1. Basic Single-Path, Single Augmentation

Input Data:

```
graph_data_1 = {
   's': {'a': {'capacity': 10, 'cost': 5}},
   'a': {'t': {'capacity': 10, 'cost': 2}}
}
```

Ans :: Max Flow: 10

Min Cost: 10×(5+2)=70

2. Capacity Bottleneck with Multiple Steps (Bottleneck Check)

```
graph_data_2 = {

   's': {'a': {'capacity': 2, 'cost': 1}},

   'a': {'b': {'capacity': 10, 'cost': 1}},

   'b': {'t': {'capacity': 3, 'cost': 1}}

}
```

Ans :: Total: Max Flow: 3, Min Cost: 6+3=9

3. Prioritizing the Cheapest Path

Input Data:

```
graph_data_3 = {

   's': {'a': {'capacity': 1, 'cost': 2}, 'b': {'capacity': 10, 'cost': 5}},

   'a': {'t': {'capacity': 1, 'cost': 3}},

   'b': {'t': {'capacity': 10, 'cost': 5}}

}
```

Ans :: Total: Max Flow: 11, Min Cost: (1×5)+(10×10)=105

4. Handling a Necessary Negative Cost Rerouting

Input Data:

```
graph_data_4 = {

    's': {'a': {'capacity': 1, 'cost': 10}, 'b': {'capacity': 1, 'cost': 1}},

    'a': {'t': {'capacity': 1, 'cost': 1}},

    'b': {'a': {'capacity': 1, 'cost': -12}} # Negative cost edge!

}
```

Ans :: Total: Max Flow: 1, Min Cost: −10 (A path can have negative net cost).

5. Checking for Unreachability
Input Data:

```
graph_data_5 = {

    's': {'a': {'capacity': 1, 'cost': 5}, 'b': {'capacity': 10, 'cost': 1}},

    'a': {'t': {'capacity': 1, 'cost': 5}}

}
```

Ans :: Total: Max Flow: 1, Min Cost: 10