torch.einsum(equation, *operands) → Tensor

该函数提供了一种使用爱因斯坦求和约定来计算多线性表达式（即乘积和）的方法。

Parameters:

等式（ string ） - 该等式根据与操作数和结果的每个维度相关联的小写字母（索引）给出。左侧列出了操作数尺寸，以逗号分隔。每个张量维度应该有一个索引字母。右侧跟在-&gt;之后，并给出输出的索引。如果省略-&gt;和右侧，则它隐式地定义为在左侧恰好出现一次的所有索引的按字母顺序排序的列表。在操作数输入之后，将输出中未显示的索引求和。如果索引对同一操作数多次出现，则采用对角线。省略号…表示固定数量的维度。如果推断出右侧，则省略号维度位于输出的开头。
操作数（张量列表） - 计算爱因斯坦和的操作数。请注意，操作数作为列表传递，而不是作为单个参数传递。
Examples:

```
>>> x = torch.randn(5)
>>> y = torch.randn(4)
>>> torch.einsum('i,j->ij', x, y)  # outer product
tensor([[-0.0570, -0.0286, -0.0231,  0.0197],
 [ 1.2616,  0.6335,  0.5113, -0.4351],
 [ 1.4452,  0.7257,  0.5857, -0.4984],
 [-0.4647, -0.2333, -0.1883,  0.1603],
 [-1.1130, -0.5588, -0.4510,  0.3838]])

>>> A = torch.randn(3,5,4)
>>> l = torch.randn(2,5)
>>> r = torch.randn(2,4)
>>> torch.einsum('bn,anm,bm->ba', l, A, r) # compare torch.nn.functional.bilinear
tensor([[-0.3430, -5.2405,  0.4494],
 [ 0.3311,  5.5201, -3.0356]])

>>> As = torch.randn(3,2,5)
>>> Bs = torch.randn(3,5,4)
>>> torch.einsum('bij,bjk->bik', As, Bs) # batch matrix multiplication
tensor([[[-1.0564, -1.5904,  3.2023,  3.1271],
 [-1.6706, -0.8097, -0.8025, -2.1183]],

 [[ 4.2239,  0.3107, -0.5756, -0.2354],
 [-1.4558, -0.3460,  1.5087, -0.8530]],

 [[ 2.8153,  1.8787, -4.3839, -1.2112],
 [ 0.3728, -2.1131,  0.0921,  0.8305]]]])

>>> A = torch.randn(3, 3)
>>> torch.einsum('ii->i', A) # diagonal
tensor([-0.7825,  0.8291, -0.1936])

>>> A = torch.randn(4, 3, 3)
>>> torch.einsum('...ii->...i', A) # batch diagonal
tensor([[-1.0864,  0.7292,  0.0569],
 [-0.9725, -1.0270,  0.6493],
 [ 0.5832, -1.1716, -1.5084],
 [ 0.4041, -1.1690,  0.8570]])

>>> A = torch.randn(2, 3, 4, 5)
>>> torch.einsum('...ij->...ji', A).shape # batch permute
torch.Size([2, 3, 5, 4])
```