## Appendix A. Merge ordering

For a **merge** operation, the order that each $\mathcal{U}^{(i)}$ is merged determines the total flop count and memory needs. When $d$ is small, a *sequential merging* is commonly applied. However, when $d$ is large, we propose a *hieratical merging* approach instead. For instance, Figures 7a and 7b show the two merge orderings when $d = 4$, arriving at a total of $2I_1I_2R^3 + 2I_1I_2I_3R^3 + 2I_1I_2I_3I_4R^2$ flops to construct $\mathcal{U}^{(1,2,3,4)}$ using a sequential ordering, and $2I_1I_2R^3 + 2I_3I_4R^3 + 2I_1I_2I_3I_4R^2$ flops using a hierarchical ordering. To see how both methods scale with $I_k$ and $d$, if and $d = 2^D$, then a sequential merging gives and Both quantities are upper bounded by $4R^3\tilde{I}^d$ which is a factor of $4R^3$ times the total degrees of freedom.

We can generalize this analysis by proving theorem 1.

*Proof.*

1. Define $\tilde{I} = I_1 = \cdots = I_d$. Any merging order can be represented by a binary tree. Figures 7a and 7b show the binary trees for sequential and hierarchical merging; note that they do not have to be balanced, but every non-leaf node has exactly 2 children. Each $U^{(i)}$ corresponds to a leaf of the tree.

   To keep the analysis consistent, we can say that the computational cost of every leaf is 0 (since nothing is actually done unless tensors are merged).

   At each parent node, we note that the computational cost of merging the two child nodes is at least $2\times$ that required in the sum of both child nodes. This is trivially true if both children of a node are leaf nodes. For all other cases, define $D$ the number of leaf node descendents of a parent node. Then the computational cost at the parent is $2R^3 \cdot \tilde{I}^D$. If only one of the two child nodes is a leaf node, then we have a recursion

   $$2R^3 \cdot \tilde{I}^D = 2R^3\tilde{I} \cdot \tilde{I}^{D-1} \geq 4R^3(\tilde{I}^{D-1})$$

   which is always true if $\tilde{I} \geq 2$. If both children are not leaf nodes, then define $D_1$, and $D_2$ the number of leaves descendant of two child nodes, with $D = D_1 + D_2$. Then the recursion is

   $$2R^3 \cdot \tilde{I}^D = 2R^3\tilde{I}^{D_1}\tilde{I}^{D_2} \geq 4R^3(\tilde{I}^{D_1} + \tilde{I}^{D_2})$$

   where the bound is always true for $\tilde{I} \geq 2$ and $D_1, D_2 \geq 2$. Note that every non-leaf node in the tree necessarily has two children, it can never be that $D_1 = 1$ or $D_2 = 1$.

   The cost of merging at the root of the tree is always $2R^3\tilde{I}^d = 2R^3I$. Since each parent costs at least $2\times$ as many flops as the child, the total flop cost must always be between $2R^3I$ and $4R^3I$.

2. For the storage bound, the analysis follows from the observation that the storage cost at each node is $R^2\tilde{I}^D$, where $D$ is the number of leaf descendants. Therefore if $\tilde{I} \geq 2$, the most expensive storage step will always be at the root, with $R^2(\tilde{I}^{d_1} + \tilde{I}^{d_2} + \tilde{I}^d)$ storage cost, where $d = d_1 + d_2$ for any partition. Clearly, this value is lower bounded by $R^2\tilde{I}^d = R^2I$. And, for any partition $d_1 + d_2 = d$, for $\tilde{I} \geq 2$, it is always $\tilde{I}^{d_1} + \tilde{I}^{d_2} \leq \tilde{I}^d$. Therefore the upper bound on storage is $2R^2\tilde{I}^d = 2R^2I$.

3. It is sufficient to show that for any $d$ power of 2, a sequential merging is more costly in flops than a hierarchical merging, since anything in between has either pure sequential or pure hierarchical trees as subtrees.

   Then a sequential merging gives $2R^3\sum_{i=2}^{d}\tilde{I}^i$ flops. If additionally $d = 2^D$ for some integer $D > 0$, then a hierarchical merging costs $2R^3\sum_{i=2}^{D}2^{D-i}\tilde{I}^{2^i}$ flops. To see this, note that in a perfectly balanced binary tree of depth $D$, at each level $i$ there are $2^{D-i}$ nodes, each of which are connected to $2^i$ leaves.

   We now use induction to show that whenever $d$ is a power of 2, hierarchical merging (a fully balanced binary tree) is optimal in terms of flop count. If $d = 2$, there is no variation in merging order. Taking $d = 4$, a sequential merging costs $2R^3(\tilde{I}^3 + \tilde{I}^3 + \tilde{I}^4)$ and a hierarchical merging costs $2R^3(2\tilde{I}^2 + \tilde{I}^4)$, which is clearly cheaper. For some $d$ a power of 2, define $S$ the cost of sequential merging and $H$ the cost of hierarchical merging. Define $G = 2R^3\tilde{I}^{2d}$ the cost at the root for any binary tree with $2d$ leaf nodes. (Note that the cost at the root is agnostic to the merge ordering.) Then for $\hat{d} = 2d$, a hierarchical merging costs $2H + G$ flops. The cost of a sequential merging is

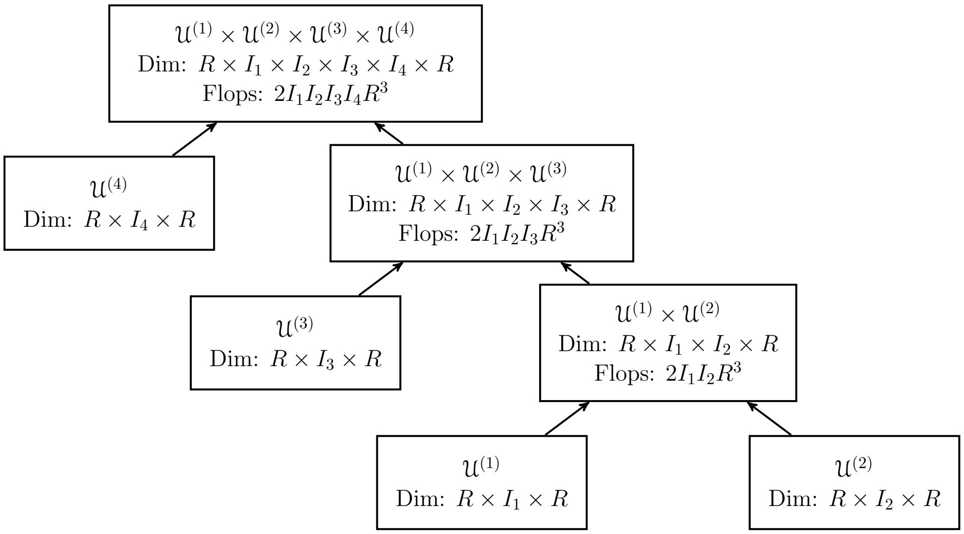
   $$\begin{aligned} S + 2R^3\tilde{I}^d\sum_{i=1}^{d}\tilde{I}^i &= S + 2R^3\tilde{I}^{d-1}\sum_{i=2}^{d}\tilde{I}^i + G \\ &= S + S\tilde{I}^{d-1} + G - 2R^3d. \end{aligned}$$

   Since $2R^3d$ is the cost at the root for $d$ leafs, $S > 2R^3d$, and therefore the above quantity is lower bounded by $G + \tilde{I}^{d-1}S$, which for $d \geq 2$ and $\tilde{I} \geq 2$, is lower bounded by $G + 2S$. By inductive hypothesis, $S > H$, so the cost of sequential merging is always more than that of hierarchical merging, whenever $d$ is a power of 2.

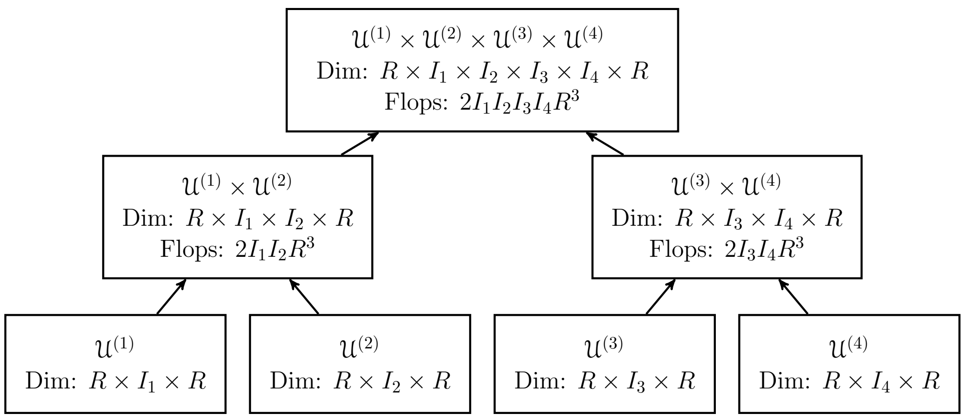
□

## Appendix B. Initialization

If $x$ and $y$ are two independent variables, then $\text{Var}[xy] = \text{Var}[x]\text{Var}[y] + \text{Var}[x](\mathbb{E}[y])^2 + \text{Var}[y](\mathbb{E}[x])^2$ [1]. Thus a product of two independent symmetric distributed random variables with mean 0 and variance $\sigma^2$ itself is symmetric




(a) Sequential merging




(b) Hierarchical merging

Figure 7: Merge ordering for a 4th order tensor ring segment of shape $R \times I_1 \times I_2 \times I_4 \times I_4 \times R$, with tensor ring rank $R$. In each node from top to bottom are tensor notation, tensor shape, and flops to obtain the tensor.

distributed with mean $0$ and variance $\sigma^4$ (not Gaussian distribution). Further extrapolating, in a matrix or tensor product, each entry is the summation of $R$ independent variables with the same distribution. The central limit theorem gives that the sum can be approximated by a Gaussian $\mathcal{N}(0, R\sigma^4)$ for large $R$. Thus if all tensor factors are drawn i.i.d. from $\mathcal{N}(0, \sigma^2)$, then after merging $d$ factors the merged tensor elements will have mean $0$ and variance $R^d\sigma^{2d}$.