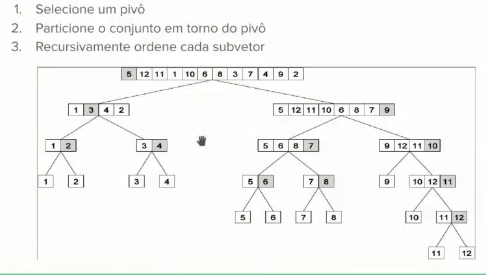
**Computação Paralela – Técnicas de Decomposição**

**Decomposição recursiva**

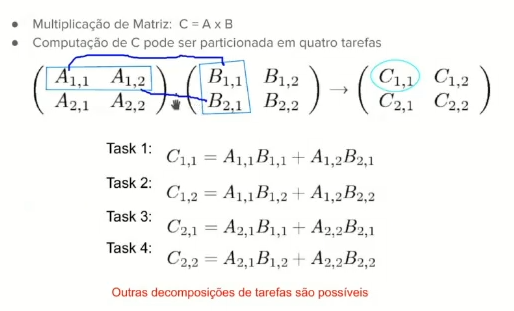
* Utilizada em problemas com abordagem “Dividir e conquistar”.
* Esta decomposição divide o problema em conjunto de sub-problemas e recursivamente divide cada sub-problema.

**Exemplo:** Quick-Sort, seleciona um Pivô e particiona o vetor em 2 vetores: os maiores que o Pivô e os menor que o Pivô, depois é escolhido um novo pivô para cada sub-vetor e isso ocorre recursivamente até chegar nas folhas.

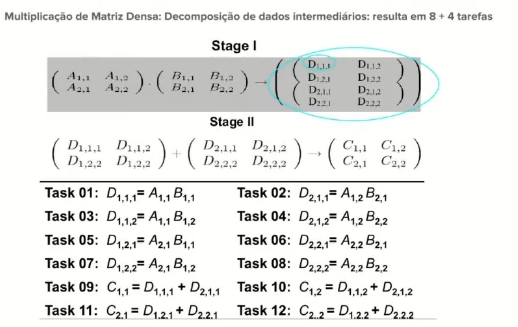


**Decomposição de dados**

* Divide os dados em várias tarefas.
* Quando a decomposição é baseada nos dados de entrada, os dados de entrada são divididos e é efetuada a operação sobre cada conjunto de dados, depois eles são agregados uns aos outros para se formar a saída. **Ex: soma vetor.**
* Quando a decomposição é baseada nos dados de saída o elemento de saída pode ser calculado independentemente então cada tarefa irá realizar a operação para as suas respectivas saídas. **Ex: multiplicação matriz por um vetor, ou multiplicação de matrizes.**

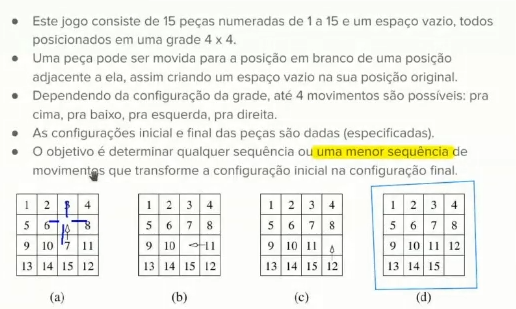
****

* Quando a decomposição é baseada nos dados intermediários é quando você divide uma etapa maior em uma intermediária, sendo que esta etapa maior pode ter sido criada para aumentar o paralelismo, os dados intermediários não são previstos em seu problema, eles surgem de acordo com as transformações no algoritmo. Este tipo de particionamento normalmente aumenta o grau de concorrências das tarefas. **Ex: multiplicação de matrizes, onde é criada uma etapa nova com dados que não existiam no princípio do algoritmo e serão usados em uma nova decomposição de tarefas para otimizar o algoritmo.**

**Neste caso particiona a multiplicação dos dados e a soma deles.**

**Decomposição exploratória**

* Particiona o espaço de busca em partes menores e a busca será efetuada nestes pedaços menores de forma concorrente até que a solução seja encontrada. **Ex: “jogo dos 15”**



* Cada um dos movimentos: ir para a esquerda, direita, para cima e para baixo gera quatro tarefas, e cada uma dessas sub-tarefas poderá no MAX criar 4 novas tarefas.
* O algoritmo que sempre encontra a melhor solução neste tipo de decomposição é o de **LARGURA, pois na busca de profundidade sempre teria quer chegar em uma folha.**

**Decomposição especulativa**

* Nesta situação, não se possui conhecimento das dependências entre as tarefas, portanto você pode possuir duas abordagens:

1. A abordagem conservativa, em que você apenas classifica a tarefa como independente quando se tem a garantia de que ela não possui realmente dependências.
2. A abordagem otimista, em que se classifica tarefas como independentes mesmo quando não se tem certeza.

* A abordagem conservativa pode produzir pouca concorrência, já a otimista será necessário um mecanismo de roll-back no caso de erros.

**Ex: Rotina de uma pessoa, podem ocorrer imprevistos, mas a priori todos eventos podem ocorrer de forma independente.**