由于需要信息无损，所以建议用流式二叉树数据结构存储权重，类似于矩阵平均分块，流式二叉树数据结构，Interweavement binary tree  
一般用于储存二进制张量，它会把张量的值为1的元素的索引按  
第1级索引从高位到低位数等1位，第2级索引从高位到低位数等1位，第3级索引从高位到低位数等1位,...,第n级索引从高位到低位数等1位,  
第1级索引从高位到低位数等2位，第2级索引从高位到低位数等2位，第3级索引从高位到低位数等2位,...,第n级索引从高位到低位数等2位,  
第1级索引从高位到低位数等3位，第2级索引从高位到低位数等3位，第3级索引从高位到低位数等3位,...,第n级索引从高位到低位数等3位  
,...,  
第1级索引从高位到低位数等m位，第2级索引从高位到低位数等m位，第3级索引从高位到低位数等m位,...,第n级索引从高位到低位数等m位  
的顺序循环交织融合成一个二进制数并通过二叉树结构储存  
（比如tensor\_a[111000101][111101011]中的[111000101][111101011]交织为[111111010001100111]  
,tensor\_b[111000101][111101011][101101001]中的[111000101][111101011][101101001]交织为[111110111011000011100010111]）  
至于只有1级索引的张量，其索引的交织结果就是其索引自身  
于是该数据结构就可以储存几乎所有的二进制张量了比如二维张量  
[  
 [0, 0, 1, 0],  
 [0, 1, 0, 0],  
 [0, 0, 0, 1],  
 [0, 0, 0, 0]  
]

首先提取值为1的元素的索引：

([00][10], [01][01], [10][11])

对索引进行交织：

([0100]，[0011]，[1101])

存储为四层二叉树（序1 序2）  
11  
11/01  
01/10/10  
01/10/01（2个位一节）  
类似地，稀疏矩阵  
1 1 0 0 0 0 0 0  
0 1 1 0 0 0 0 0  
0 0 1 1 0 0 0 0  
0 0 0 1 1 0 0 0  
0 0 0 0 1 1 0 0  
0 0 0 0 0 1 1 0  
0 0 0 0 0 0 1 1  
0 0 0 0 0 0 0 1  
存储为  
1101  
1101 / 0010 / 1101  
1101 / 0010 / 1101 / 0010 / 1101 / 0010 / 1101  
可见，它只用了 44 个位，虽然可能还要存一些额外信息，比如层索引，但 44<64，可见对于大的稀疏矩阵它是挺有用的，而且它可以直接用来计算（如果与向量进行计算需要是类似的二叉树数据结构向量(序1，序2)），涉及的相应机制如下：  
1. 用二叉树数据结构存储向量，用四叉树数据结构存储矩阵  
2. 采用多个1父级2子级的父子级指针并行遍历树数据结构的每父子级层  
3. 用节掩码来指导剪枝，注意是节掩码，而非位掩码，每个节有1 bit 掩码，其功能与由 4 位一节改为 5 位一节并以多出来的一位用作掩码类似。