

# Offline 02: Hill Climbing and Simulated Annealing

Solve the 8-puzzle problem using **steepest ascent hill climbing algorithm** and **simulated annealing algorithm**.

- The successor of each state is found by moving the blank space to *Left, Right, Up* or *Down*.
- Implement the **#misplace-tiles** and **manhattan-distance** as the heuristic cost  $h(n)$  of a state  $n$ .
- Terminate the algorithm when the goal state is reached or after 1000 iterations.
- The memory requirement of your implementation must be  $O(1)$ .
- The schedule function of simulated annealing must be such that as  $t$  increases,  $T$  decreases and eventually becomes 0.
- A template code is available [here](#)
- Run hill climbing for sample inputs using **#misplace-tiles** and **manhattan-distance** and **log the results in [this report](#)**. Similarly, run hill climbing for sample input using **#misplace-tiles** and **manhattan-distance** and **log the results in the same report**.

0	1	2
3	4	5
6	7	8

Goal state (**Assume 0 means blank**)

Sample Input:

Sample Input	Sample output
3 1 2 6 4 5 0 7 8	Neighbor [[3, 1, 2], [0, 4, 5], [6, 7, 8]] $h=2$ Neighbor [[0, 1, 2], [3, 4, 5], [6, 7, 8]] $h=0$ Neighbor [[1, 0, 2], [3, 4, 5], [6, 7, 8]] $h=2$  solution [[0, 1, 2], [3, 4, 5], [6, 7, 8]] $h=0$ [Note that you need to follow the sample output format but the actual values may be different depending on the heuristic and the algorithm]
3 1 2 6 4 0 7 8 5	

## Instructions:

- Read the questions very carefully and answer all parts of the question.
- **The input will be given in input.txt file** and will be in the same folder as your code.
- Your code must be implemented for the given sample input format. Your output should also match the sample output format. Your code will be tested on other inputs not given in the sample input.
- **You will get -100% for adopting any unfair means.**
- **Your marks will fully depend on your viva and understanding.**
  - **Total 20 marks**
    - Heuristics = 2+4 marks
    - Hill climbing = 6 marks (implementation) + 2 marks (report)
    - Simulated annealing = 4 marks (implementation) + 2 marks (report)
- **Submit the .ipynb file**

# Pseudocodes

## Steepest ascent hill climbing

**function** HILL-CLIMBING(*problem*) **returns** a state that is a local maximum

*current*  $\leftarrow$  MAKE-NODE(*problem*.INITIAL-STATE)

**loop do**

*neighbor*  $\leftarrow$  a highest-valued successor of *current*

**if** *neighbor*.VALUE  $\leq$  *current*.VALUE **then return** *current*.STATE

*current*  $\leftarrow$  *neighbor*

- Memory requirement  $O(1)$
- If you use heuristic cost instead of heuristic value, then you should pick the lowest-cost successor as neighbor and stop when neighbor cost is higher than current cost.

## Simulated annealing

**function** SIMULATED-ANNEALING(*problem*, *schedule*) **returns** a solution state

**inputs:** *problem*, a problem

*schedule*, a mapping from time to “temperature”

*current*  $\leftarrow$  MAKE-NODE(*problem*.INITIAL-STATE)

**for**  $t = 1$  **to**  $\infty$  **do**

*T*  $\leftarrow$  *schedule*(*t*)

**if**  $T = 0$  **then return** *current*

*next*  $\leftarrow$  a randomly selected successor of *current*

$\Delta E \leftarrow$  *next*.VALUE - *current*.VALUE

**if**  $\Delta E > 0$  **then** *current*  $\leftarrow$  *next*

**else** *current*  $\leftarrow$  *next* only with probability  $e^{\Delta E/T}$

- Memory requirement  $O(1)$
- If you use heuristic cost instead of heuristic value, then  $\Delta E$  should be *current*.cost - *next*.cost.