1. Throw n dices

```
def csum(n, k, S):
   dp = [[0 for _ in range(S + 1)] for _ in range(k + 1)]
   dp[0][0] = 1
   for i in range(1, k + 1):
      for j in range(1, S + 1):
         dp[i][j] = 0
         for x in range(1, n + 1):
            if j - x >= 0:
               dp[i][j] += dp[i - 1][j - x]
   return dp[k][S]


n = 6
k = 3
S = 10
print(csum(n, k, S))
```

2. Tsp using dp

```
def tsp(dis):
   n = len(dis)
   dp = [[float('inf')] * n for _ in range(1 << n)]
   dp[1][0] = 0
   for mask in range(1 << n):
      for u in range(n):
         if mask & (1 << u):
            for v in range(n):
               if mask & (1 << v) == 0:
                  newm = mask | (1 << v)
                  dp[newm][v] = min(dp[newm][v], dp[mask][u] + dis[u][v])
   minc = min(dp[(1 << n) - 1][i] + dis[i][0] for i in range(1, n))

   return minc

# Example usage
distances = [
   [0, 10, 15, 20],
   [10, 0, 35, 25],
   [15, 35, 0, 30],
   [20, 25, 30, 0]
]

result = tsp(distances)
print(f"The minimum cost to visit all cities is: {result}")
```

3. Obst using dp

```
def op(keys, p):
    n = len(keys)
    dp = [[0] * (n + 1) for _ in range(n + 1)]
    for i in range(1, n + 1):
        dp[i][i] = p[i - 1]
    for length in range(2, n + 1):
        for i in range(1, n - length + 2):
            j = i + length - 1
            dp[i][j] = float('inf')
            sum_p = sum(p[i - 1:j])

            for r in range(i, j):
                cost = dp[i][r - 1] + dp[r + 1][j] + sum_p
                if cost < dp[i][j]:
                    dp[i][j] = cost

    return dp[1][n]

if __name__ == "__main__":
    keys = [10, 12, 20]
    p = [0.34, 0.33, 0.33]

    min_cost = op(keys, p)
    print("Minimum Search Cost of Optimal BST:", min_cost)
```