**BackTracking**

Read wiki article : <https://en.wikipedia.org/wiki/Backtracking>

**Backtracking** is a general algorithm for finding all (or some) solutions to some computational prolems, notably [constraint satisfaction problems](https://en.wikipedia.org/wiki/Constraint_satisfaction_problem), that incrementally builds candidates to the solutions, and abandons each partial candidate *c* ("backtracks") as soon as it determines that *c* cannot possibly be completed to a valid solution. Backtracking can be applied only for problems which admit the concept of a "partial candidate solution" and a relatively quick test of whether it can possibly be completed to a valid solution. It is useless, for example, for locating a given value in an unordered table. When it is applicable, however, *backtracking is often much faster than*[*brute force enumeration*](https://en.wikipedia.org/wiki/Brute_force_search)*of all complete candidates, since it can eliminate a large number of candidates with a single test*. The backtracking algorithm enumerates a set of *partial candidates* that, in principle, could be *completed* in various ways to give all the possible solutions to the given problem. The completion is done incrementally, by a sequence of *candidate extension steps.* For example, in case of maze related problems, instead of generating complete maze states from scratch and applying acceptability test as in brute force, backtracking technique generates states partially and applies constraints/acceptibilty and thus saves unnecessary computations by not generating complete states which are sure to be rejected in the end , so it’s a bit predictive of foreseeing in nature.

**To be more precise:**

Conceptually, the partial candidates are represented as the nodes of a [tree structure](https://en.wikipedia.org/wiki/Tree_structure), the *potential search tree.* Each partial candidate is the parent of the candidates that differ from it by a single extension step; the leaves of the tree are the partial candidates that cannot be extended any further.

The backtracking algorithm traverses this search tree [recursively](https://en.wikipedia.org/wiki/Recursion_(computer_science)), from the root down, in [depth-first order](https://en.wikipedia.org/wiki/Depth-first_search). At each node *c*, the algorithm checks whether *c* can be completed to a valid solution. If it cannot, the whole sub-tree rooted at *c* is skipped (*pruned*). Otherwise, the algorithm (1) checks whether *c* itself is a valid solution, and if so reports it to the user; and (2) recursively enumerates all sub-trees of *c*. The two tests and the children of each node are defined by user-given procedures.

Therefore, the *actual search tree* that is traversed by the algorithm is only a part of the potential tree. The total cost of the algorithm is the number of nodes of the actual tree times the cost of obtaining and processing each node. This fact should be considered when choosing the potential search tree and implementing the pruning test.