1. Computing binomial coefficient

main.py

Save Run

```python
import math
def binomial_coefficient(n, k):
    return math.factorial(n) // (math.factorial(k) * math.factorial(n - k))
n = 5
k = 2
result = binomial_coefficient(n, k)
print(f"The binomial coefficient of ({n}, {k}) is: {result}")

```

Output

```
The binomial coefficient of (5, 2) is: 10

=== Code Execution Successful ===
```

2. Bellman ford

main.py

Save Run

```python
def bellman_ford(graph, source):
    vertices = graph.keys()
    distances = {vertex: float('infinity') for vertex in vertices}
    distances[source] = 0
    for _ in range(len(vertices) - 1):
        for u in vertices:
            for v, weight in graph[u].items():
                if distances[u] + weight < distances[v]:
                    distances[v] = distances[u] + weight
    for u in vertices:
        for v, weight in graph[u].items():
            if distances[u] + weight < distances[v]:
                print("Graph contains negative weight cycle")
                return
    print("Vertex Distance from Source")
    for vertex in vertices:
        print(f"{vertex}\t\t{distances[vertex]}")
graph = {
    'A': {'B': -1, 'C': 4},
    'B': {'C': 3, 'D': 2, 'E': 2},
    'C': {},
    'D': {'B': 1, 'C': 5},
    'E': {'D': -3}
}
bellman_ford(graph, 'A')
```

Output

```
Vertex Distance from Source
A       0
B       -1
C       2
D       -2
E       1

=== Code Execution Successful ===
```

3. Warshal Floyd

main.py

Save Run

```python
def floyd_warshall(graph):
    n = len(graph)
    dist = [[float('inf') for _ in range(n)] for _ in range(n)]
    for i in range(n):
        for j in range(n):
            dist[i][j] = graph[i][j]
    for k in range(n):
        for i in range(n):
            for j in range(n):
                dist[i][j] = min(dist[i][j], dist[i][k] + dist[k][j])
    return dist
graph = [
    [0, 5, float('inf'), 10],
    [float('inf'), 0, 3, float('inf')],
    [float('inf'), float('inf'), 0, 1],
    [float('inf'), float('inf'), float('inf'), 0]
]
result = floyd_warshall(graph)
for row in result:
    print(row)

```

Output

```
[0, 5, 8, 9]
[inf, 0, 3, 4]
[inf, inf, 0, 1]
[inf, inf, inf, 0]

=== Code Execution Successful ===
```

4. Meet in the middle technique

main.py

Save Run

```python
def meet_in_the_middle(target, nums):
    result = []
    for i in range(1 << len(nums)):
        subset = [nums[j] for j in range(len(nums)) if (i & (1 << j))]
        if sum(subset) == target:
            result.append(subset)
    return result
target_sum = 9
numbers = [3, 1, 4, 6, 5, 2]
print(meet_in_the_middle(target_sum, numbers))

```

Output

```
[[3, 6], [3, 1, 5], [4, 5], [3, 4, 2], [1, 6, 2]]

=== Code Execution Successful ===
```