Heuristic Search Techniques

Generate and Test

- 1. Generate a possible solution(i.e. path/state)
- 2. Test to see if this is actually a solution by comparing to a state/ a path
- 3. If a solution has been found, quit. Otherwise return to step 1

- Very simple strategy - just keep guessing.

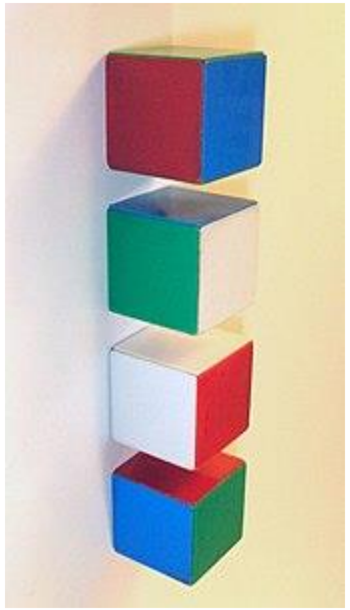
```
do while goal not accomplished
   generate a possible solution
   test solution to see if it is a goal
```

- Heuristics may be used to determine the specific rules for solution generation.

- Depth First Search – Complete Solution must be generated before tested
- Simply Exhaustive Search of the problem space.

$$\begin{tikzcd}
\text{Generation of solution} \arrow[r] \arrow[dd] & \overset{\text{(Basically DFS)}}{\text{Systematically}} \arrow[r] \arrow[d] & \text{Find a solution if one exists} \\
& \text{May take a very long time} & \\
\text{Randomly} \arrow[r] & \text{No guarantee of solution (British museum Algo .)} &
\end{tikzcd}$$

- Search process proceeds systematically, but some paths are not considered because they seem unlikely to lead to a solution. **This evaluation is performed by a heuristic function.**
- For simple problems, exhaustive generate-and-test is often a reasonable technique
- eg. Puzzle of four six-sided cubes, with each side of each cube painted one of four colors. arrangement of the cubes in a row such that on all four sides of the row one block face of each color is showing
- - solved using exhaustively
- - Using generate-and-test several configuration can be avoided.



- Infer Structure

Of organic

Compound using

Mass spectrogram and NMR

- **Observation: more red faces, then don't use red color for cube face initially**
- **Generate-and-test not useful much for harder application but when combined with other techniques to restrict the space in which to search even further, the techniques can be very effective**
- **eg. AI→DENDERAL→Plan-Generate-Test**

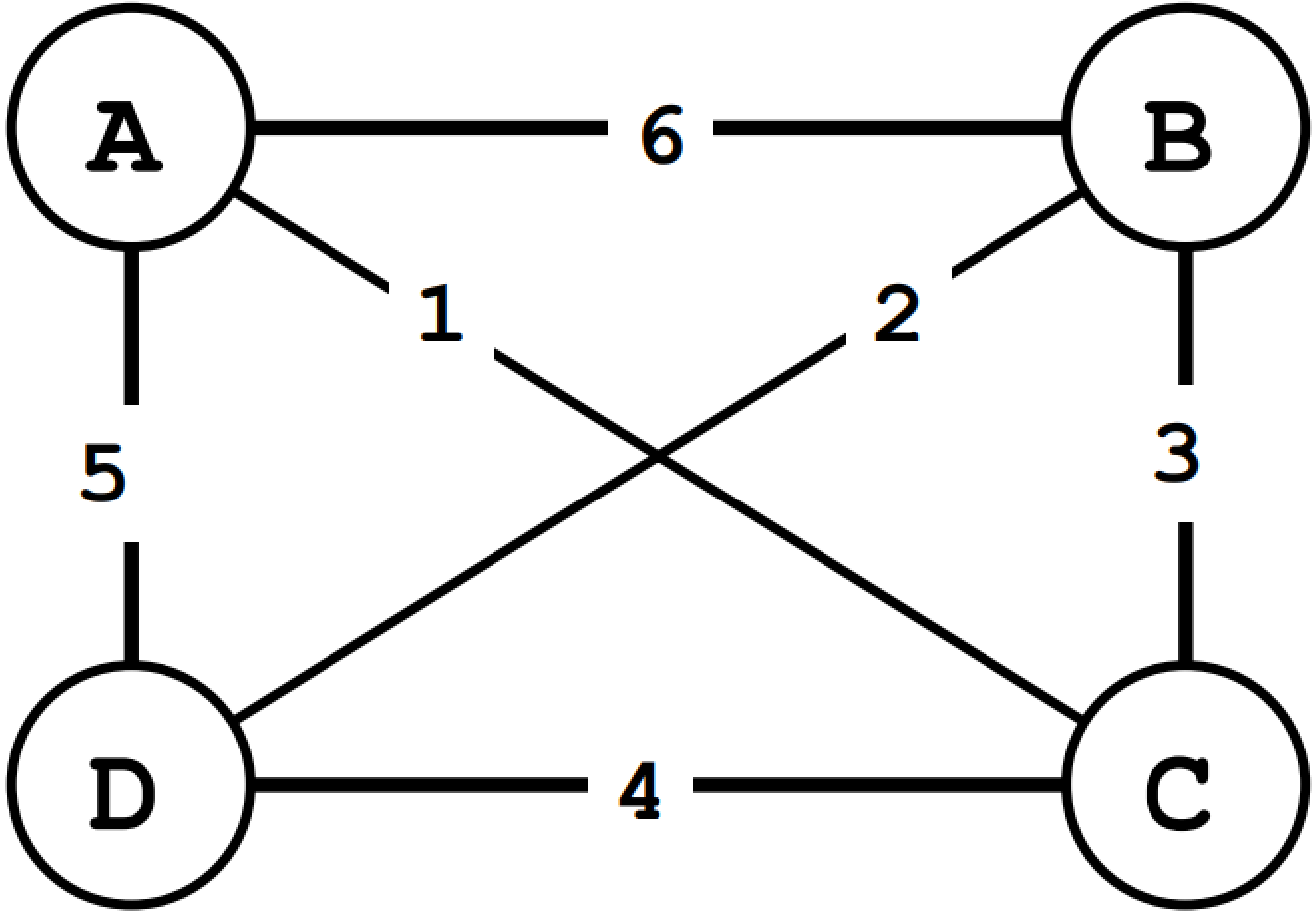
  ↓

  Used Constrain satisfaction technique
  Lists out recommended and contraindicated substructures
- **Limitation of planning: Produce somewhat inaccurate solutions as no feedback is available after plan**
- **Still planning is used to avoid trying unnecessary exploration and avoid combinatorial explosion**

Example - Traveling Salesman Problem (TSP)

- Traveler needs to visit *n* cities.
- Know the distance between each pair of cities.
- Want to know the shortest route that visits all the cities once.
- *n=80* will take millions of years to solve exhaustively!

TSP Example

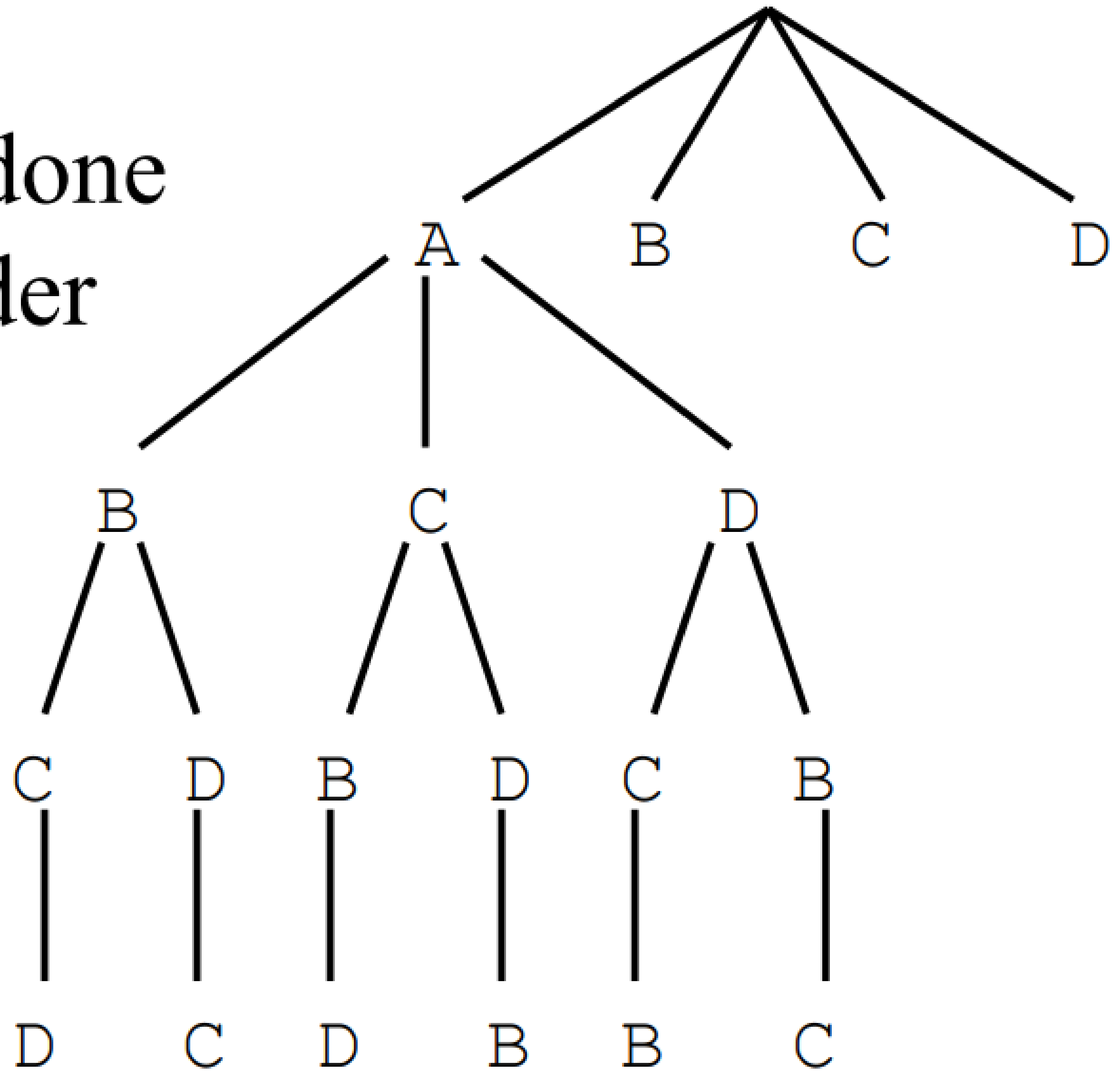


Generate-and-test Example

- TSP - generation of possible solutions is done in lexicographical order of cities:

  1. A – B – C – D
  2. A – B – D – C
  3. A – C – B – D
  4. A – C – D – B

  ...



Hill Climbing

- Variation on generate-and-test:
  - *generation* of next state depends on feedback from the *test* procedure.
  - *Test* now includes a heuristic function that provides a guess as to how good each possible state is.
- There are a number of ways to use the information returned by the *test* procedure.

$$\begin{tikzcd} \text{❑ Hill Climbing: test} \arrow[r, "\text{has}"] & \text{Heuristic function} \arrow[d] \\ & \text{Provides an estimation how close a given state is from goal state} \end{tikzcd}$$

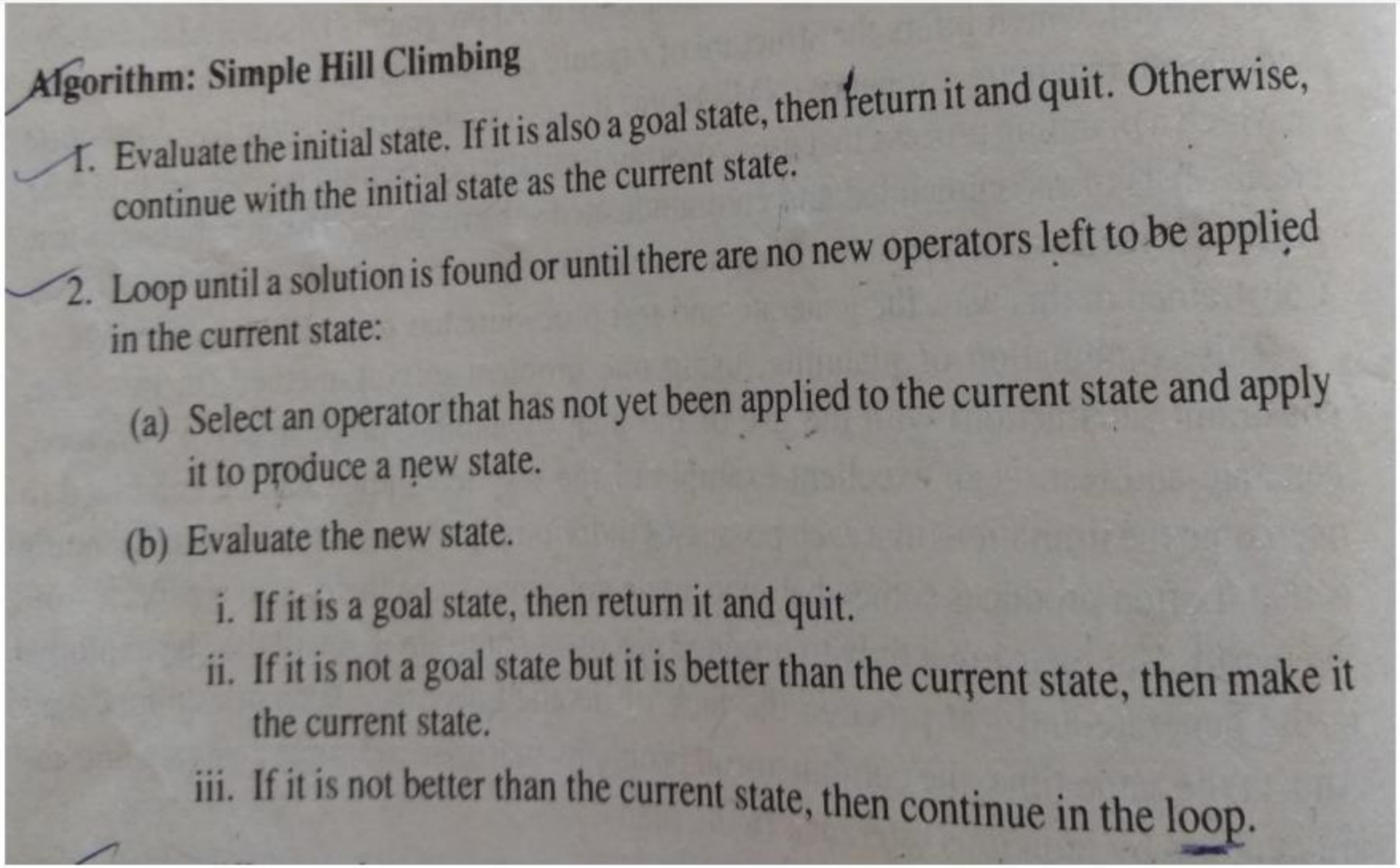
- **Hill Climbing is useful when good heuristic function is available for evaluating states but no other useful knowledge is available**

Simple Hill Climbing

- Use heuristic to move only to states that are *better* than the current state.

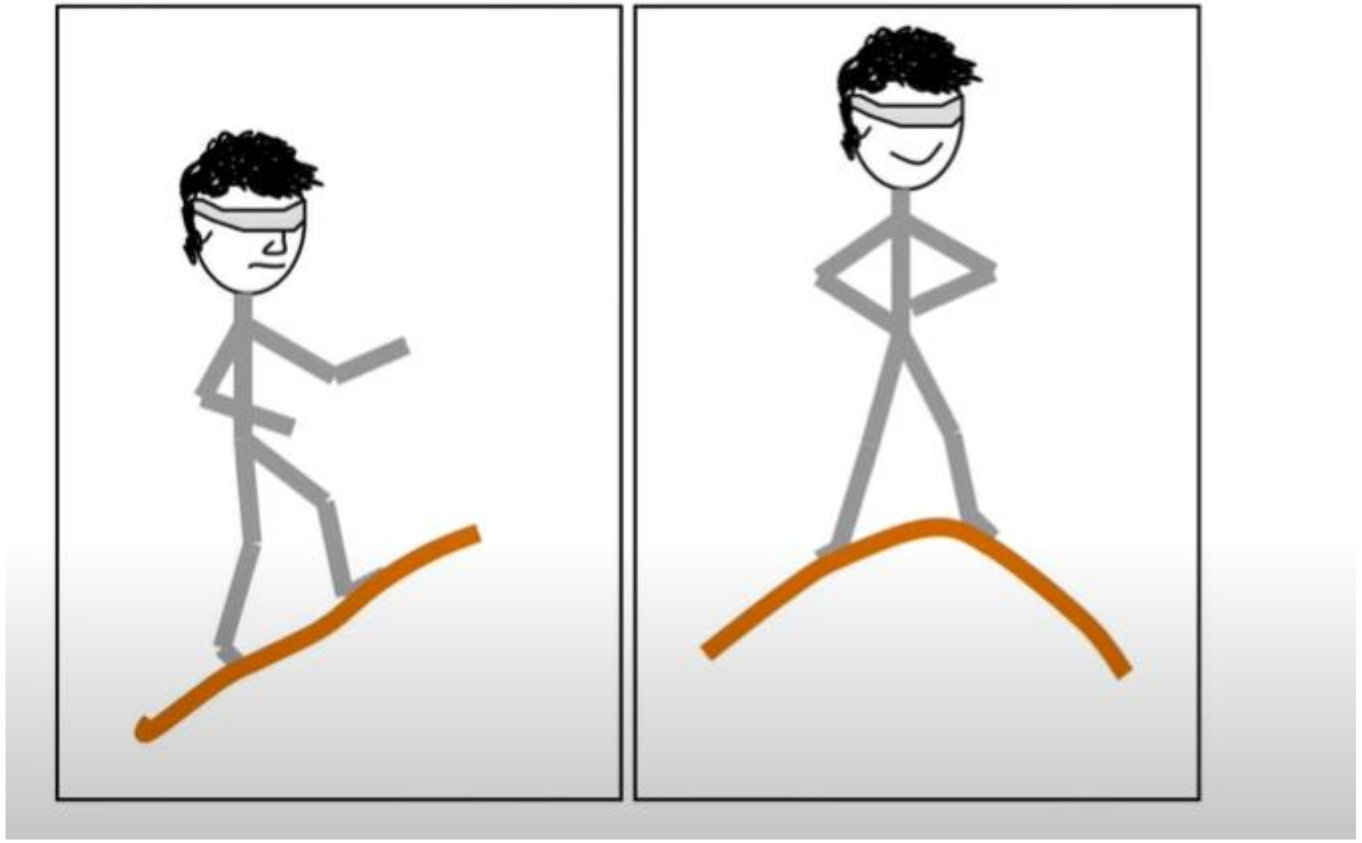
- Always move to better state when possible.

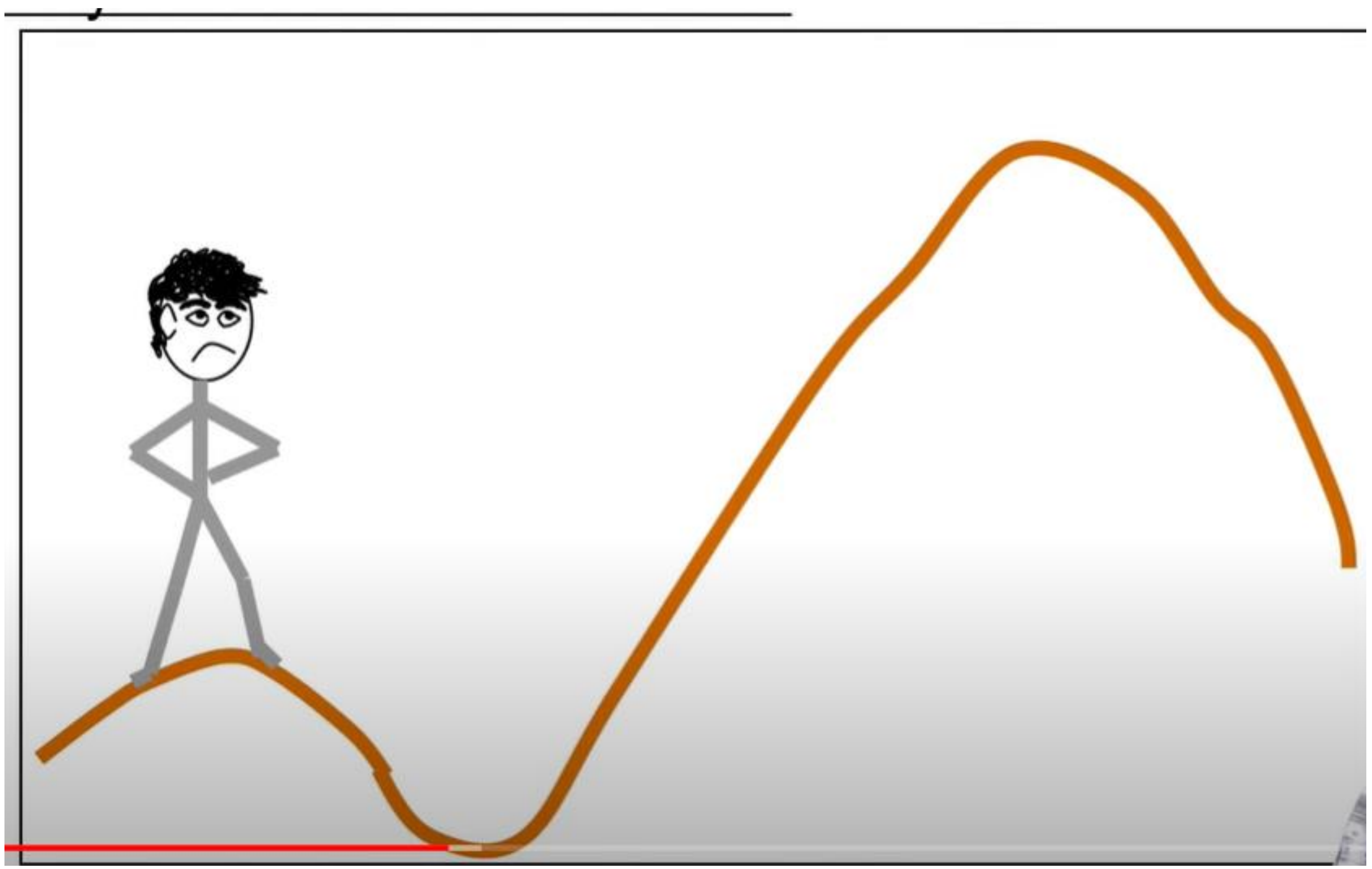
- The process ends when all operators have been applied and none of the resulting states are better than the current state.



Algorithm: Simple Hill Climbing

1. Evaluate the initial state. If it is also a goal state, then return it and quit. Otherwise, continue with the initial state as the current state.
2. Loop until a solution is found or until there are no new operators left to be applied in the current state:
   (a) Select an operator that has not yet been applied to the current state and apply it to produce a new state.
   (b) Evaluate the new state.
      i. If it is a goal state, then return it and quit.
      ii. If it is not a goal state but it is better than the current state, then make it the current state.
      iii. If it is not better than the current state, then continue in the loop.





Problem of Local Optimum

Potential Problems with Simple Hill Climbing

- Will terminate when at local optimum.
- The order of application of operators can make a big difference.
- Can’t see past a single move in the state space.

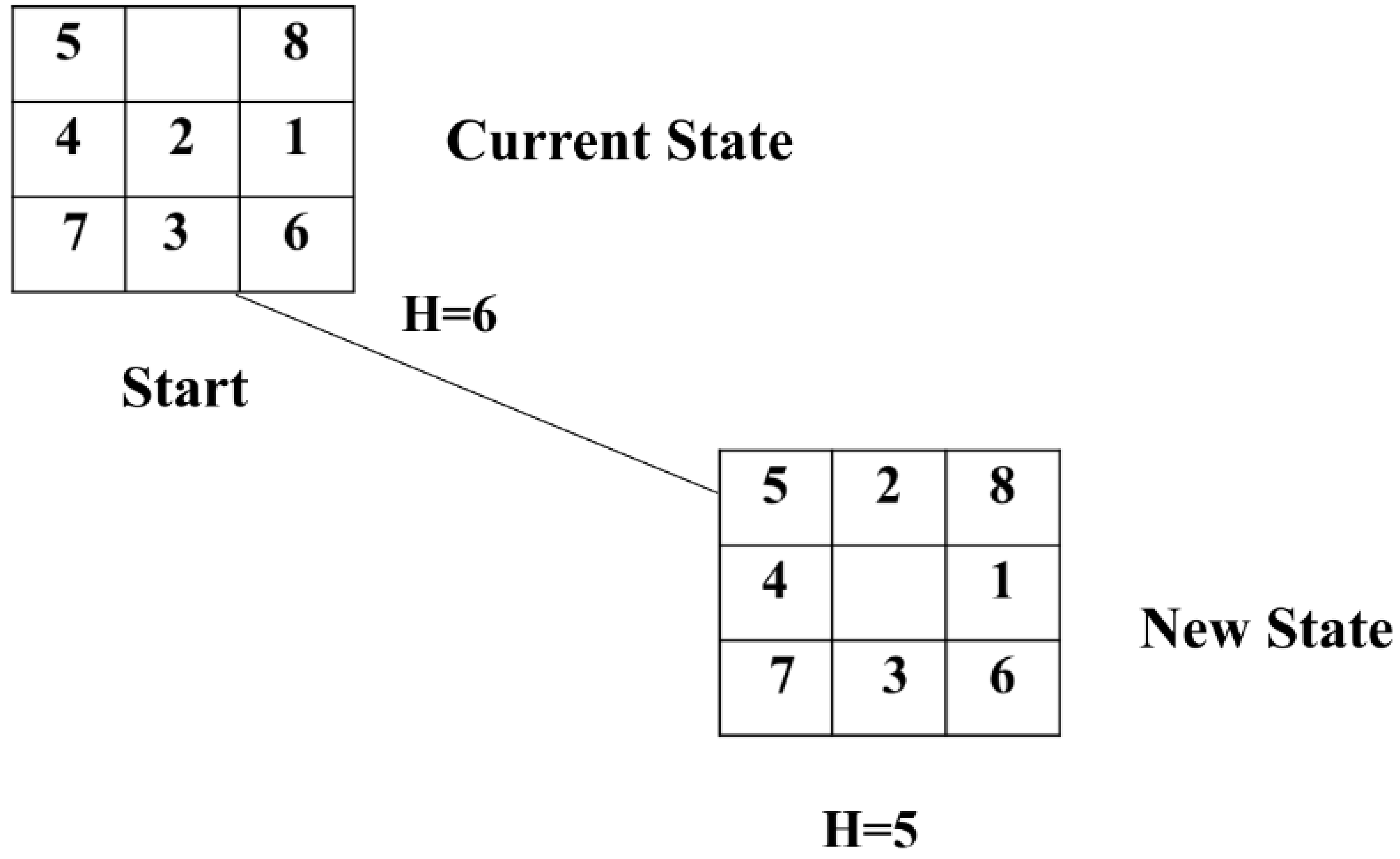
5		8
4	2	1
7	3	6

Start

1	2	3
4	5	6
7	8	

Goal

Heuristic1 = number of misplaced numbered tiles

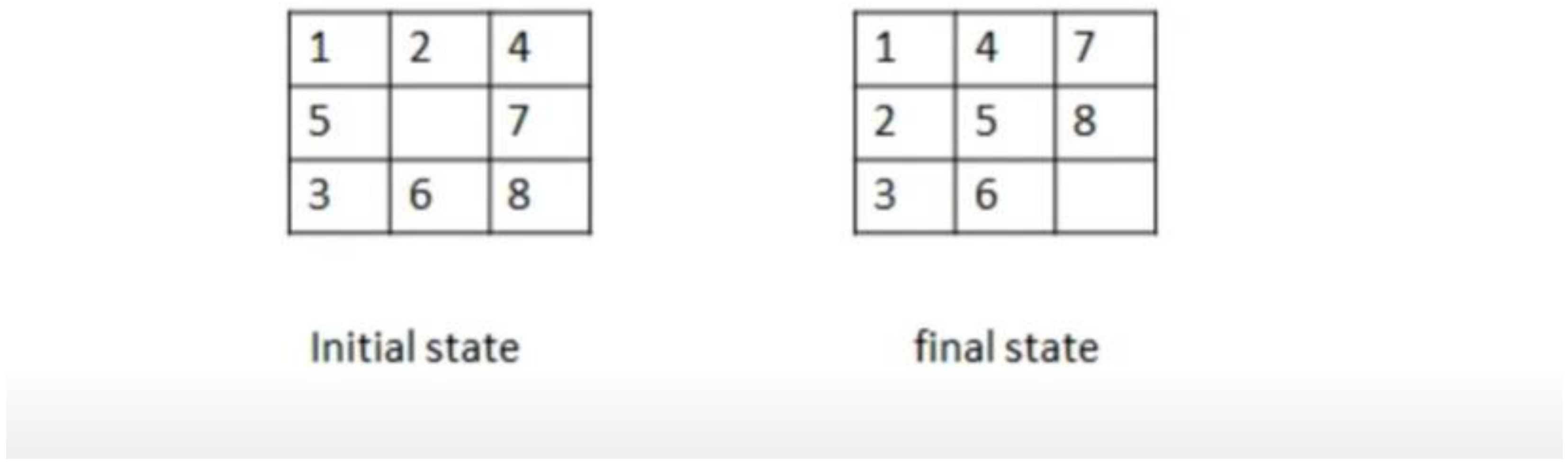


New State better than current state

.

- Difference between hill climbing and generate-and-test is the use of evaluation function to inject task specific knowledge into the control process
- Knowledge gives power to solve some interactable problems
- heuristic function=evaluation function
- How to decide new state is better than current state?
  - Value returned by evaluation function
  - It can be higher the better/lower the better

Example:



1	2	4
5		7
3	6	8

Initial state

1	4	7
2	5	8
3	6	

final state

Chapter3

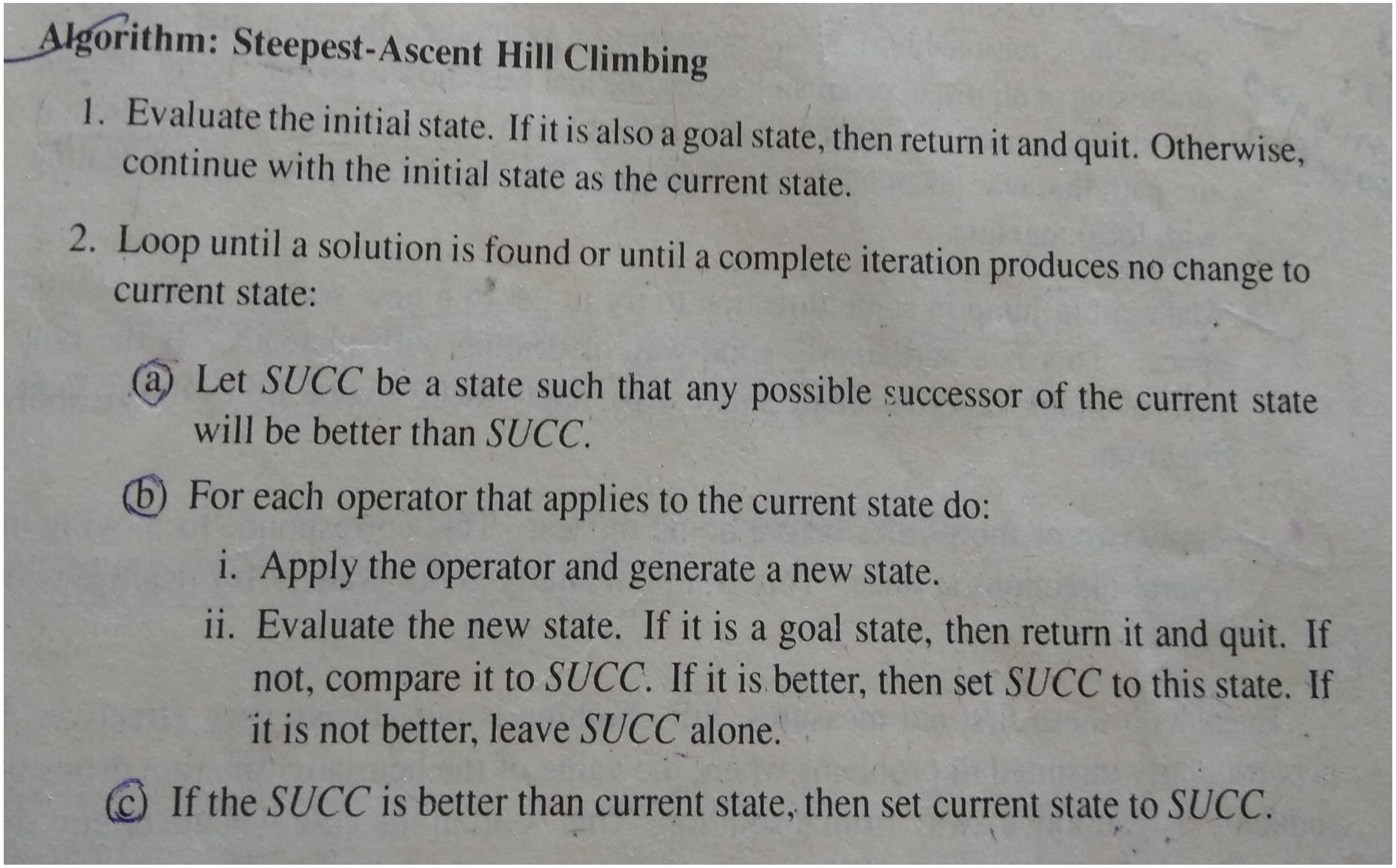
Heuristic Search Techniques

(Cont….)

Steepest-Ascent Hill Climbing (Gradient search)

- Variation on simple hill climbing
- Considers all the moves from the current state and selects the best one as the next state
- Contrasts with the basic method in which the first state that is better than the current state is selected

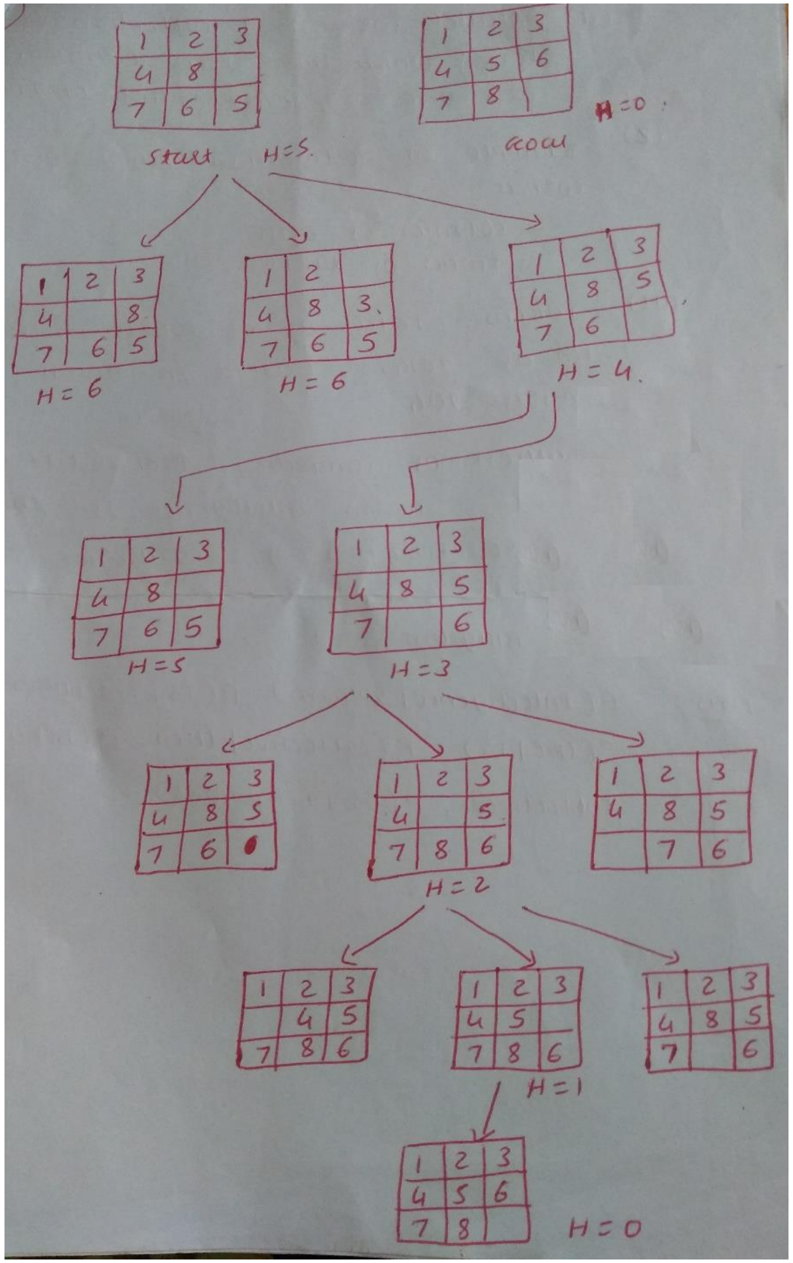
Steepest-Ascent Hill Climbing



Algorithm: Steepest-Ascent Hill Climbing

```
1. Evaluate the initial state. If it is also a goal state, then return it and quit. Otherwise,
   continue with the initial state as the current state.

2. Loop until a solution is found or until a complete iteration produces no change to
   current state:

   (a) Let SUCC be a state such that any possible successor of the current state
       will be better than SUCC.

   (b) For each operator that applies to the current state do:

       i.  Apply the operator and generate a new state.
       ii. Evaluate the new state. If it is a goal state, then return it and quit. If
           not, compare it to SUCC. If it is better, then set SUCC to this state. If
           it is not better, leave SUCC alone.

   (c) If the SUCC is better than current state, then set current state to SUCC.
```

Steepest-Ascent Hill Climbing



Steepest-Ascent Hill Climbing

Heuristic = sum of the (Manhattan) distance of every numbered tile to its goal position

= 0 + 0 + 0 + 0 + 1 + 0 + 2 + 2

Start = 5

Steepest-Ascent Hill Climbing

- Simple Hill Climbing: number of moves required to get to a solution is longer

$\updownarrow$ trade-off

- Steepest-Ascent Hill Climbing: Time required to select a move is longer

- Both basic and steepest-ascent hill climbing may fail to find a solution, either algorithm may terminate not by finding a goal state but by getting to a state from which no better states can be generated

- This will happen if the program has reached either a <u>local maximum, a plateau or a ridge</u>

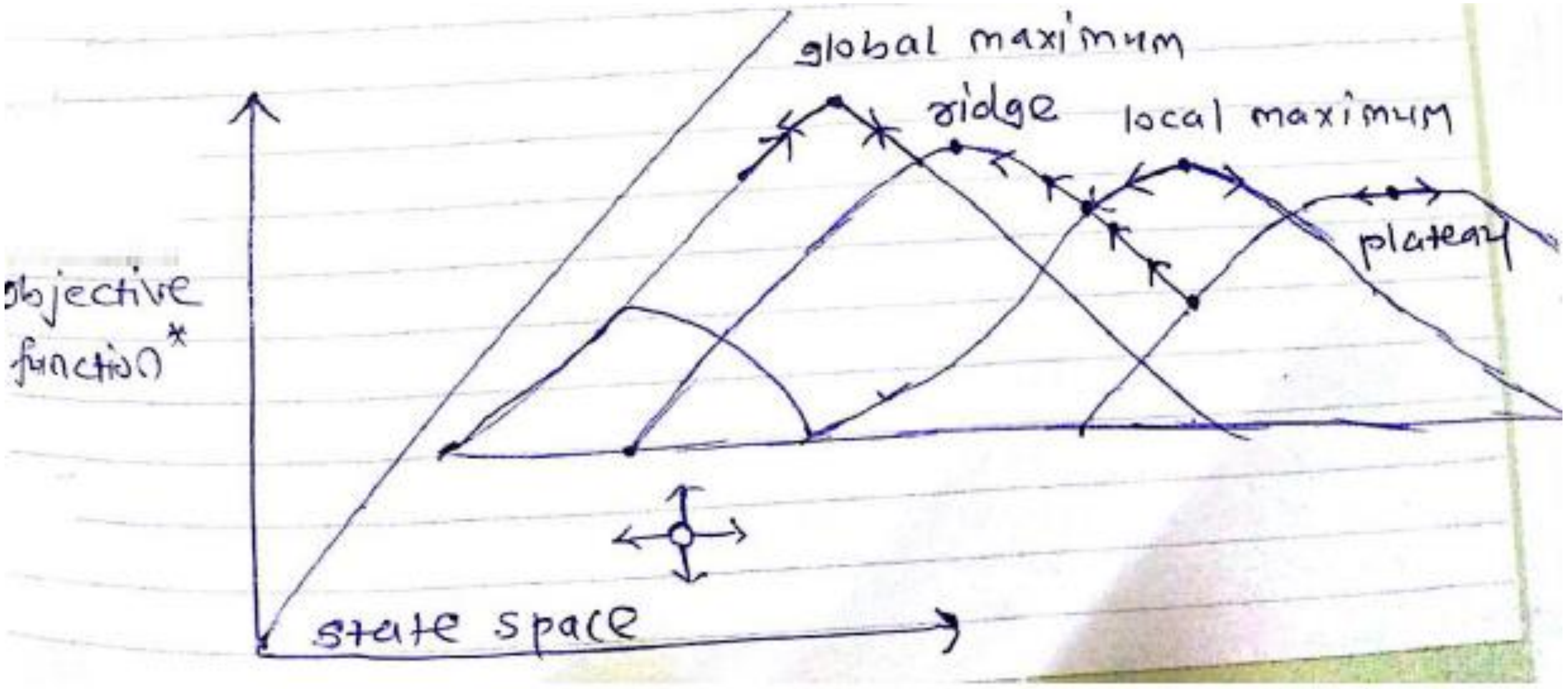
Disadvantage of Hill Climbing

- **Local maximum:**
  - **Is a state that is better than all its neighbours but is not better than some other states farther away**
- **Plateau:**
  - **Is a flat area of search space in which whole set of neighboring states have the same value**
  - **Not possible to determine the best direction in which to move by making local comparisons**

Disadvantage of Hill Climbing

- **Ridge:**
  - **Special kind of local maximum**
  - **It is an area of the search space that is higher than surrounding areas and that itself has a slope (Which one would like to climb). But the orientation of the high region, compared to the set of available moves and the directions in which they move, makes it impossible to traverse a ridge by single moves**

Disadvantage of Hill Climbing



Ways to solve Hill Climbing problems

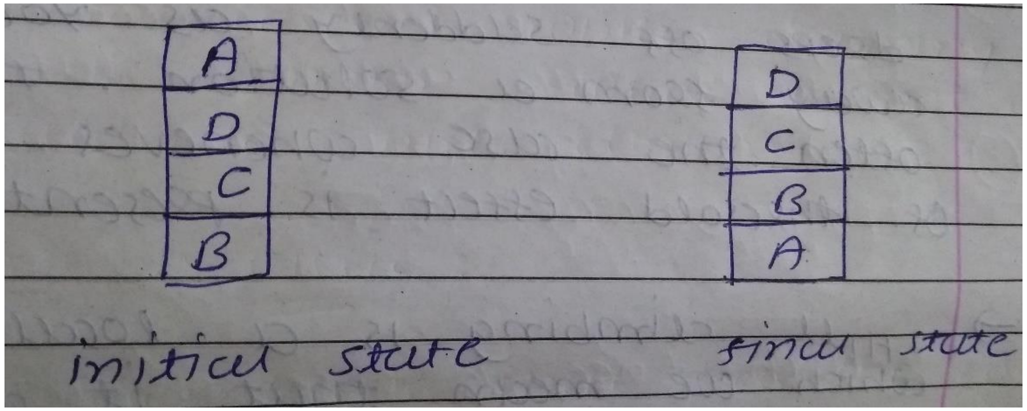
- **Local maximum:**
  - **Backtrack to some earlier node and try going in a different direction**
- **Plateau:**
  - **Make a big jump in some direction to try to get to a new section of the search space**
  - **If the only rules available describe single small steps, apply them several times in the same direction**
- **Ridge:**
  - **Apply two or more rules before doing the test**
  - **This corresponds to moving in several directions at once**

- **Even with these first-aid measures, hill climbing is not always very effective**
- **It is particularly unsuited to problems where the value of heuristic function drops off suddenly as you move away from a solution. These is often the case whenever any sort of threshold effect is present**
- **Hill climbing is a local method by which we mean that it decides what to do next by looking only at the “immediate” consequences of its choice rather than by exhaustively exploring all the consequences**
- **Advantage of being local heuristic: less combinatorially explosive**
- **Disadvantage of being local heuristic: lack of a guarantee that it will be effective**

- How to make hill climbing use global information?

  - provide global information in heuristic function

Example: Blocks world problem



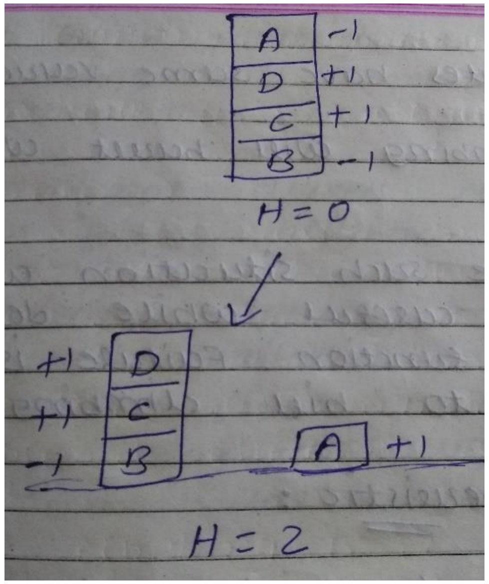
- **Local heuristic:**
  - **+1 for each block that is resting on the thing it is suppose to be resting on**
  - **-1 for each block that is resting on wrong thing**

Blocks world problem

Start state : (-1) + (+1) + (+1) + (-1) = 0

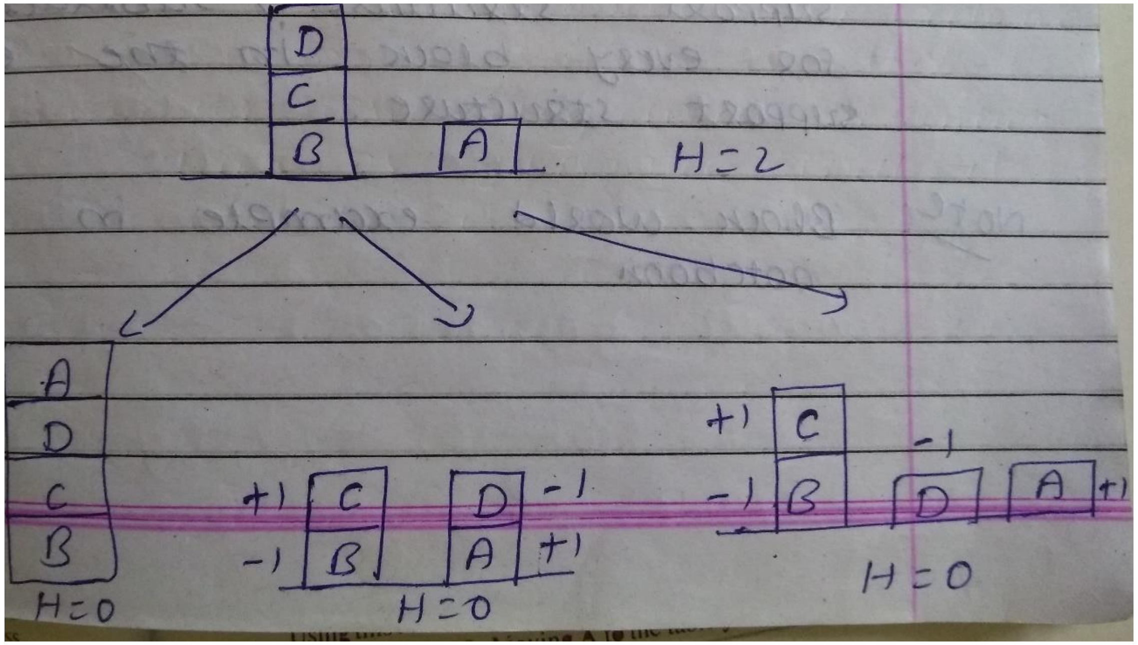
Goal state: (+1) + (+1) +(+1) + (+1) = 4

- **Apply operator to current state**



- **If new state is better than current state than new state becomes current state**

Blocks world problem

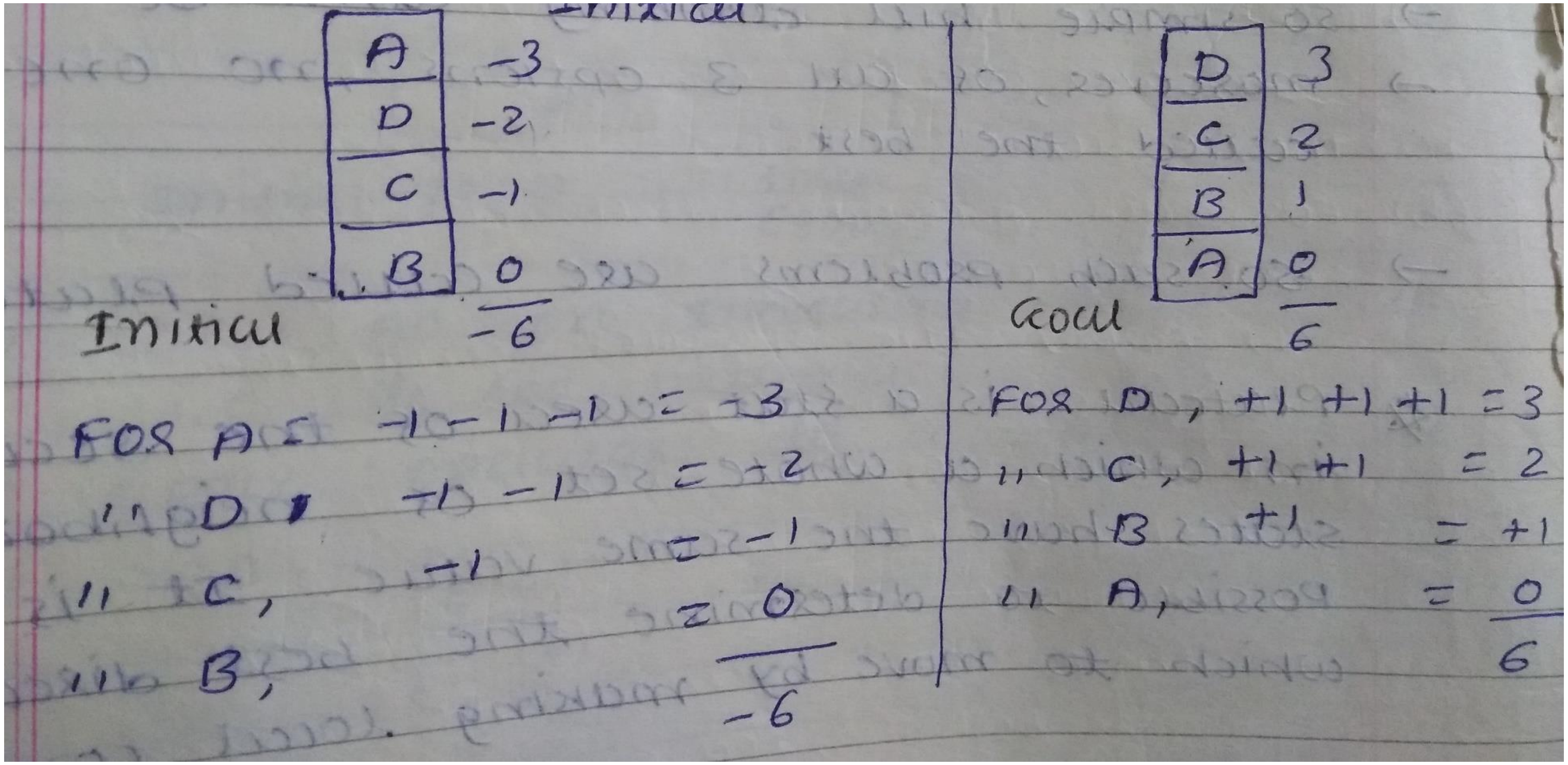


- ❑ **All the states have same value: Plateau**

Blocks world problem

- Global heuristic:
  - For each block that has the correct support structure(i.e. the complete structure underneath it is exactly as it should be) , add one point for every block in the support structure
  - For each block that has an incorrect support structure, subtract one point for every block in the existing support structure

Blocks world problem



A	-3
D	-2
C	-1
B	0
	-6

Initial

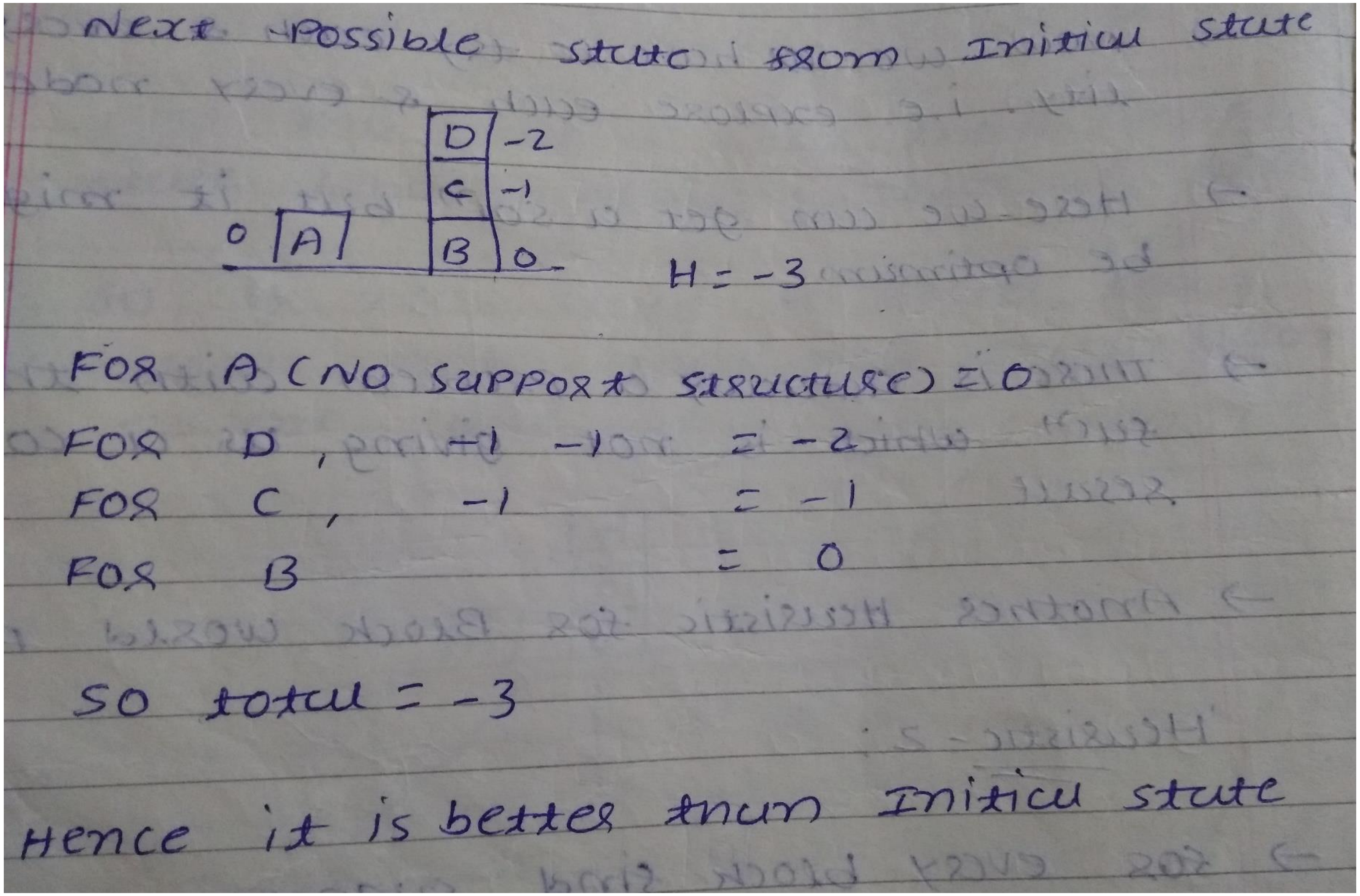
D	3
C	2
B	1
A	0
	6

Goal

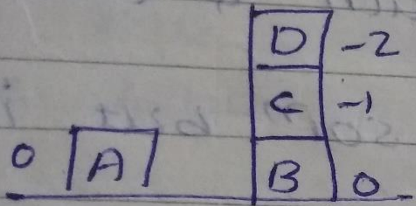
FOR A = -1 -1 -1 = -3
" D -1 -1 = -2
" C, -1 = -1
" B, = 0
$\overline{-6}$

FOR D, +1 +1 +1 = 3
" C, +1 +1 = 2
" B +1 = +1
" A, = 0
$\overline{6}$

Blocks world problem



Next possible state from Initial state



H = -3

For A (No support structure) = 0

For D , -1 -1 = -2

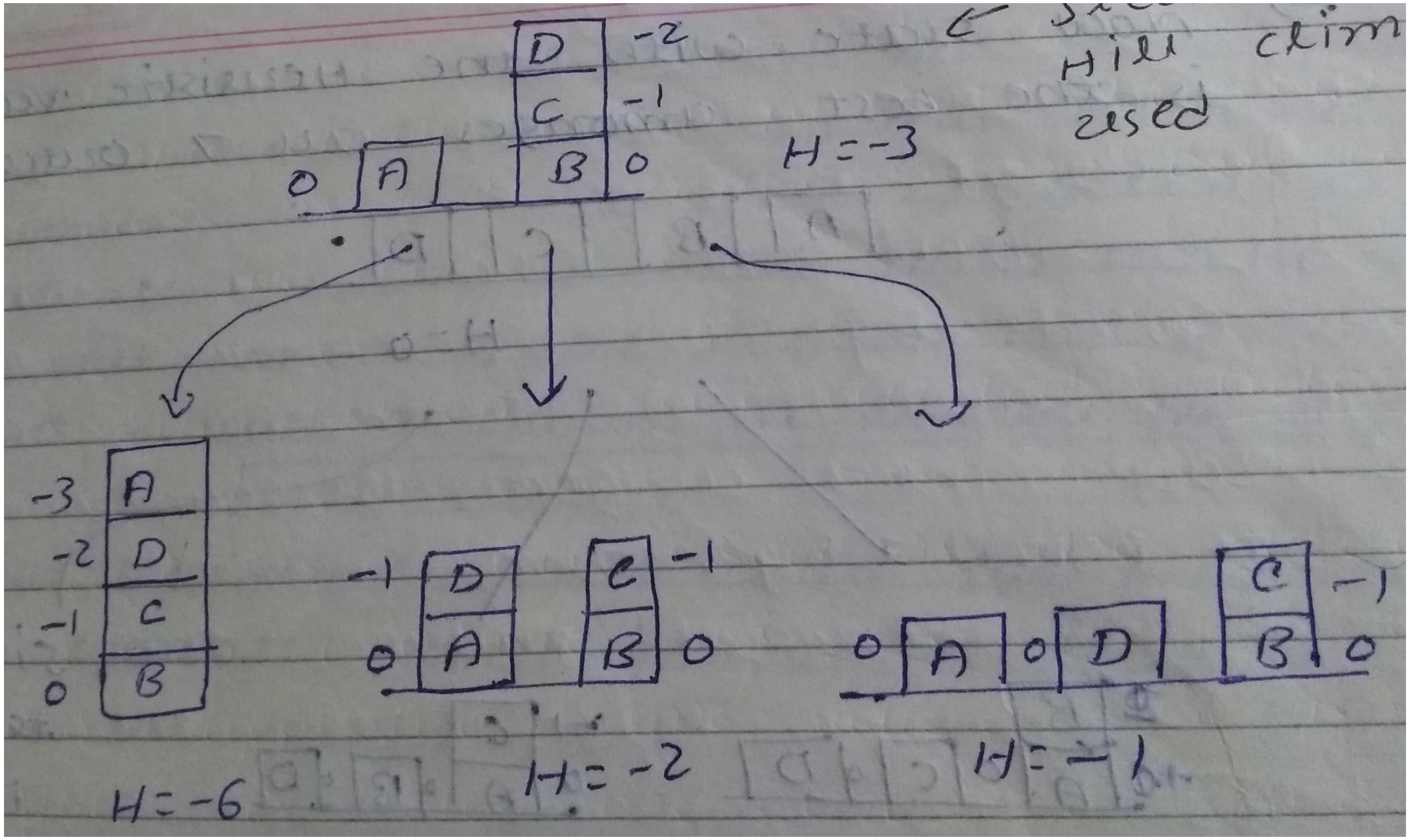
For C , -1 = -1

For B = 0

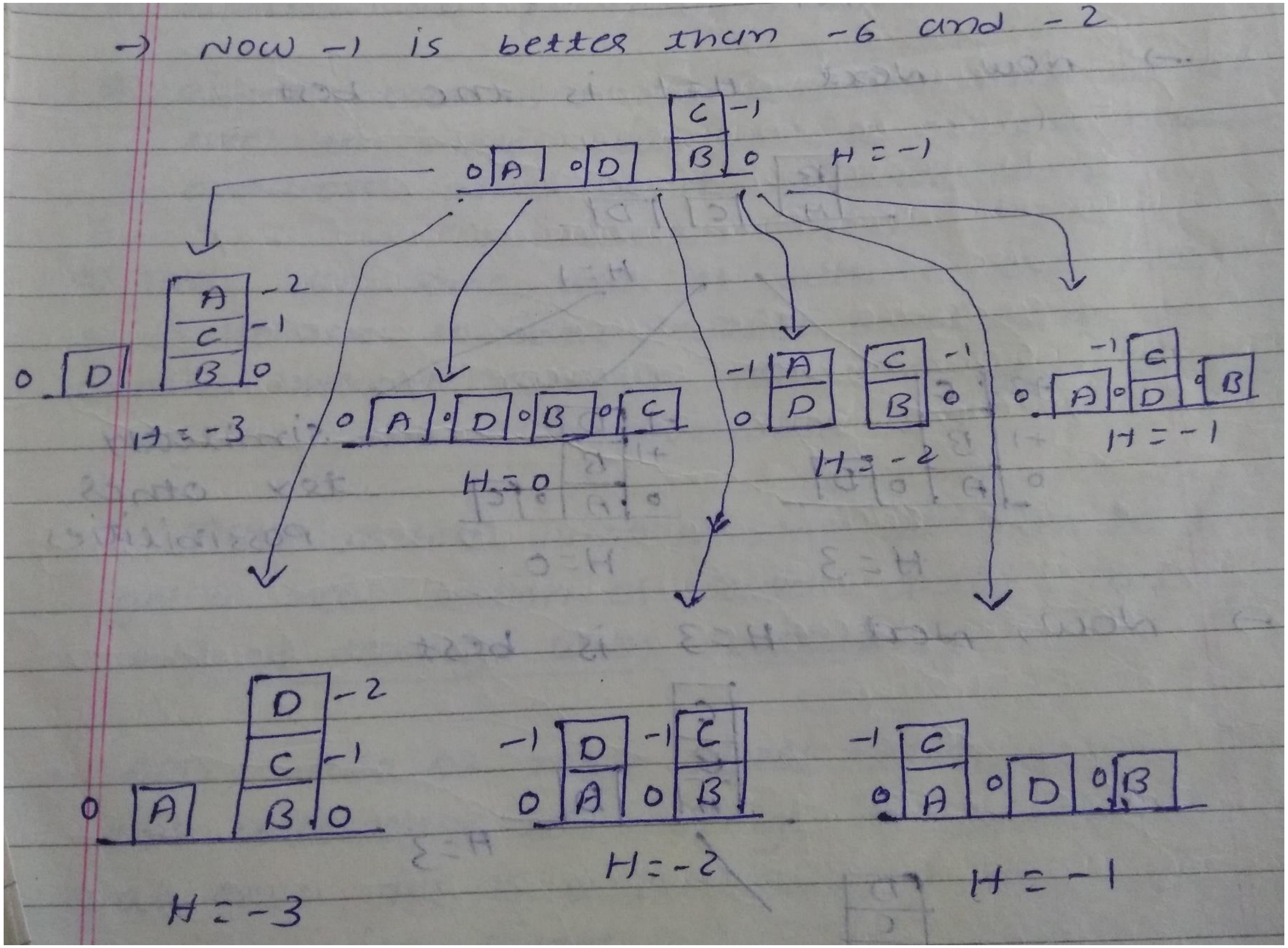
So total = -3

Hence it is better than Initial state

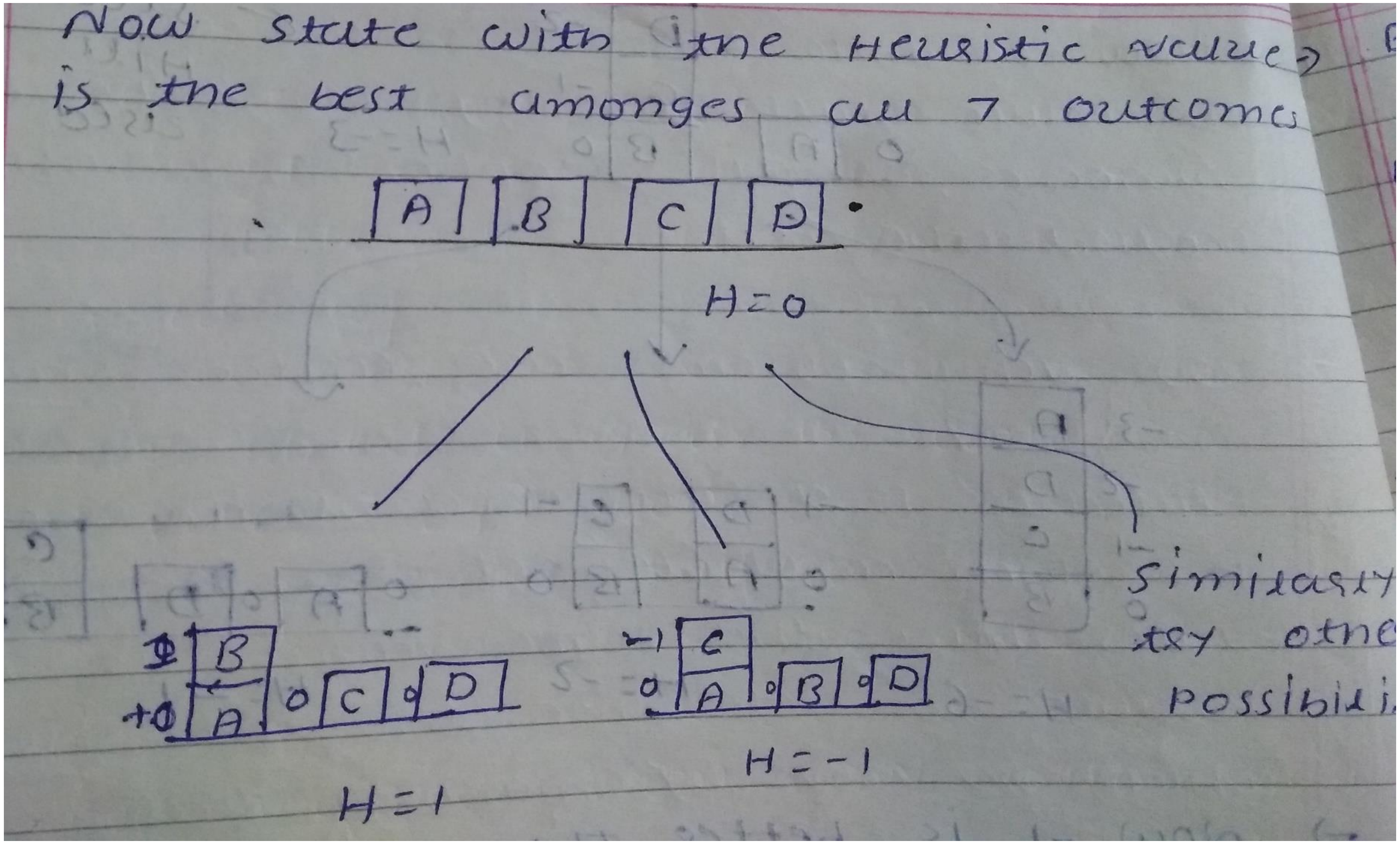
Blocks world problem



Blocks world problem



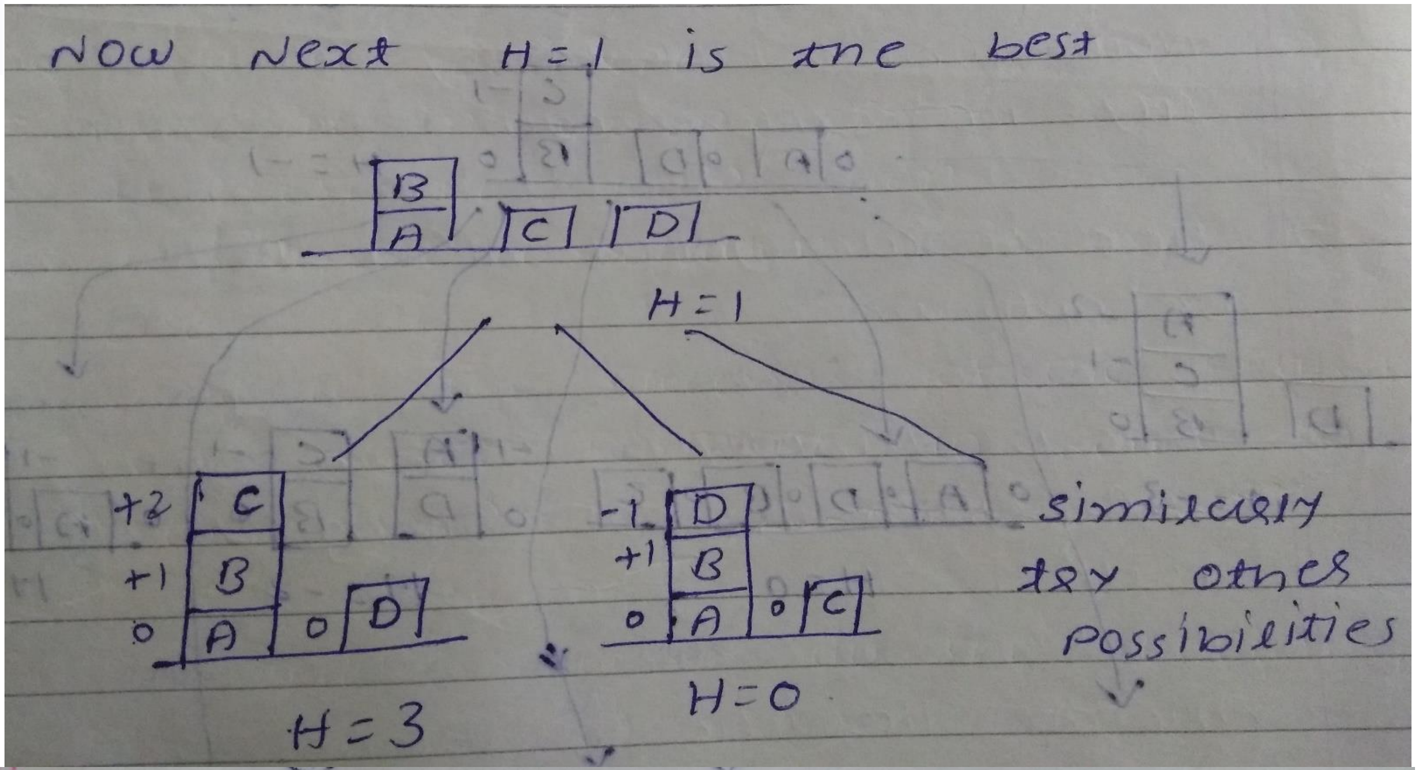
Blocks world problem



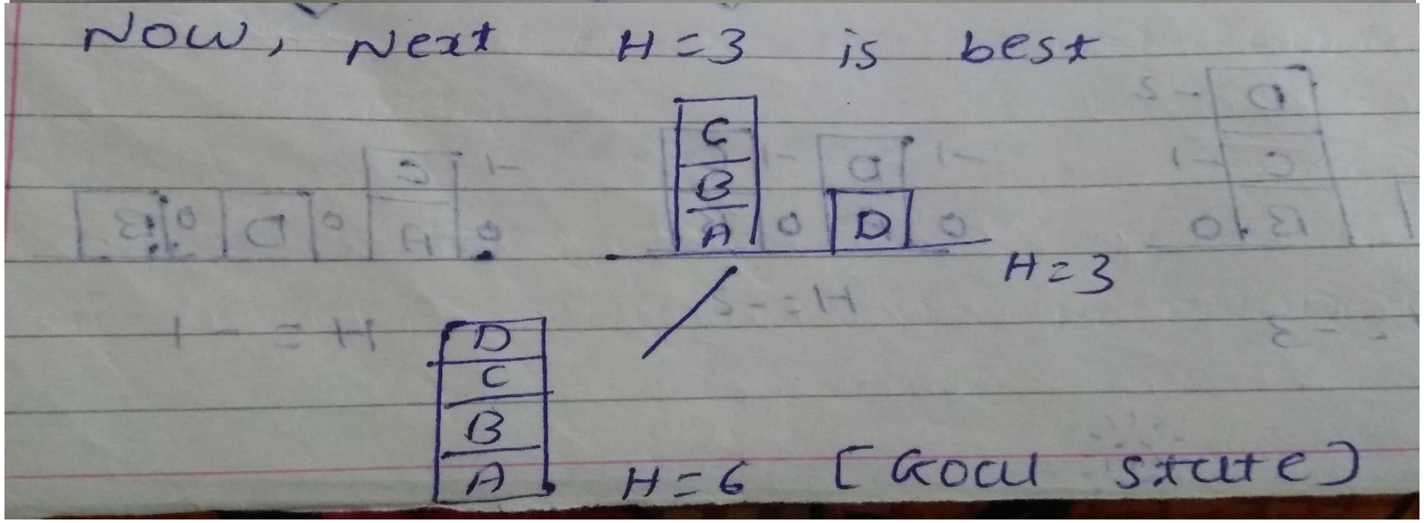
Now state with the Heuristic value → is the best amonges all 7 outcomes

A B C D
H=0
1 B
+1 A
0 C
0 D
H=1
-1 C
0 A
0 B
0 D
H=-1
similarly try othe possibili

Blocks world problem


Now Next H=1 is the best

similarly try others possibilities


Now, Next H=3 is best

H=6 [Goal state]