Assignment 4

1.1 Which network(s) can accurately represent P(Flavor, Wrapper, Shape)?

Analyze whether each network captures all necessary dependencies between the variables to align with the given scenario.

Only 1 network properly represents the story, that is network (iii). This structure correctly models the relationship where the flavor influences the probabilities of both shape and wrapper. Since both shape and wrapper depend on flavor, the arrows should originate from flavor and point toward both shape and wrapper. Among the given Bayesian networks, only network (iii) correctly reflects this dependency.

1.2 Which network provides the best representation for this scenario?

Evaluate based on the compactness of the representation and how easily one can determine the necessary values in the conditional probability tables.

Since network (iii) is the only fully accurate representation, it is the best choice. The question seems to imply that both networks (i) and (iii) could be correct in part 1.1, but because the scenario states that shape and wrapper are determined by flavor, flavor must act as the parent node. Based on this reasoning, network (iii) best aligns with the given information.

1.3 Does network (i) imply that Wrapper is independent of Shape?

Yes, because there is no direct connection between wrapper and shape, this statement holds true. In network (i), flavor is dependent on both wrapper and shape, but since there is no direct arrow linking wrapper and shape, there is no causal relationship between them.

In [4]:
```
import numpy as np

# given probabilities
P_strawberry = 0.7
```

```
P_anchovy = 0.3
P_red_given_strawberry = 0.8
P_red_given_anchovy = 0.1

# 1.4 probability of red wrapper
P_red = (P_strawberry * P_red_given_strawberry) + (P_anchovy * P_red_give
print(f"probability of red wrapper: {P_red:.2f}")

# given probabilities for round shape
P_round_given_strawberry = 0.8
P_round_given_anchovy = 0.1

# 1.5 probability of strawberry given red and round (bayes' theorem)
P_red_round_given_strawberry = P_red_given_strawberry * P_round_given_str
P_red_round_given_anchovy = P_red_given_anchovy * P_round_given_anchovy
P_red_round = (P_red_round_given_strawberry * P_strawberry) + (P_red_roun

P_strawberry_given_red_round = (P_red_round_given_strawberry * P_strawber
print(f"probability of strawberry given red and round: {P_strawberry_give

# 1.6 expected value of unopened candy box
s = 1  # value of strawberry candy
a = 0.5  # value of anchovy candy

EV = (P_strawberry * s) + (P_anchovy * a)
print(f"expected value of unopened candy box: {EV:.2f}")
```

```
Probability of a red wrapper: 0.59
Probability that the candy is strawberry given it is red and round: 0.993
Expected value of an unopened candy box: 0.85
```

Task 2: Utility and Decision Making

2.1 Choosing Between a Certain $500 or a Lottery

Assume Mary has an **exponential utility function** with a risk tolerance of $500. She must choose between:

- **Option A**: Receiving $500 with certainty.
- **Option B**: A lottery with a **60% chance of winning $5000** and a **40% chance of winning $0**.

To decide, we calculate the **expected utility** for both options using:

[U(x) = -e^{-\frac{x}{R}}]

Since a rational agent maximizes **expected utility**, we compare:

[EU(A) = U(500)]

[EU(B) = 0.6 \times U(5000) + 0.4 \times U(0)]

If (EU(A) > EU(B)), Mary picks the certain $500.

In [7]:
```
import numpy as np

# risk tolerance
R = 500

# utility function
U = lambda x: -np.exp(-x / R)

# expected utility for option A (certain $500)
EU_A = U(500)

# expected utility for option B (60% $5000, 40% $0)
EU_B = 0.6 * U(5000) + 0.4 * U(0)

print(f"expected utility of option A: {EU_A:.3f}")
print(f"expected utility of option B: {EU_B:.3f}")

# decision based on expected utility
if EU_A > EU_B:
    print("Mary should choose option A (certain $500).")
else:
    print("Mary should choose option B (lottery).")
```

```
expected utility of option A: -0.368
expected utility of option B: -0.400
Mary should choose option A (certain $500).
```

2.2 Finding Risk Tolerance (R) for Indifference

Now, we find the value of (R) that makes Mary indifferent between:

- **Option 1**: A certain **$100**.
- **Option 2**: A **50% chance of winning $500** and a **50% chance of winning $0**.

We solve for (R) in:

[U(100) = 0.5 \times U(500) + 0.5 \times U(0)]

Using the **bisection method**, we approximate (R).

In [6]:
```
from scipy.optimize import bisect

# function to find R where EU(100) = 0.5 * EU(500) + 0.5 * EU(0)
def utility_diff(R):
    U_100 = -np.exp(-100 / R)
```

```
    U_500 = -np.exp(-500 / R)
    return U_100 - 0.5 * U_500 - 0.5 * (-1)

# find R using bisection method
R_value = bisect(utility_diff, 10, 500)

print(f"value of R that makes an individual indifferent: {R_value:.3f}")
```

```
value of R that makes an individual indifferent: 152.380
```

In [ ]: