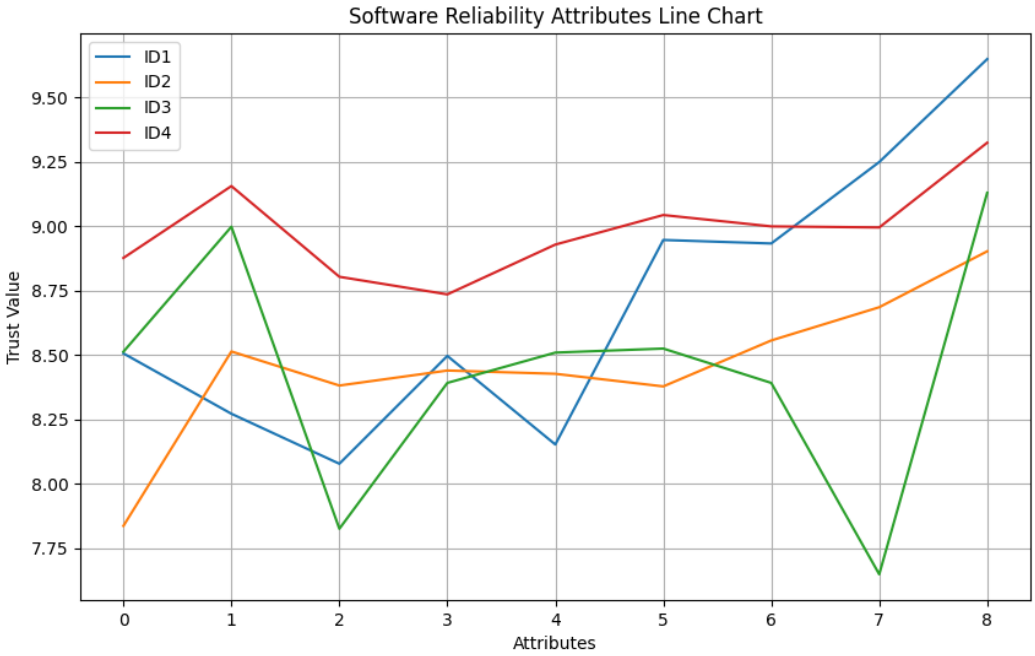


图 3: 软件可信属性折线图

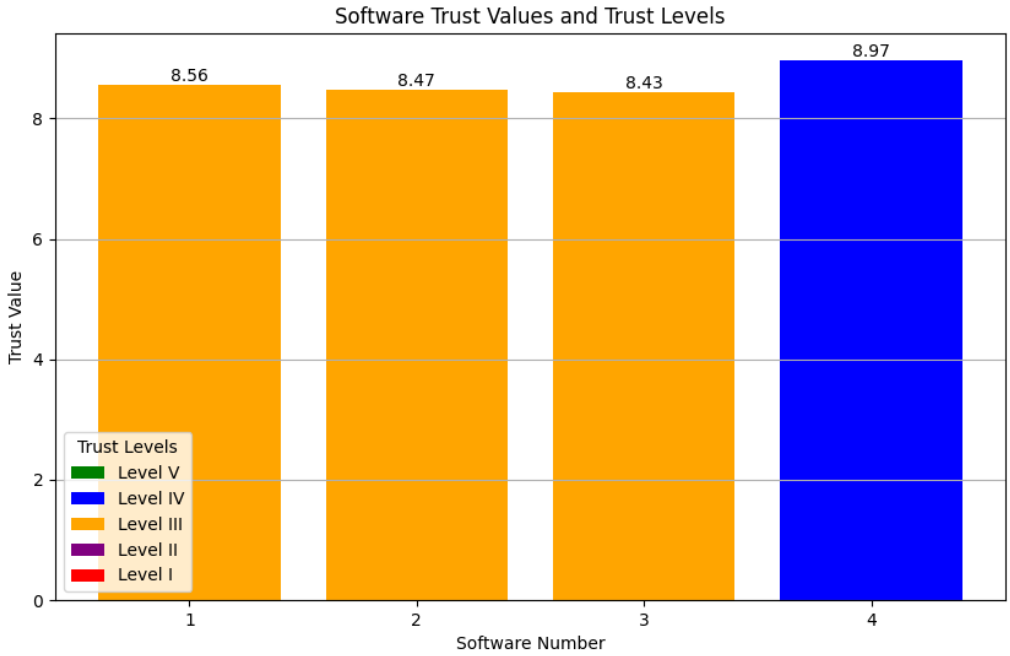


图 4: 软件可信等级柱状图

3 第三题

3.1 相关知识点

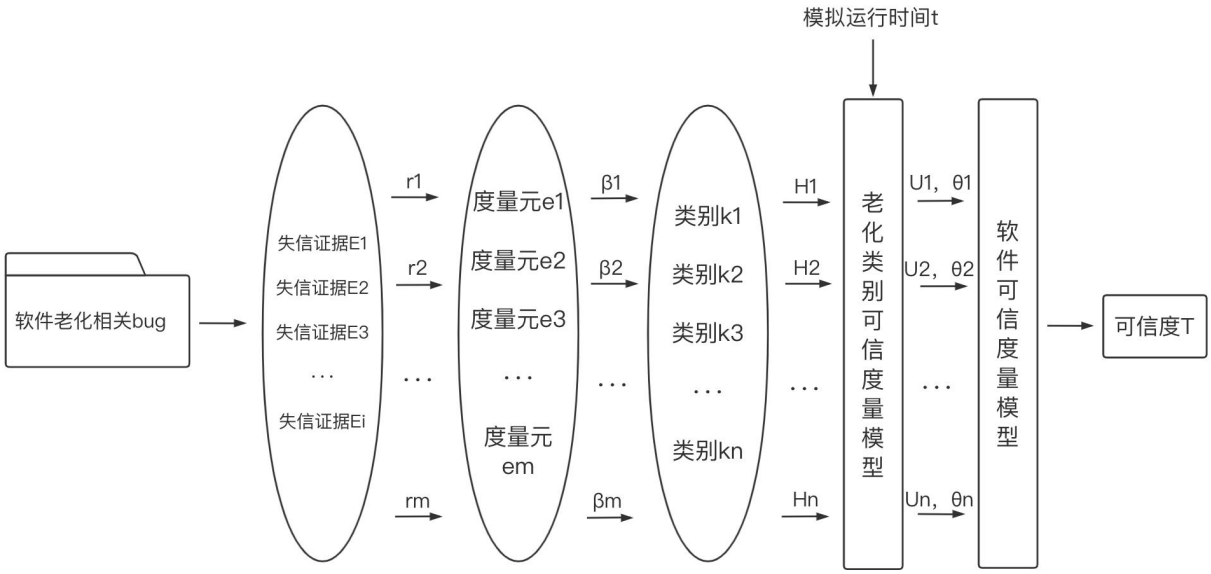


图 5: 软件老化可信计算框架

3.2 答案

3.2.1 代码运行结果

```
/opt/anaconda3/envs/test/bin/python /Users/wanghaisheng/Desktop/Coding/Courses/SoftwareQualityAnalysis/code4test/hw12-软件老化可信计算框架.py
类别数: 5
各个类别的权重: 0.539 0.125 0.238 0.049 0.049
各个类别的度量元数量: 15 4 7 2 2
第1个类别-各个度量元的权重: 0.0506 0.1845 0.0238 0.0238 0.0774 0.0774 0.0774 0.0238 0.0238 0.0506 0.0506 0.0238 0.0238 0.1042 0.1845
第1个类别-各个度量元的该时刻最大风险值: 4 4 4 1 1 1 1 1 1 4 4 4 4 4
第2个类别-各个度量元的权重: 0.25 0.25 0.25 0.25
第2个类别-各个度量元的该时刻最大风险值: 1 4 1 4
第3个类别-各个度量元的权重: 0.2340 0.1064 0.1064 0.2340 0.1064 0.1064 0.1064
第3个类别-各个度量元的该时刻最大风险值: 4 1 4 4 1 1 4
第4个类别-各个度量元的权重: 0.2 0.8
第4个类别-各个度量元的该时刻最大风险值: 1 1
第5个类别-各个度量元的权重: 0.10 0.90
第5个类别-各个度量元的该时刻最大风险值: 4 4
各类别的熵: ['0.38881', '0.30103', '0.40988', '0.00000', '0.60206']
各类别的可信值: ['6.77863', '7.40056', '6.63728', '10.00000', '5.47682']
可信值: 6.878

Process finished with exit code 0
```

图 6: `t = 0` 运行结果

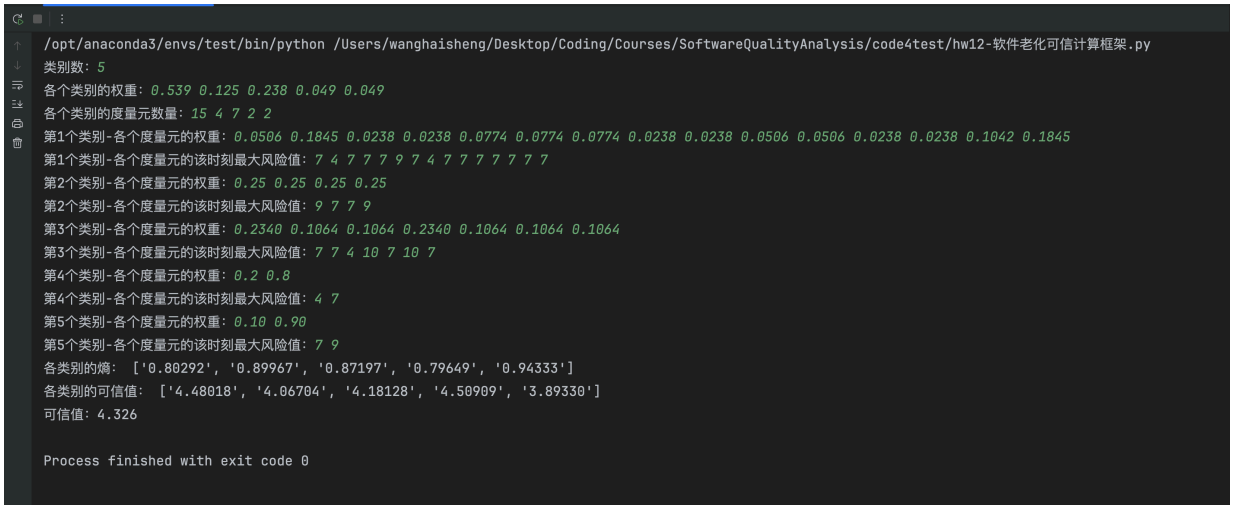


图 7: `t = 10` 运行结果

3.2.2 题目答案

t = 0 时：

各类别的熵：['0.38881', '0.30103', '0.40988', '0.00000', '0.60206'] 各类别的可信值：['6.77863', '7.40056', '6.63728', '10.00000', '5.47682'] 可信值：6.878

t = 10 时：

各类别的熵：['0.80292', '0.89967', '0.87197', '0.79649', '0.94333'] 各类别的可信值：['4.48018', '4.06704', '4.18128', '4.50909', '3.89330'] 可信值：4.326

4 附录：源代码

4.1 第一题

```
import numpy as np


# 定义正互反判断矩阵A
A = np.array([
[1, 1/2, 2, 2, 1/2, 1/4],
[2, 1 ,1, 2, 1/2, 1/3],
[1/2, 1, 1, 2, 1, 1/3],
[1/2, 1/2, 1/2, 1, 1/2, 1/5],
[2, 2, 1, 2, 1, 1/2],
[4, 3, 3, 5, 2, 1]
])

```

```
14      n = len(A)
15
16      # EV方法
17      def ev_method(A):
18      eigenvalues, eigenvectors = np.linalg.eig(A)
19      max_eigenvalue_index = np.argmax(eigenvalues.real)
20      principal_eigenvector = eigenvectors[:, max_eigenvalue_index].real
21      normalized_weights = principal_eigenvector / np.sum(principal_eigenvector)
22      return normalized_weights
23
24      # LLSM方法
25      def lls_method(A):
26      n = len(A)
27      product_list = []
28      for i in range(n):
29      product = 1
30      for j in range(n):
31      if A[i][j] == 0:
32      raise ValueError("Matrix element is zero.")
33      product *= A[i][j]
34      product_list.append(product**(1/n))
35      sum_product = sum(product_list)
36      weights = [product / sum_product for product in product_list]
37      return list(map(float, weights))
38
39
40      def csm_method(A, epsilon=1e-10, max_iter=1000):
41      n = len(A)
42      w = np.ones(n) / n  # Initial weight vector
43
44      for iteration in range(max_iter):
45      max_val, m = None, None
46
47      # Step 1: Find the index `m` with the largest discrepancy
48      for i in range(n):
49      discrepancy = 0
50      for j in range(n):
51      discrepancy += ((1 + A[j, i] ** 2) * (w[i] / w[j]) - (1 + A[i, j] ** 2) * (w
            [j] / w[i]))
52      discrepancy = abs(discrepancy)
53      if max_val is None or discrepancy > max_val:
54      max_val = discrepancy
55      m = i
56
```

```
    # Step 2: Check for convergence
    if max_val <= epsilon:
    break

    # Step 3: Update the weight w[m]
    up, bottom = 0, 0
    for j in range(n):
    if j != m:
    up += (1 + A[m, j] ** 2) * (w[j] / w[m])
    bottom += (1 + A[j, m] ** 2) * (w[m] / w[j])

    # Update weight vector
    T = np.sqrt(up / bottom)
    w[m] *= T
    w /= np.sum(w)  # Normalize weights

    return w


    # 计算每种方法的权重向量
    W_EV = ev_method(A)
    W_LLSM = lls_method(A)
    W_CSM = csm_method(A)

    # 打印权重向量
    print("Weight vector using EV method:", [f"{w:.5f}" for w in W_EV])
    print("Weight vector using LLSM method:", [f"{w:.5f}" for w in W_LLSM])
    print("Weight vector using CSM method:", [f"{w:.5f}" for w in W_CSM])

    # 计算TD（Total Deviation）
    def total_deviation(W, A):
    n = len(A)
    TD = 0
    for i in range(n):
    for j in range(n):
    TD += abs(A[i][j] - W[i] / W[j])
    return TD


    TD_EV = total_deviation(W_EV, A)
    TD_LLSM = total_deviation(W_LLSM, A)
    TD_CSM = total_deviation(W_CSM, A)

    # 打印总偏差（TD）
```

```
101     print("TD of EV method: {:.5f}".format(TD_EV))
102     print("TD of LLSM method: {:.5f}".format(TD_LLSM))
103     print("TD of CSM method: {:.5f}".format(TD_CSM))
104
105     # 最终应当选择TD值最小的方法
```

4.2 第二题

```
1     import math
2     import matplotlib.pyplot as plt
3
4
5     # 权重
6     weight = [0.05, 0.17, 0.20, 0.15, 0.09, 0.09, 0.11, 0.05, 0.09]
7
8     # 子权重
9     childWeight = [
10    0.31, 0.36, 0.33,  # 指标1子权重
11    0.33, 0.33, 0.34,  # 指标2子权重
12    0.16, 0.17, 0.17, 0.17, 0.17, 0.16,  # 指标3子权重
13    0.33, 0.34, 0.33,  # 指标4子权重
14    0.34, 0.33, 0.33,  # 指标5子权重
15    0.5, 0.5,          # 指标6子权重
16    0.33, 0.34, 0.33,  # 指标7子权重
17    0.5, 0.5,         # 指标8子权重
18    0.33, 0.33, 0.34   # 指标9子权重
19    ]
20
21
22
23    # 计算可信值
24    def calculate_trust(values, weights):
25    trust_value = 1.0
26    for i in range(len(values)):
27    trust_value *= math.pow(values[i], weights[i])
28    return trust_value
29
30    # 判断可信等级
31    def judge_trust_level_07(component_trust_values, key_component_count,
          overall_trust):
32    total_components = len(component_trust_values)
33    key_threshold = 3
34    low_key_count9_5 = 0
```

```
    low_key_count8_5 = 0
    low_key_count7_0 = 0
    low_key_count4_5 = 0
    has_low85 = False
    has_low70 = False
    has_low45 = False

    # 统计关键组件低于对应阈值的数量
    for i in range(key_component_count):
    if component_trust_values[i] < 9.5:
    low_key_count9_5 += 1
    if component_trust_values[i] < 8.5:
    low_key_count8_5 += 1
    if component_trust_values[i] < 7.0:
    low_key_count7_0 += 1
    if component_trust_values[i] < 4.5:
    low_key_count4_5 += 1

    # 统计所有组件低于对应阈值的情况
    for i in range(total_components):
    if component_trust_values[i] < 8.5:
    has_low85 = True
    if component_trust_values[i] < 7.0:
    has_low70 = True
    if component_trust_values[i] < 4.5:
    has_low45 = True

    # 判断信任等级
    if overall_trust >= 9.5 and low_key_count9_5 <= key_threshold and not
        has_low85:
    return "V"
    elif overall_trust >= 8.5 and not has_low70 and low_key_count8_5 <=
        key_threshold:
    return "IV"
    elif overall_trust >= 7.0 and not has_low45 and low_key_count7_0 <=
        key_threshold:
    return "III"
    elif overall_trust >= 4.5 and low_key_count4_5 <= key_threshold:
    return "II"
    else:
    return "I"

    # 绘制折线图
    def plot_line_chart(title, x_label, y_label, data, labels):
```

```
76      plt.figure(figsize=(10, 6))
77      for i, series in enumerate(data):
78      plt.plot(series, label=labels[i])
79      plt.title(title)
80      plt.xlabel(x_label)
81      plt.ylabel(y_label)
82      plt.legend()
83      plt.grid(True)
84      plt.show()
85
86      # 绘制条形图
87      def plot_bar_chart(software_numbers, trust_values, trust_levels):
88      # 定义等级对应的颜色
89      level_colors = {
90              "V": "green",
91              "IV": "blue",
92              "III": "orange",
93              "II": "purple",
94              "I": "red"
95      }
96
97      # 获取每个软件对应的颜色
98      colors = [level_colors[level] for level in trust_levels]
99
100     plt.figure(figsize=(10, 6))
101     bars = plt.bar(software_numbers, trust_values, color=colors)
102
103     # 添加数值标签
104     for bar in bars:
105     height = bar.get_height()
106     plt.text(bar.get_x() + bar.get_width() / 2, height, f"{height:.2f}", ha='
            center', va='bottom')
107
108     plt.title("Software␣Trust␣Values␣and␣Trust␣Levels")
109     plt.xlabel("Software␣Number")
110     plt.ylabel("Trust␣Value")
111     plt.xticks(software_numbers)
112     plt.grid(True, axis='y')
113
114     # 添加图例
115     from matplotlib.patches import Patch
116     legend_elements = [
117     Patch(facecolor='green', label='Level␣V'),
118     Patch(facecolor='blue', label='Level␣IV'),
```

```
119     Patch(facecolor='orange', label='Level III'),
120     Patch(facecolor='purple', label='Level II'),
121     Patch(facecolor='red', label='Level I')
122     ]
123     plt.legend(handles=legend_elements, title="Trust Levels")
124
125     plt.show()
126
127     # 主函数
128     def main():
129     # 4组28个子属性平均值
130     child_trust_values_list = [
131     [9.1, 8.9, 7.6, 9.2, 7.8, 7.9, 8.9, 7.8, 7.9, 7.6, 7.5, 9.0, 9.1, 7.6, 8.9,
            9.0, 7.9, 7.6, 9.2, 8.7, 8.9, 8.9, 9.0, 9.1, 9.4, 10, 10, 9.0],
132     [7.7, 7.9, 7.9, 9.0, 8.7, 7.9, 8.7, 8.2, 8.7, 8.2, 7.7, 8.9, 9.0, 7.7, 8.7,
            8.7, 7.9, 8.7, 9.0, 7.8, 7.9, 8.9, 8.9, 9.2, 8.2, 8.9, 7.9, 10],
133     [7.9, 8.9, 8.7, 9.2, 8.9, 8.9, 7.9, 7.9, 8.7, 6.2, 7.9, 8.7, 8.7, 8.7, 7.8,
            7.8, 8.9, 8.9, 9.2, 7.9, 8.7, 8.7, 7.8, 7.7, 7.6, 9.3, 9.2, 8.9],
134     [8.7, 9.2, 8.7, 9.3, 9.5, 8.7, 8.7, 9.3, 8.7, 7.7, 8.9, 9.7, 9.7, 8.7, 7.9,
            8.7, 8.9, 9.2, 9.4, 8.7, 8.9, 9.4, 8.7, 8.7, 9.3, 9.5, 9.6, 8.9]
135     ]
136
137     # 处理每一组数据
138     attribute_trust_values_list = []
139     software_trust_values = []
140     trust_levels = []
141
142     for i in range(len(child_trust_values_list)):
143     child_trust_values = child_trust_values_list[i]
144
145     # 计算各属性信任值
146     attribute_trust_values = []
147     index = 0
148     for attr in range(len(weight)):
149     sub_trusts = []
150     sub_weights = []
151     if attr == 0:
152     sub_trusts = [child_trust_values[index], child_trust_values[index+1],
            child_trust_values[index+2]]
153     index += 3
154     elif attr == 1:
155     sub_trusts = [child_trust_values[index], child_trust_values[index+1],
            child_trust_values[index+2]]
156     index += 3
```

```
157     elif attr == 2:
158     sub_trusts = [child_trust_values[index], child_trust_values[index+1],
            child_trust_values[index+2],
159     child_trust_values[index+3], child_trust_values[index+4], child_trust_values
            [index+5]]
160     index += 6
161     elif attr == 3:
162     sub_trusts = [child_trust_values[index], child_trust_values[index+1],
            child_trust_values[index+2]]
163     index += 3
164     elif attr == 4:
165     sub_trusts = [child_trust_values[index], child_trust_values[index+1],
            child_trust_values[index+2]]
166     index += 3
167     elif attr == 5:
168     sub_trusts = [child_trust_values[index], child_trust_values[index+1]]
169     index += 2
170     elif attr == 6:
171     sub_trusts = [child_trust_values[index], child_trust_values[index+1],
            child_trust_values[index+2]]
172     index += 3
173     elif attr == 7:
174     sub_trusts = [child_trust_values[index], child_trust_values[index+1]]
175     index += 2
176     elif attr == 8:
177     sub_trusts = [child_trust_values[index], child_trust_values[index+1],
            child_trust_values[index+2]]
178     index += 3
179
180     sub_weights = childWeight[index-len(sub_trusts):index]
181     attribute_trust_values.append(calculate_trust(sub_trusts, sub_weights))
182
183     attribute_trust_values_list.append(attribute_trust_values)
184
185     # 计算软件信任值
186     software_trust_value = calculate_trust(attribute_trust_values, weight)
187     software_trust_values.append(software_trust_value)
188
189     # 判断信任等级
190     trust_level = judge_trust_level_07(attribute_trust_values, len(
            attribute_trust_values), software_trust_value)
191     trust_levels.append(trust_level)
192
193     # 输出每组的各属性信任值
```

```
194     print(f"ID{i␣+␣1}: ", end="␣")
195     for attr_trust in attribute_trust_values:
196     print(f"{attr_trust:.5f}", end="␣")
197
198     # 输出软件信任值及信任等级
199     print(f"␣软件信任值为␣{software_trust_value:.3f}␣␣信任等级为␣{trust_level}")
200
201     # 可视化属性信任值
202     labels = [f"ID{i␣+␣1}" for i in range(len(child_trust_values_list))]
203     plot_line_chart("Software␣Reliability␣Attributes␣Line␣Chart", "Attributes",
            "Trust␣Value", attribute_trust_values_list, labels)
204
205     # 可视化软件信任值（条形图）
206     software_numbers = range(1, 1 + len(child_trust_values_list))
207     plot_bar_chart(software_numbers, software_trust_values, trust_levels)
208
209     if __name__ == "__main__":
210     main()
```

4.3 第三题

```
1     n = int(input("类别数："))
2     theta = list(map(float, input("各个类别的权重：").split()))
3     m = list(map(int, input("各个类别的度量元数量：").split()))
4     R = []
5     BETA = []
6     for i in range(n):
7     beta = list(map(float, input("第{0}个类别-各个度量元的权重：".format(i + 1))
          .split()))
8     r = list(map(float, input("第{0}个类别-各个度量元的该时刻最大风险值：".
          format(i + 1)).split()))
9     BETA.append(beta)
10    R.append(r)
11
12    import math
13    Hs = []
14    Us = []
15    for i in range(n):
16    H = 0
17    for j in range(m[i]):
18    H += BETA[i][j] * math.log10(R[i][j])
19    U = max(10 * math.exp(-H), 1)
20    Hs.append(H)
```

```
21      Us.append(U)
22
23      formatted_Hs = [f"{x:.5f}" for x in Hs]
24      print("各类别的熵：", formatted_Hs)
25
26      formatted_Us = [f"{x:.5f}" for x in Us]
27      print("各类别的可信值：", formatted_Us)
28
29      T = 1
30      for i in range(n):
31      T *= math.pow(Us[i], theta[i])
32      print("可信值：{0:.3f}".format(T))
```