Hill climbing search

A kind of Local search

Only goal state is important, how we reach the goal state from source is not important

→ path

local search not applicable for
- → solving 8 puzzle problem / Rubix cube
- → Getting shortest path to reach goal from source.

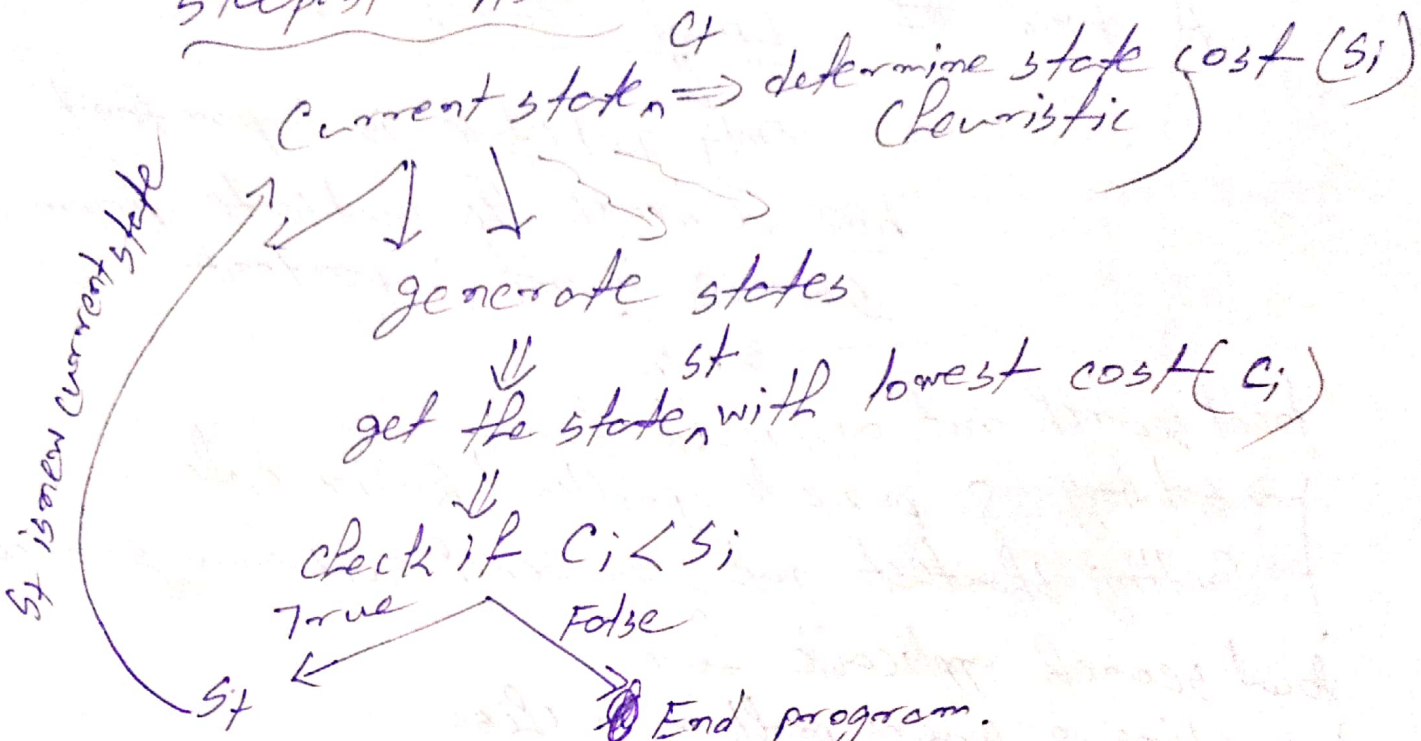
local search applicable for
- → solving 8 queens problem in less
- → solving class routine conflict
- → sorting an array (not done in real life)

** In games state space is really large. There are endless possibilities. Local search is used often, as local search does not go for optimal solution, rather goes for fast, greedy suboptimal solution.

During algorithm => only stores current state
O(1) memory requirement

* Many variants of Hill climbing search
- → steepest Ascent (~~22.5~~ 11 marks, pseudocode given)
- → first choice
- → random restart
- → simulated annealing (~~12.5 marks bonus, pseudocode not given~~ 11.5)

Steepest Ascent



BUT!! we may not reach goal state this way.
TRUE!! we may get stuck in a state.
There is no guarantee of reaching goal state using steepest Ascent Hill Climbing

Problem ⇒ Array sorting in ascending order

itr 1 current state$_n^{C_t}$ ⇒ [2, 1, 5, 0]

cost S_i ⇒ 2 + 1 + 1 + 0 = 4

Intuition: Numbers in front of a particular element should be smaller than that number

Generated states:

(1) [1, 2, 5, 0] ⇒ 1 + 1 + 1 = 3

(2) [5, 1, 2, 0] ⇒ 3 + 1 + 1 = 5

(3) [0, 1, 5, 2] ⇒ 0 + 0 + 1 = 1

(4) [2, 5, 1, 0] ⇒ 2+2+1 = 5

(5) [2, 0, 5, 1] ⇒ 2+0+1 = 3

(6) [2, 1, 0, 5] ⇒ 2+1+0 = 3

lowest cost state ⇒ [0, 1, 5, 2], cost = 1

1 < 4

∴ This is the current state

itr 2

Current state ⇒ [0, 1, 5, 2]

cost ⇒ 1

Generated states

(1) [1, 0, 5, 2] ⇒ 1+0+1 = 2

(2) [5, 1, 0, 2] ⇒ 3+1+0 = 4

(3) [2, 1, 5, 0] ⇒ 2+1+1 = 4

(4) [0, 5, 1, 2] ⇒ 0+2+0 = 2

(5) [0, 2, 5, 1] ⇒ 0+1+1 = 2

(6) [0, 1, 2, 5] ⇒ 0+0+0+0 = 0

lowest cost state ⇒ [0, 1, 2, 5], cost = 0

0 < 1

∴ This is current state

itr 3 If you generate states from [0, 1, 2, 5], all state costs will be higher or equal.

So, $c_i < s_i$ condition will be cost.

Ans: [0, 1, 2, 5]

Simulated Annealing

Problem of algorithms like steepest Ascent is: It may get stuck in a state before reaching goal.

** Do we always need to go to better state from current state??

=> We shall certainly go to a better state if we can. But sometimes we must cope up with the downs in our life.

Algorithm

(1) Get cost S_i from current state C_f

(2) Generate states from C_f

while generating states:

=> for each generated state:

if the state has lower cost than S_i:

Immediately move to that state

if costs are equal:

$\Delta E = -1$, moveornot(ΔE)

else:

ΔE = Current state cost(S_i) − generated state cost(C_i)

moveornot(ΔE)

(3) Keep iterating until goal node reached.

move or not (ΔE):

Calculate $e^{\Delta E}$ (suppose, 0.15)

generate random number $\Rightarrow [0, 1]$

if Random $\Rightarrow [0, 0.15] \Rightarrow$ move to generated state

else $\Rightarrow$ do not go there

Analogy:

$S_i = 5$ (current state cost)

$C_{i_1} = 7$ $\quad C_{i_2} = 10$

$\Delta E_1 = -2$ $\quad \Delta E_2 = -5$

$e^{\Delta E_1} = 0.135$ $\quad e^{\Delta E_2} = 0.007$

Random $\Rightarrow [0, 0.135]$ $\quad$ Random $\Rightarrow [0, 0.007]$

C_{i_1} is much more likely to occur ~~that~~ than C_{i_2}. Better states with lower cost will be ~~less~~ more likely to be visited.

~~#~~ Why $\Delta E = -1$ for equal cost??

If $\Delta E = 0 \Rightarrow e^{\Delta E} = e^{00} = 1$

Random $\Rightarrow [0, 1] \Rightarrow$ will always move here.

But we do not want this.