Breadth First Search:

Implementation:
Using queue as data structure

For the current state, all actions are matched with their precondition list, if satisfied the new state is pushed onto the queue.

A* heuristic:

Implementation:
Using priority queue as data structure with help of heapq in python

Calculation of Heuristic:
We relaxed the original block world problem to get the new state by:

1. When we get new states, delete lists are ignored.

2. If a state is expanded, all possible actions are applied to it, counting the total number of actions required to reach the goal.

This heuristic is calculated in a Breadth First manner for each of the actions until a state has sentences of the goal state. The heuristic assigned is actually, the total number of actions to reach the goal state.

This heuristic is inadmissible and thus may not result in a optimal solution.

Goal Stack Planning:

Implementation:
Using stack to push sentences to make them true and a random ordering of sentences is used to add to stack.
As sentences are added to the stack randomly,the plan length may vary according to the order on which sentences are added.

Deciding which action to take next is defined specific to this problem by seeing the preconditions it satisfies.

Observations:

- When,
  Number of blocks: 4
  Initial state:(ontable 1) (ontable 2) (clear 2) (clear 4) (on 3 1) (on 4 3) (empty)
  Goal state: (ontable 3) (clear 2) (on 4 3) (on 1 4) (on 2 1) (empty)

	Plan length	**Number of nodes Expanded**	**Time taken**
BFS	10	11704	6.1507 s
Astar	10	90	0.6197 s
Goal Stack Planning	20 or 24 (depends on order of pushing conjucts)	N.A.	0.0027 s

- When,
  Number of blocks: 5
  Initial state:(ontable 5) (clear 2) (on 1 3) (on 2 1) (on 3 4) (on 4 5) (empty)
  Goal state: (ontable 5) (ontable 4) (clear 3) (clear 4) (on 1 5) (on 2 1) (on 3 2) (empty)

	Plan length	**Number of nodes Expanded**	**Time taken**
BFS	14	155008	140.1002 s
Astar	14	2616	31.4432 s
Goal Stack Planning	22 or 26 (depends on order of pushing conjucts)	N.A.	0.0032 s

From above tables, it could be clearly seen that,

- **BFS expands more number of nodes than Astar**. This is because astar uses a heuristic in addition to actual path cost.

Also, astar solves the relaxed problem to reach to goal state thus requiring less number of nodes to expand.

- As more number of nodes are expanded in BFS, the **time taken to reach to the goal state is also maximum in case of BFS**.

  While time taken to solve astar is approximately one fourth to that of time taken by BFS, despite the extra time taken to calculate the heuristic.

- The increase in number of expanded nodes by increasing one block is huge in case of BFS.
  Only with 5 blocks BFS expands close to 1 lakh nodes. So with larger number of blocks, given the memory constraints of CPU, it may not be able to provide a solution.

- Despite astar, less number of nodes expanded and less time taken, the plan length of both astar and bfs are same.
  Even, **the series of actions to reach the goal state are the same in the above cases**.

- The heuristic defined for astar is not a admissible heuristic. Thus, it may not give an optimal solution.

- Goal stack planning, takes milliseconds to reach to the goal state and gets the plan in least duration of time, if the solution exists.

- **Plan length of goal stack planning is greater than astar or BFS**. It clearly denotes, that GSP is “not optimal” and there could be extra work that needs to be done depending on the order in which we push the sentences onto the stack.

  The plan length of GSP also varies, as sentences are pushed randomly onto the stack.

- Goal stack planning implemented, may not find a solution to all the problems and is “semi decidable” in nature.