Parallel Algorithms

Lecture 9-10

Parallel & Distributed Computing

Course Outlines

Course Name: Parallel and Distributed Computing
Credit Hours: 3(3-0)
Prerequisites: Data Communications and Computer Networks

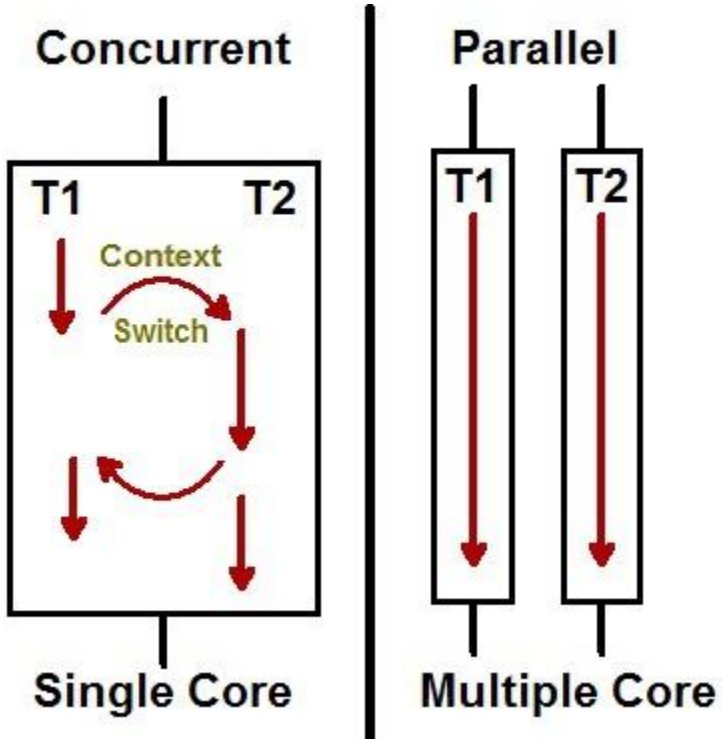
Course Outlines:

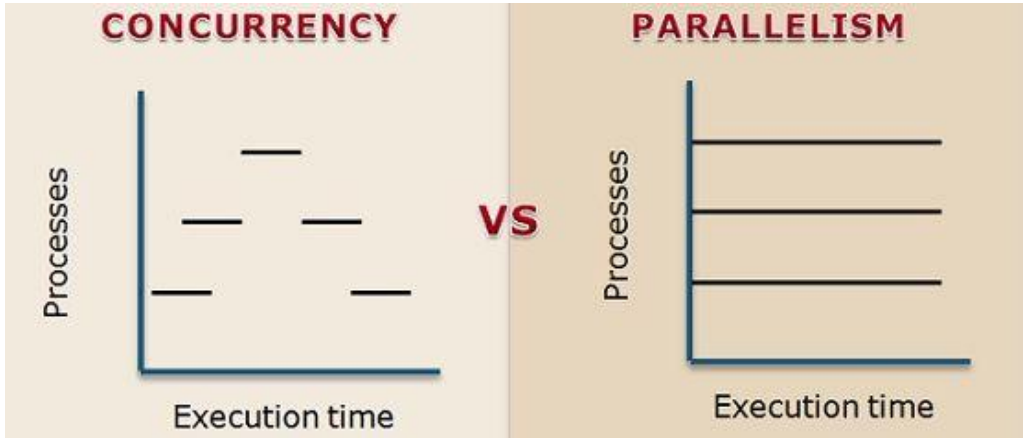
Why use parallel and distributed systems? Why not use them? Speedup and Amdahl's Law, Hardware architectures: multiprocessors (shared memory), networks of workstations (distributed memory), clusters (latest variation). Software architectures: threads and shared memory, processes and message passing, distributed shared memory (DSM), distributed shared data (DSD). Possible research and project topics, Parallel Algorithms, Concurrency and synchronization, Data and work partitioning, Common parallelization strategies, Granularity, Load balancing, Examples: parallel search, parallel sorting, etc. Shared-Memory Programming: Threads, Pthreads, Locks and semaphores, Distributed-Memory Programming: Message Passing, MPI, PVM. Other Parallel Programming Systems, Distributed shared memory, Aurora: Scoped behaviour and abstract data types, Enterprise: Process templates. Research Topics.

Parallel Algorithms

- Parallel Algorithm, is an algorithm which can do ***multiple operations in a given unit time***.

- Parallel algorithms ***executed concurrently*** most of the time.





Parallel Algorithms cont…

- Which aspect of an algorithm is ***parallel*** and which is ***concurrent***, not being clearly distinguished as both are ***correlated***.
- Algorithms vary significantly in how parallelizable they are, ranging from easily parallelizable to completely unparallelizable.

Parallel Algorithms cont…

- Some problems are ***easy to divide up into pieces*** so those could be ***solved as parallel problems***.
- Some problems ***cannot be split up into parallel portions***, as they ***require the results from a preceding step*** to effectively carry on with the next step, these are called inherently serial problems.

Parallel Algorithms cont…

- Parallel algorithms on Personal Computers have become more common since the early 2000s because of extensive improvements in ***multiprocessing systems*** and the rise of ***multi-core processors***.

Parallel Algorithms cont…
Structure

- To apply any algorithm properly, it is very important to select a proper ***data structure***.
- It is because a ***particular operation*** performed on a ***data structure*** *may take more time as compared to the same operation performed on another data structure*.

Parallel Algorithms cont…
Structure

- Therefore, the selection of a data structure must be done considering the ***architecture*** and the ***type of operations*** to be performed.
- The following data structures are commonly used in parallel programming:

1. Linked List
2. Arrays
3. Hypercube Network

Parallel Algorithms cont…
Structure

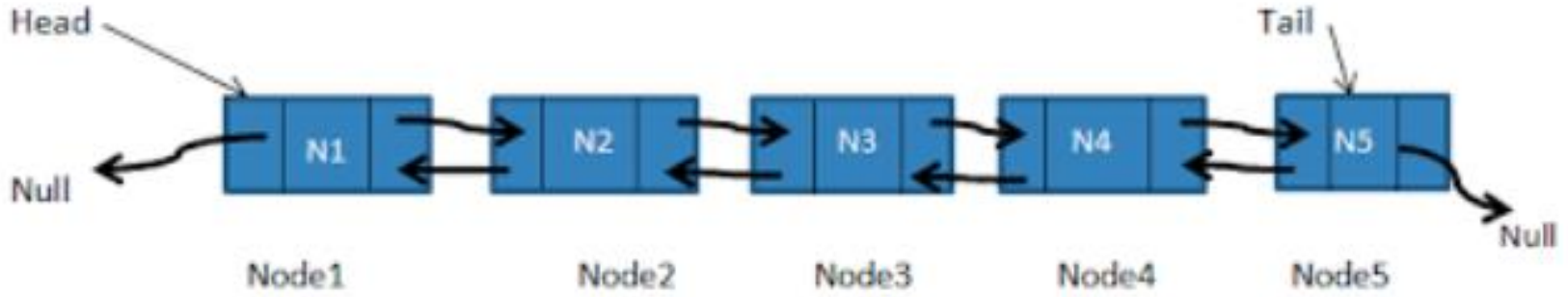
- A linked list is a data structure having zero or more nodes connected by pointers.
- Nodes may or may not occupy consecutive memory locations.
- Each node has two or three parts − one data part that stores the data and the other two are link fields that store the address of the previous or next node.

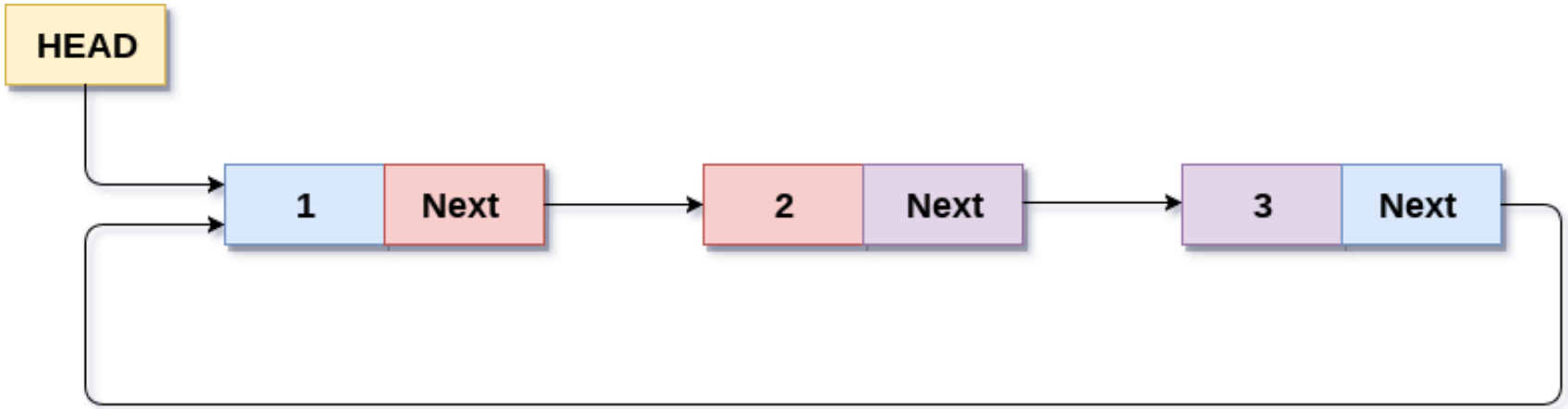
Parallel Algorithms cont…
Structure (linked lists)

- The first node's address is stored in an external pointer called head. The last node, known as tail, generally does not contain any address. There are three types of linked lists:

1. Singly Linked List
2. Doubly Linked List
3. Circular Linked List

Parallel Algorithms cont…
Structure (linked lists)





Parallel Algorithms cont...
Structure (Arrays)

- An array is a data structure where we can store similar types of data.
- It can be one-dimensional or multi-dimensional.
- Arrays can be created:

1. Statically
2. Dynamically

Parallel Algorithms cont...
Structure (Arrays)

- In statically declared arrays, dimension and size of the arrays are known at the time of compilation.
- In dynamically declared arrays, dimension and size of the array are known at runtime.

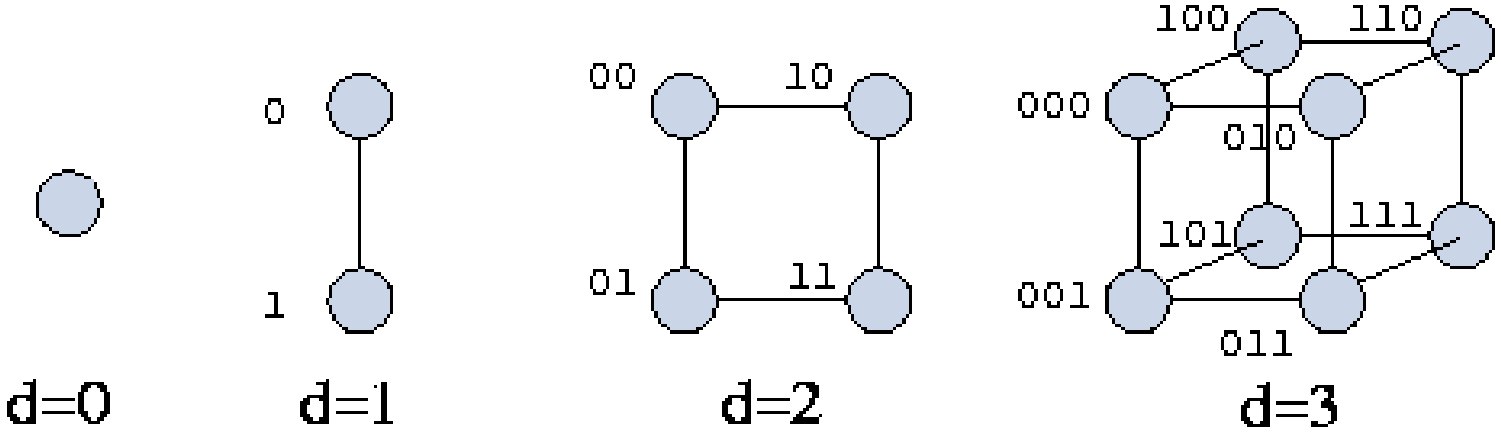
Parallel Algorithms cont…
Structure (Arrays)

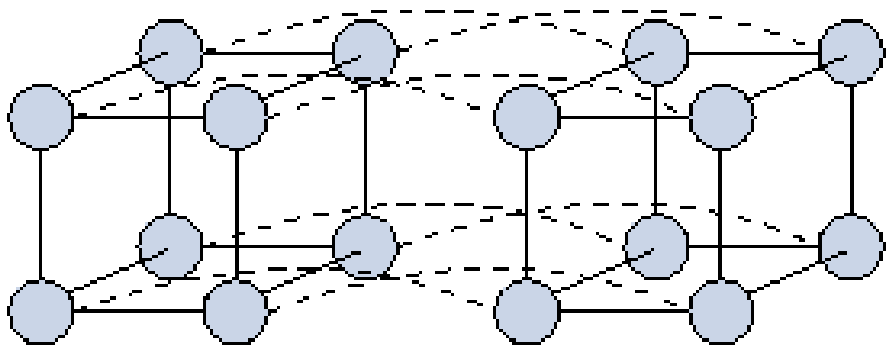
- For shared memory programming, arrays can be used as a common memory and for data parallel programming, they can be used by partitioning into sub-arrays.

Parallel Algorithms cont…
Structure (Hypercube Network)

- Hypercube architecture is helpful for those parallel algorithms where each task has to communicate with other tasks.
- Hypercube topology can easily embed other topologies such as ring and mesh.
- It is also known as n-cubes, where **n** is the number of dimensions.

Parallel Algorithms cont…
Structure (Hypercube Network)





Parallel Algorithms cont…
Design Technique

- Selecting a proper designing technique for a parallel algorithm is the most difficult and important task.
- Most of the parallel programming problems may have more than one solution.

Parallel Algorithms cont…
Design Technique

- The following designing techniques can be used for parallel algorithms:

1. Divide and conquer
2. Greedy Method
3. Dynamic Programming
4. Backtracking
5. Branch & Bound

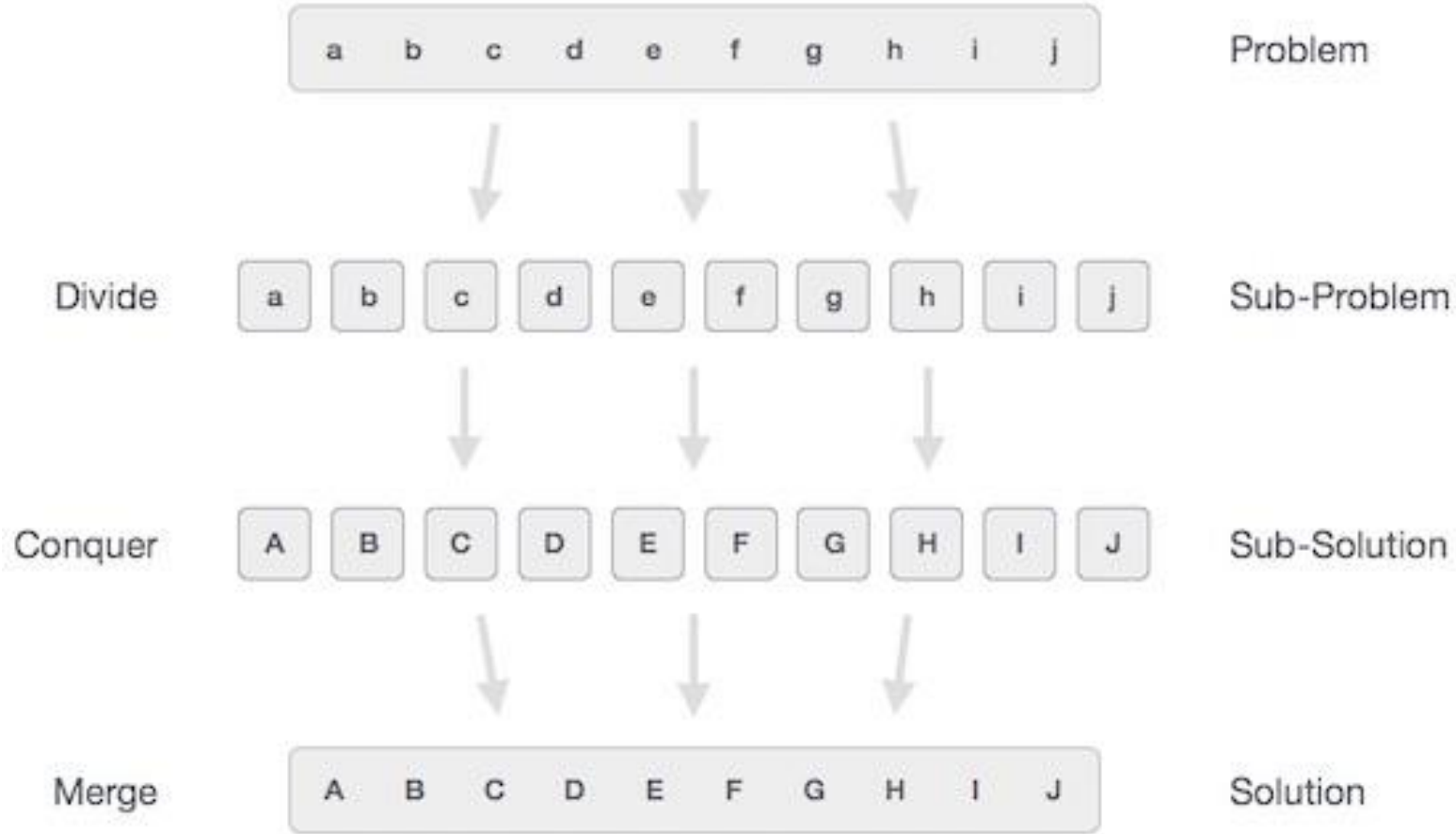
Parallel Algorithms cont…
Design Technique (Divide and Conquer)

- In the divide and conquer approach, the problem is divided into several small sub-problems.
- Then the sub-problems are solved recursively and combined to get the solution of the original problem.
- The divide and conquer approach involves the following steps at each level:

Parallel Algorithms cont…
Design Technique (Divide and Conquer)

- **Divide:** The original problem is divided into sub-problems.
- **Conquer:** The sub-problems are solved recursively.
- **Combine:** The solutions of the sub-problems are combined together to get the solution of the original problem.

Parallel Algorithms cont…
Design Technique (Divide and Conquer)



Parallel Algorithms cont…
Design Technique (Divide and Conquer)

- The divide and conquer approach is applied in the following algorithms:

1. Binary search
2. Quick sort
3. Merge sort
4. Matrix multiplication

Parallel Algorithms cont…
Design Technique (Greedy Algorithm)

- In greedy algorithm approach, solutions are made from the given solution domain.
- As being greedy, the ***closest solution that seems to provide an optimum solution*** is chosen.

Parallel Algorithms cont…
Design Technique (Greedy Algorithm)

- Greedy algorithms try to find a ***localized optimum*** solution, which may eventually lead to globally optimized solutions. However, generally greedy algorithms do not provide ***globally optimized*** solutions.

Parallel Algorithms cont…
Design Technique (Greedy Algorithm)

- The problem is to count to a ***desired value*** by choosing the ***least possible coins*** and the greedy approach forces the algorithm to pick the ***largest possible coin***.
- If we are provided coins of 1, 2, 5 and 10 and we are asked to count 18 then the greedy procedure will be:

Parallel Algorithms cont…
Design Technique (Greedy Algorithm)

1. Select one 10 coin, the remaining count is 8
2. Then select one 5 coin, the remaining count is 3
3. Then select one 2 coin, the remaining count is 1
4. And finally, the selection of one 1 coin solves the problem

Parallel Algorithms cont…
Design Technique (Greedy Algorithm)

- If we slightly change the problem then the same approach may not be able to produce the same optimum result.
- For the currency system, where we have coins of 1, 7, 10 value, counting coins for value 18 will be absolutely optimum but for count like 15, it may use more coins than necessary.

Parallel Algorithms cont…
Design Technique (Greedy Algorithm)

- For example, the greedy approach will use 10 + 1 + 1 + 1 + 1 + 1, total 6 coins.
- Whereas the same problem could be solved by using only 3 coins (7 + 7 + 1)
- Hence, we may conclude that the greedy approach picks an immediate optimized solution and may fail where global optimization is a major concern.

Parallel Algorithms cont…
Design Technique (Greedy Algorithm)

- A greedy algorithm always makes the choice that looks best at the moment.
  - Locally optimal choice leading to a globally optimal solution.
  - Greedy algorithms do not always yield optimal solutions
- Examples: Shortest path algorithm (Dijkstra), Minimum spanning tree(Kruskal and Prim)

Parallel Algorithms cont…
Design Technique (Dynamic Programming)

- Dynamic programming approach is similar to divide and conquer in breaking down the problem into smaller and yet smaller possible sub-problems.
- But unlike, divide and conquer, these sub-problems are not solved independently.

Parallel Algorithms cont…
Design Technique (Dynamic Programming)

- Rather, results of these smaller sub-problems are remembered and used for similar or overlapping sub-problems.
- Dynamic programming is used where we have problems, which can be divided into similar sub-problems, so that their results can be re-used. Mostly, these algorithms are used for optimization.

Parallel Algorithms cont…
Design Technique (Dynamic Programming)

- Before solving the in-hand sub-problem, dynamic algorithm will try to examine the results of the previously solved sub-problems.
- The solutions of sub-problems are combined in order to achieve the best solution.

Parallel Algorithms cont...
Design Technique (Dynamic Programming)

Dynamic Programming Example

- Let's find the **Fibonacci sequence** upto 5th term. A **Fibonacci series** is the sequence of numbers in which each number is the sum of the two preceding ones. For example, 0,1,1, 2, 3. Here, each number is the sum of the two preceding numbers.

Algorithm

- Let n be the number of terms.

1. If n <= 1, return 1.
2. Else, return the sum of two preceding numbers.

Parallel Algorithms cont…
Design Technique (Dynamic Programming)

Dynamic Programming Example

- We are calculating the **Fibonacci sequence** up to the 5th term.
    1. The first term is 0.
    2. The second term is 1.
    3. The third term is sum of 0 (from step 1) and 1(from step 2), which is 1.
    4. The fourth term is the sum of the third term (from step 3) and second term (from step 2) i.e. 1 + 1 = 2.
    5. The fifth term is the sum of the fourth term (from step 4) and third term (from step 3) i.e. 2 + 1 = 3.

Parallel Algorithms cont…
Design Technique (Dynamic Programming)

Dynamic Programming Example

Hence, we have the sequence 0, 1, 1, 2, 3. Here, we have used the results of the previous steps as shown below. This is called a dynamic programming approach.

F(0) = 0

F(1) = 1

F(2) = F(1) + F(0)

F(3) = F(2) + F(1)

F(4) = F(3) + F(2)

Parallel Algorithms cont…
Design Technique (Backtracking)

- Backtracking is an optimization technique to solve combinational problems.
- In backtracking, we start with a possible solution, which satisfies all the required conditions.
- Then we move to the next level and if that level does not produce a satisfactory solution, we return one level back and start with a new option.

Parallel Algorithms cont…
Design Technique (Backtracking)

- A **backtracking algorithm** is a problem-solving algorithm that uses a **brute force approach** for finding the desired output.

  The **Brute force approach** tries out all the possible solutions and chooses the desired/best solutions.

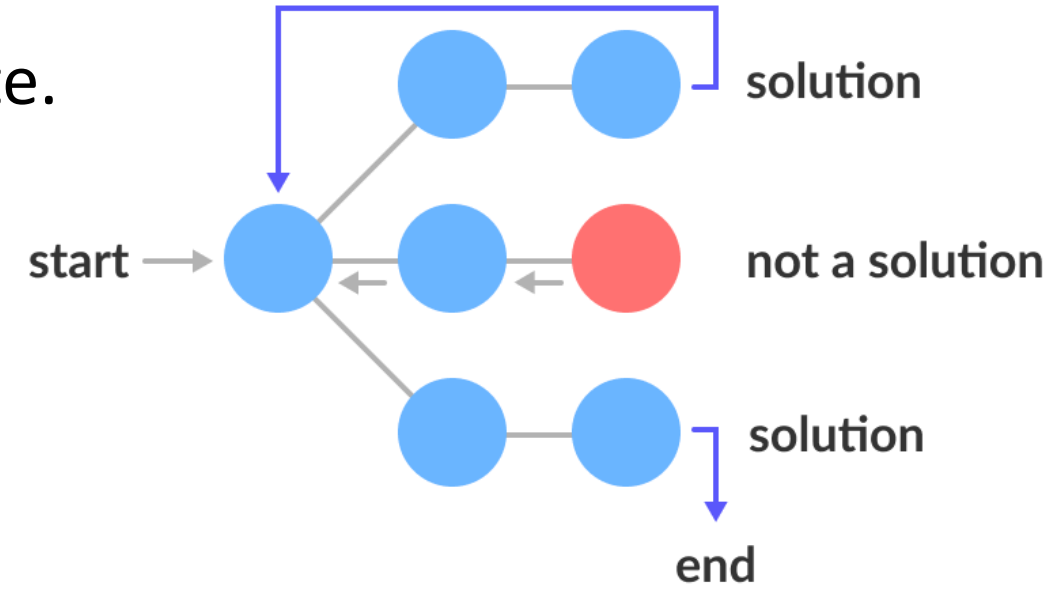
- The term backtracking suggests that if the current solution is not suitable, then backtrack and try other solutions. Thus, recursion is used in this approach.

- This approach is used to solve problems that have multiple solutions. If you want an optimal solution, you must go for dynamic programming.

Parallel Algorithms cont…
Design Technique (Backtracking)

State Space Tree

A space state tree is a tree representing all the possible states (solution or nonsolution) of the problem from the root as an initial state to the leaf as a terminal state.



Parallel Algorithms cont…
Design Technique (Backtracking)

Backtracking Algorithm

```
Backtrack(x)
    if x is not a solution
        return false
    if x is a new solution
        add to list of solutions
    backtrack(expand x)
```

Parallel Algorithms cont…
Design Technique (Backtracking)

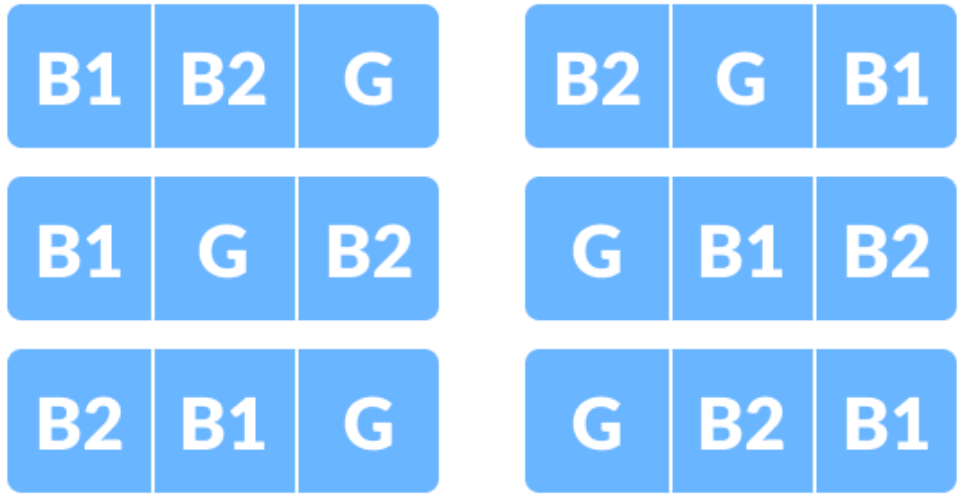
Example of Backtracking Approach

Problem: You want to find all the possible ways of arranging 2 boys and 1 girl on 3 benches.

Constraint: Girl should not be on the middle bench.

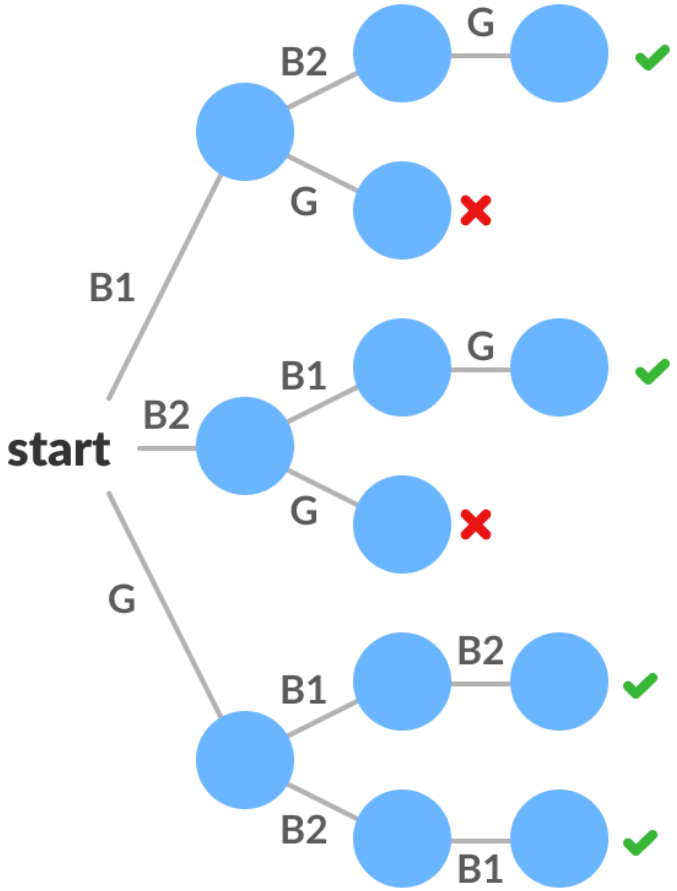
Solution: There are a total of 3! = 6 possibilities. We will try all the possibilities and get the possible solutions. We recursively try all the possibilities.

All the possibilities are:



Parallel Algorithms cont…
Design Technique (Backtracking)

The following state space tree shows the possible solutions.



Parallel Algorithms cont…
Design Technique (Branch and Bound)

- Branch and bound is an algorithm design paradigm which is generally used for solving combinatorial optimization problems.
- These problems are typically exponential in terms of time complexity and may require exploring all possible permutations in worst case.
- The Branch and Bound Algorithm technique solves these problems relatively quickly.

Parallel Algorithms cont…
Design Technique (Branch and Bound)

- A branch and bound algorithm is an optimization technique to get an optimal solution to the problem.
- It looks for the best solution for a given problem in the entire space of the solution.
- The bounds in the function to be optimized are merged with the value of the latest best solution.

Parallel Algorithms cont…
Design Technique (Branch and Bound)

- It allows the algorithm to find parts of the solution space completely.
- Once a solution is found, it can keep improving the solution.
- Branch and bound search is implemented in **depth-bounded search** and **depth–first search**.