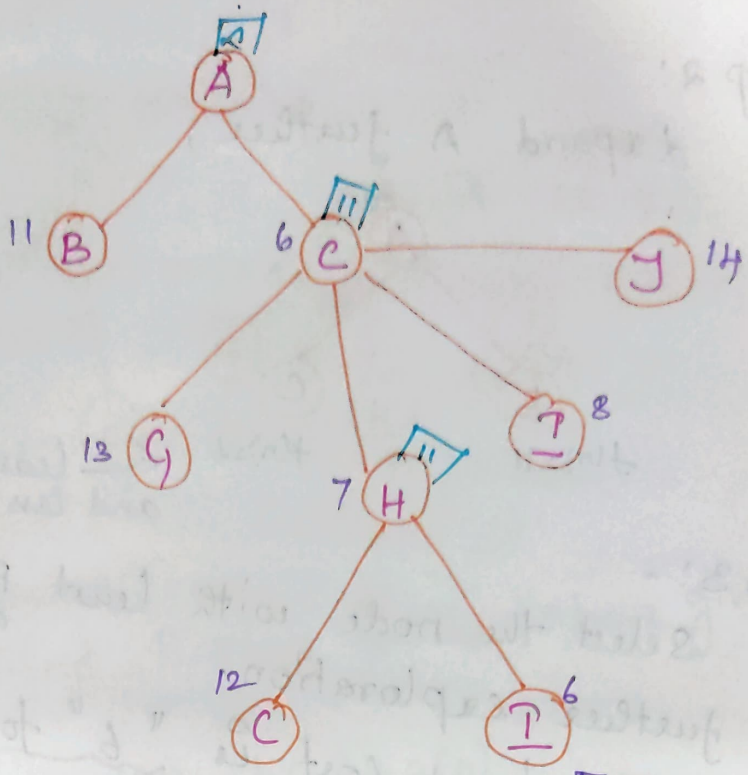


↖ least cost
6 < 11
and goal node

Step 4:-

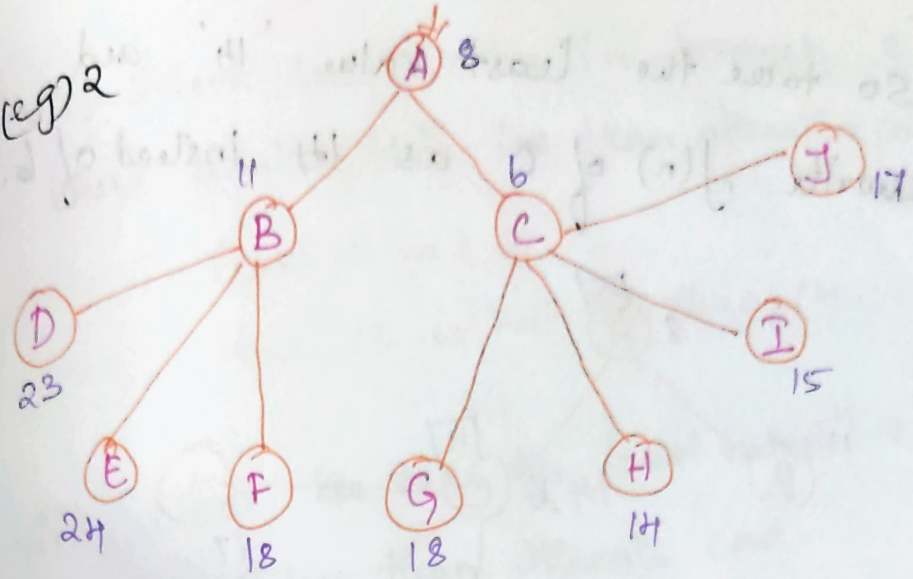
* Select the node with least cost (i.e) I = 6
* So expand "I" further and check whether it is less than alternative cost too
* But 'I' is the Goal node, we have reached the Goal.

Path: A → C → H → I

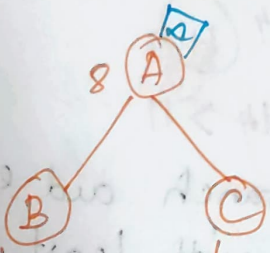
What happens if least value is not less than alternate cost?

Consider the below given graph to solve the problem of find path when least value is not less than alternate cost.

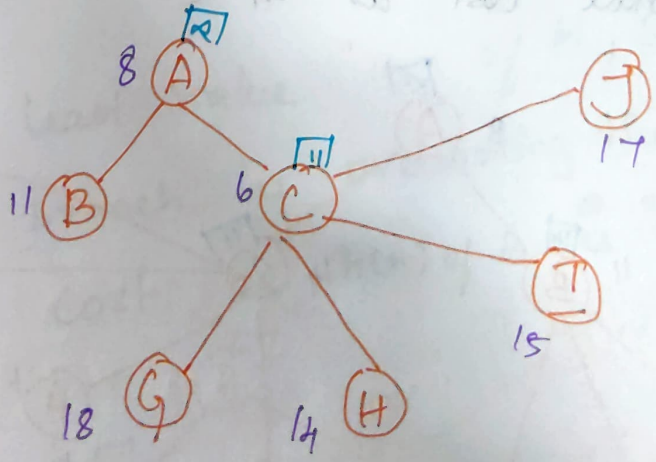
eg) 2



i) Expand A with alternate cost as ∝



ii) The least value is "6" so expand "C" with alternate cost as " "



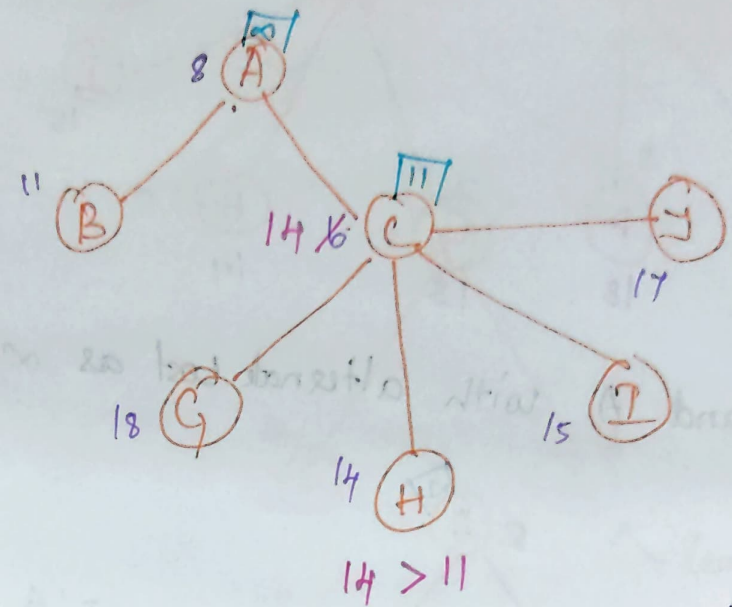
iii) select least values and check whether it is less than alternate cost as well.

18 ≮ 11 15 ≮ 11

14 ≮ 11 17 ≮ 11

Least value is "14" for H

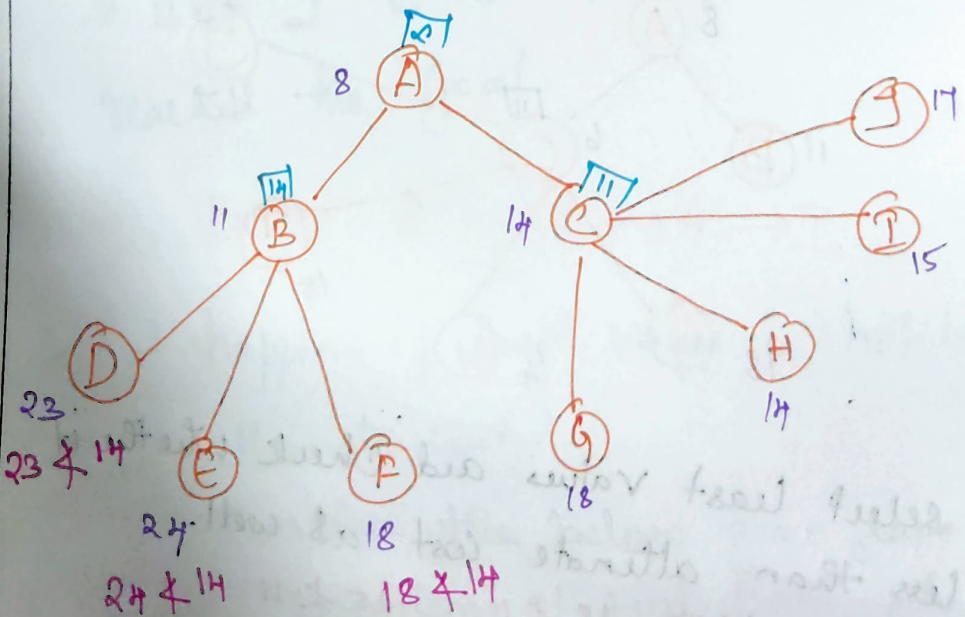
iv, So take the least value '14' and overwrite f(n) of C as 14 instead of 6.



v, Move to next Branch and expand the least one.

vi, Now the node with least cost is "B"

vii, So expand "B" and assume the alternate cost as 14 (ii) h(n) of C = 14



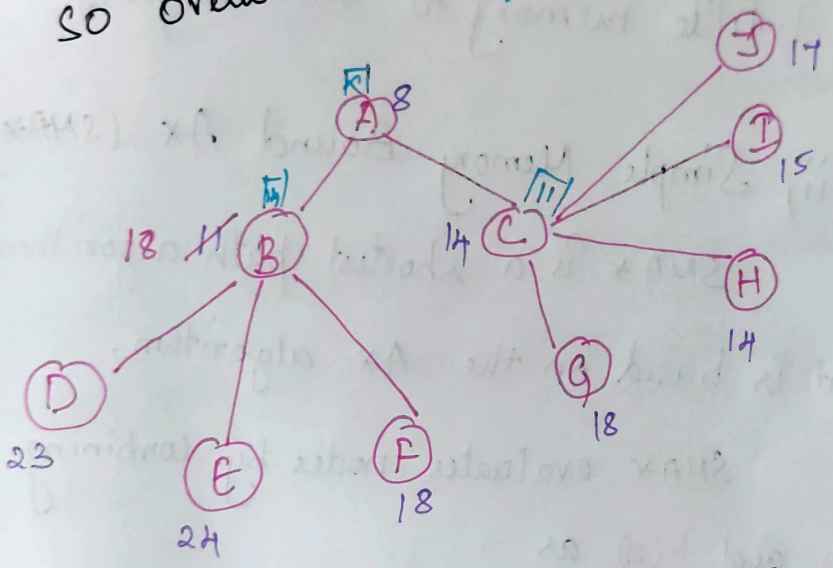
vii, Select least value in branch "B" and check whether it is less than alternative cost = 14.

Least value = 18

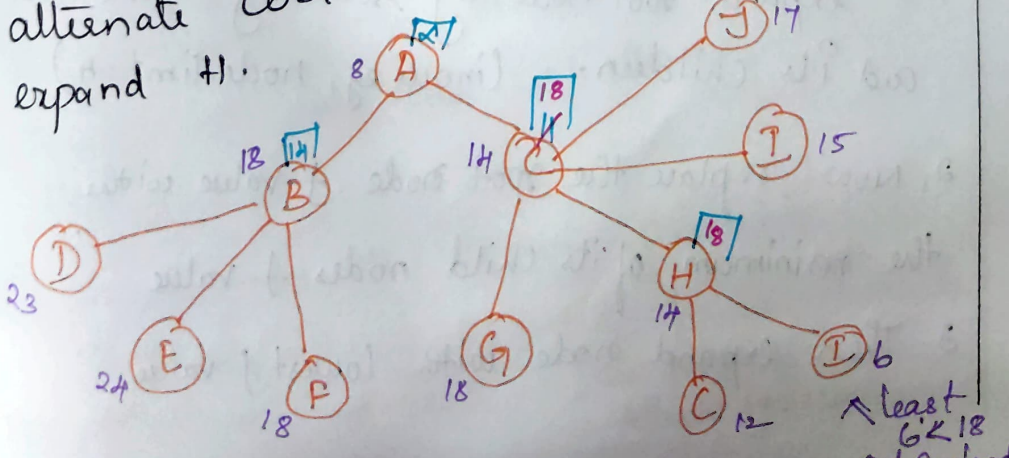
but 18 is not less than 14.

viii, Now take the least value "18" but it is not less than alternate cost.

so overwrite h(n) of B = 18



ix, The least value "14" of H, so move to next branch by overwriting the alternate cost as h(n) of B is "18" and expand H.



x) The least value is 6 and $6 < 18$ we can further explore 'I' if needed by we have reached the goal. So we can terminate the process.

Disadvantage :-

1. little memory to save the path.

E. iii) Simple Memory Bound A* (SMA*)

SMA* is a shortest path algorithm that is based on the A* algorithm.

SMA* evaluates nodes by combining $g(n)$ and $h(n)$ as

$$f(n) = g(n) + h(n)$$

Procedure :-

1. 1st expand root node by selecting a node-limit and its children. (in our eg, node-limit = 4)
2. Now replace the root node f-value with the minimum of its child nodes f-value
3. Then expand node with lowest f-value.

4. If the node-limit exceeds, drop the node with highest f-value. Keep its f-value in paranthesis at its parent node.

5. After all the childs of parent has been visited, replace parent's f-value with minimum of paranthesized f-value & child's f-value.

6. If at the last level (limit-4), replace the fvalue of node with ∞ since it reached the limit.

7. continue dropping and adding of nodes till goal node is reached with minimum f-value.

Peoformance Evaluation:-

1. Will be complete if the allowed memory is high enough to store the shallowest solution.

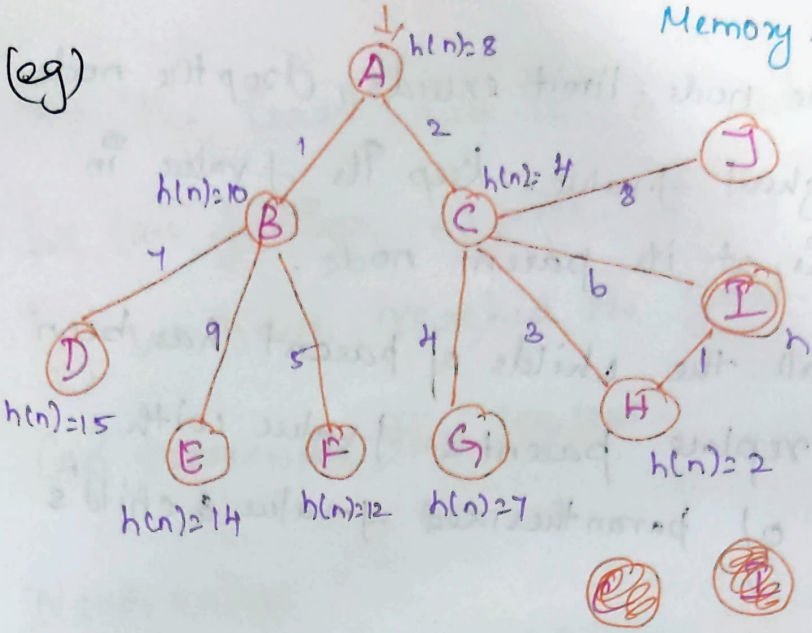
2. Optimal if the allowed memory is high enough to store the shallowest optimal solution.

3. It will use all memory available.

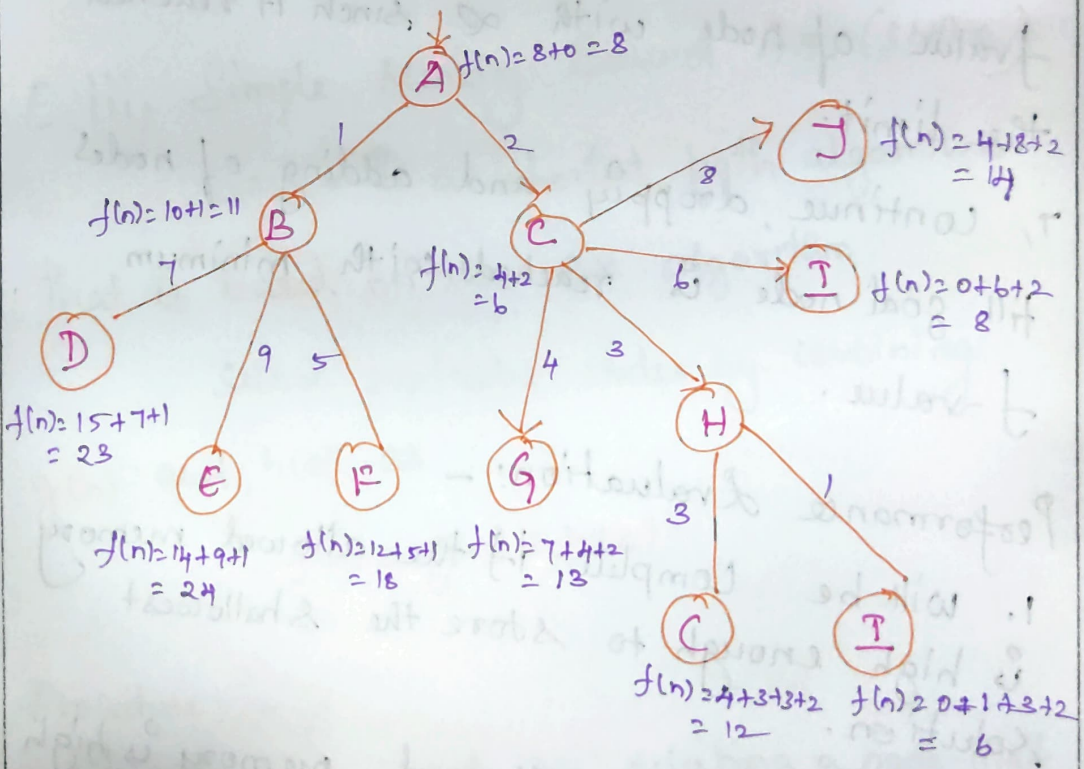
4. Time complexity reduced because it avoids repeated state as long as the memory bound allows it.

Memory size = 4 nodes

(eg)



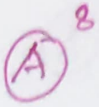
f-score

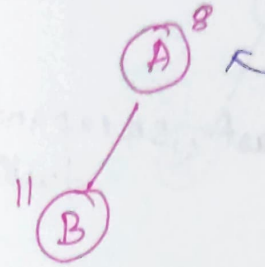


Note :-

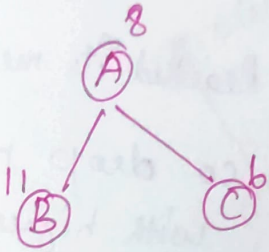
1. Expand the node with least cost.
2. Replace the parent node's f-score with the dropped child's f-score.
3. To add new node beyond the given memory size then drop the existing node with highest f-score.

Expand A

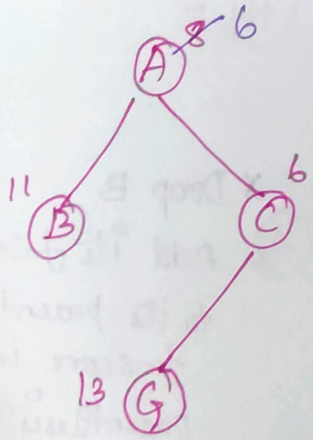




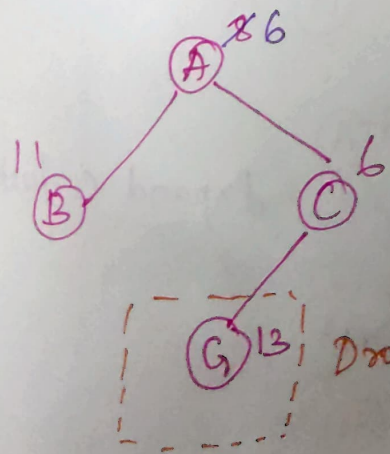
← least cost so expand A.



← least cost so expand C.

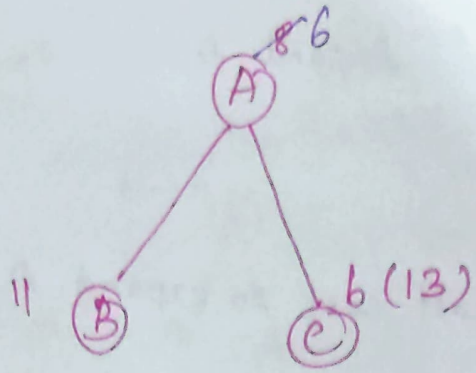


* Reached the memory limit.
* To add further we have to drop the node with highest f-score
* Replace its parents $f(n)$ as expanded child's $f(n)$

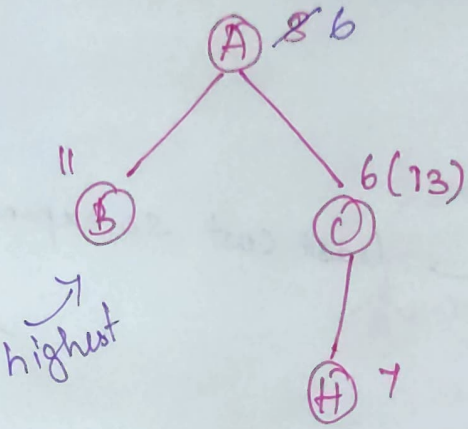


Drop G.

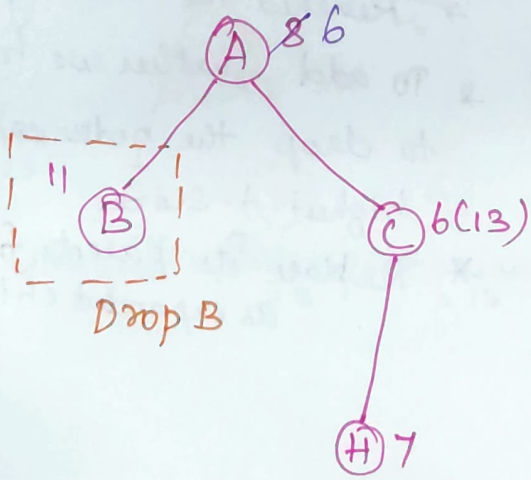
* Drop G
* Add its f-score to its parent node.



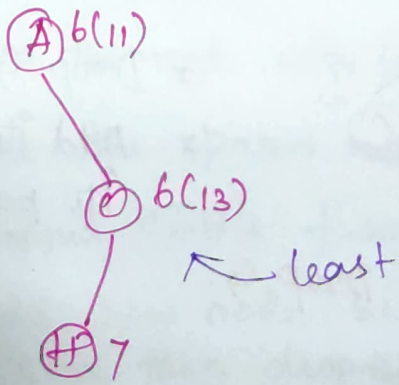
← least, so expand further



* Reached its memory limit
* so drop node with highest f-score

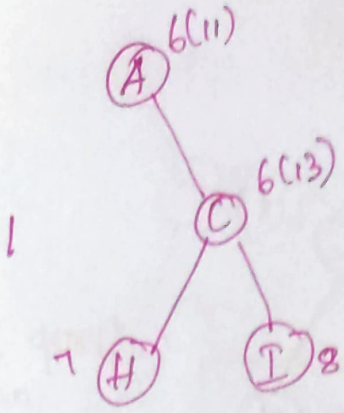


* Drop B
* Add its f-score to its parent f-score in a parenthesis



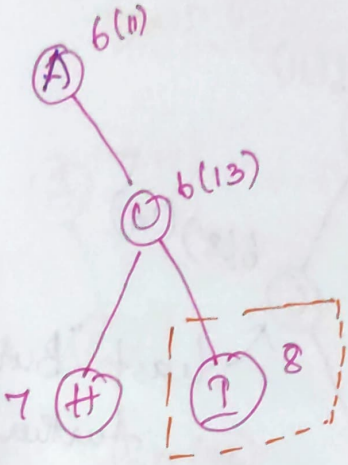
← least

Expand C further

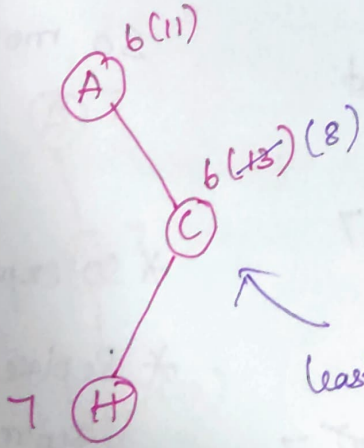


* Reached the memory limit
* Drop node with highest score
* Add its f-score to its parents in parenthesis

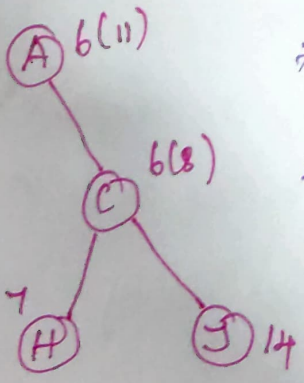
← highest f score



* Add f score of I to C

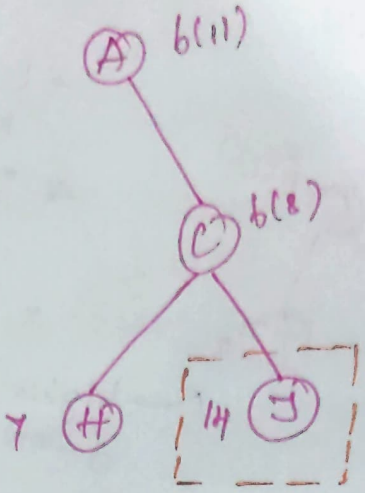


* Expand further

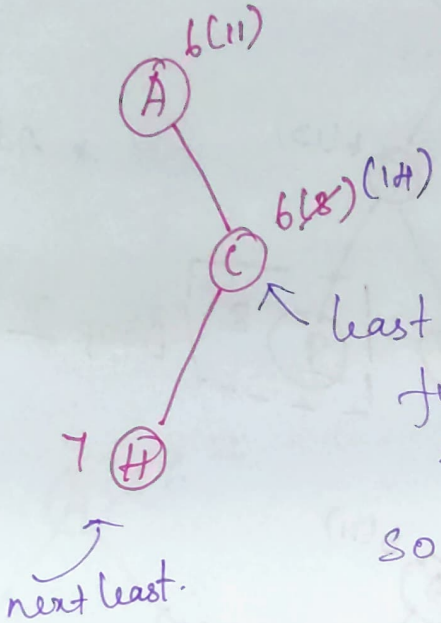


* Reached the memory limit 4.
* Drop J with highest f-score
* Add J's f score to its parent

↖ highest f-score



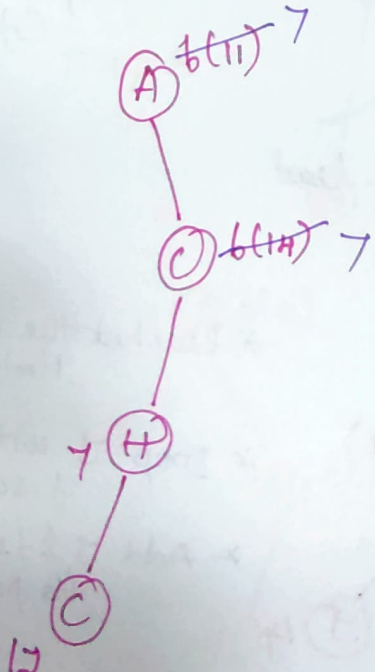
Drop J and add its f-score to its parent



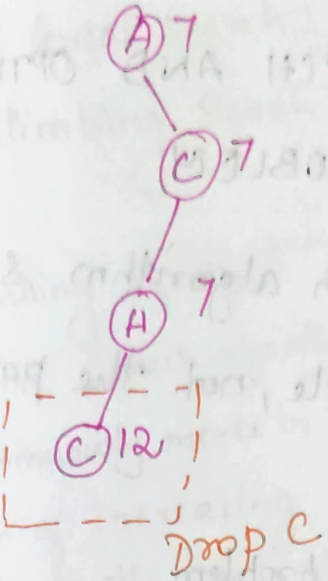
↖ least But there is no further node to expand in C.

So move to next least

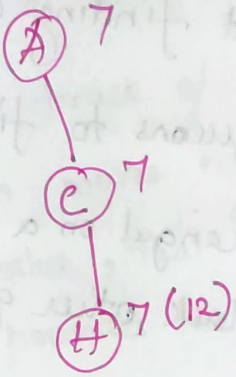
next least.



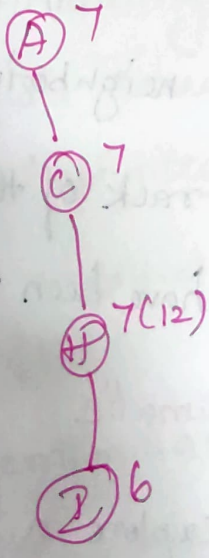
* So expand H.
* Replace its parent f-score as H's f-score



* Reached its memory limit 4.
* Drop the node with highest f-score
* Add the f-score of dropped node to its parent in parenthesis



* Expand H further



LOCAL SEARCH AND OPTIMIZATION PROBLEM

The local search algorithm searches only the final state, not the path to get there.

(eg) 8-queens problem

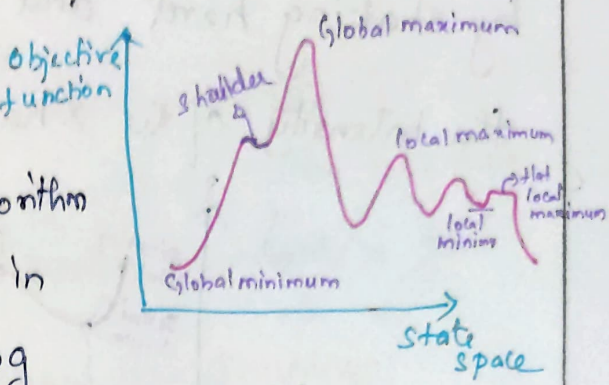
We care only about finding a valid final configuration of 8-queens to final state.

Note: 8-queens arranged on a chess board and no queen can attack other queens.

* Local search Algorithms operate by searching from a start state to neighboring states.
* Without keeping track of the paths, nor the set of states that have been reached.
* They are not systematic
* They might never explore a portion of the search space where a solution actually resides.
* They searches only the final state.

Types of local search: -

a, Hill- climbing Seach Algorithm



Hill climbing algorithm is a Heuristic Search algorithm which continuously moves in the direction of increasing value to find the peak of the mountain or best solution to the problem.

It keeps track of one current state and on each iteration moves to the neighboring state with highest value, that is) it heads in the direction that provides the steepest Ascent.

b, Simulate Annealing: -

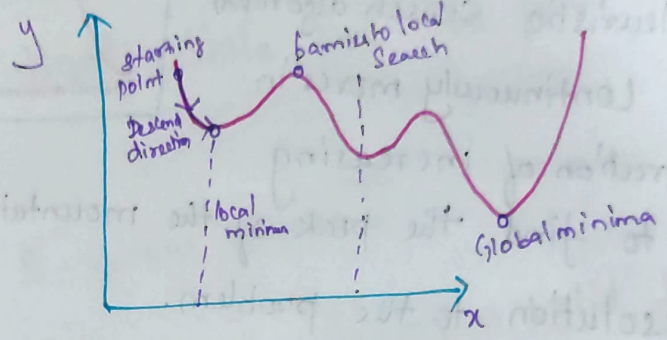
Simulate Annealing is a stochastic Global seaech optimization algorithm.

It is a modified version of stochastic hill climbing.

This algorithm is appropriate for nonlinear objective functions where other local search algorithm don't operate well.

It very much usefull when there are a lot of local minima.

The simulate-annealing solution is to start by shaking hard, and then gradually reduce the intensity of the shaking.

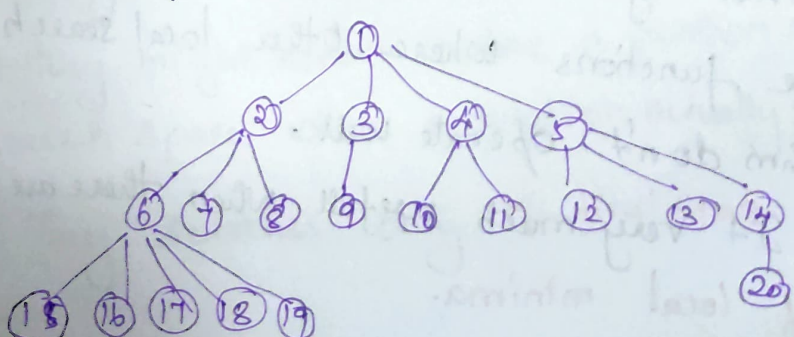


C) Beam Search Algorithm

Beam Search Algorithm is a heuristic search Algorithm that examines a graph by extending the most promising node in a limited set.

The number of nodes n represents the beam width.

This algorithm only keeps the lowest number of nodes on open list



components of Beam Search:-

It takes 3 components as its input

1. Problem:

The problem usually represented as graph and contains a set of nodes in which one or more of the nodes represents a goal.

2. Set of Heuristic rules for Pruning:

These are rules specific to the problem domain and prunes unfavorable nodes from memory regardig the problem domain.

3, Memory with limited capacity:

The memory is where the "beam" is stored, memory is full, and a node is to be added to the beam, the most costly node will be deleted, such that the memory limit is not exceeded.

D, Genetic Algorithm:

Genetic Algorithm is a search technique used to find true or approximate solutions.

It is categorized as Global search Heuristics.

GA uses techniques inspired by evolutionary biology such as Inheritance, Mutation, selection and Cross over (also called as recombination)

procedure in GA:-

* The evolution usually starts from a population of randomly generated individuals and happens in generations.

* In each generation, the fitness of each individual in the population is evaluated.

* Multiple Individuals are seleted from the current population based on their fitness.

* Modified to a form a new population.

* The new population is then used in the next iteration of the algorithm.

CONSTRAINT SATISFACTION PROBLEM

CSP are mathematical questions defined as a set of objects whose state must satisfy a number of constraints or limitation.

It represents the entities in a problem as a homogeneous collection of finete constraints over variables, which is solved by constraint satisfaction method.

Constraint Programming:-

CP is the field of research that specifically focuses on tackling these kinds of problem.

Formally, CSP is defined as triple(V, D, C),

Where $V = \{V_1, V_2, V_3, \cdots V_n\}$ set of variables

$D = \{D_1, D_2, D_3, \cdots D_n\}$ set of their respective domain of values

$C = \{C_1, C_2, C_3, \cdots C_m\}$ set of constraints

Value Assignment:-

The variables are assigned with values in 3 methods,

1. **Consistent or legal assignment**

if a task complies with all laws and constraints.

2, Complete Assignment

Each variable will be assigned with values

3, Partial Assignment:

Some of the variables will be assigned with values.

Domains :-

1. Discrete Domain :-

This limitless area allows for the existence of single state with numerous variable.

(for eg) every parameter may receive a endless number of begining states.

2, Finite Domain :-

Continuous phases that can be described just one area for just one particular variable.

Constraints :- Rules & Regulation

1. Unary Restriction :-

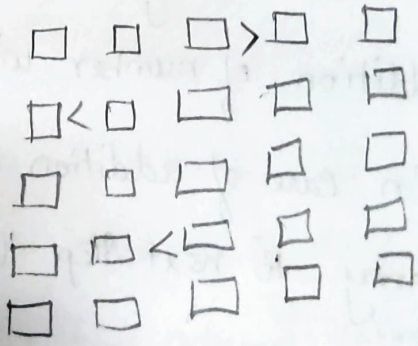
Only limits the value of one variable.

2. Global Resource limit:
* unrestricted amound of variables.
(eg) Types of inference - progeamming language, Natural language

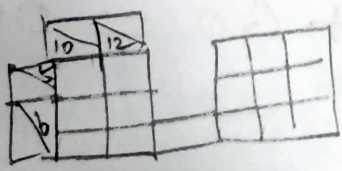
3. Binary Resource limit
- Restriction connect two parameter
- A value between x3 & x1
Can be found in variable named.

Application:-
8 Queens
Map Coloring Problem
Maximum cut problem
Sudoku
Cross word.

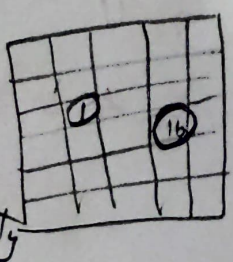
Futoshiki
(fill 1 to 5)



Kakuro



Hidato
fill 1-16 only once.
two adjacent numbers are consequity



Types of Constraint Satisfaction Problem:-

1. Crypt - Arithmetic Problem:

It is a type of encryption problem in which the written message in an alphabetical form (which is easily readable and understanble) is converted into numerical form (which is neither readable nor understandable)

Comstraints:-

1. Every Character must have unique value
2. Digits should be 0-9 only
3. Starting Character of number can't be zero
4. Will have only one solution.
5. Addition of number with itself is always even.
6. In case of addition of two numbers, if there is carry to next step then Carry can only be '1'.

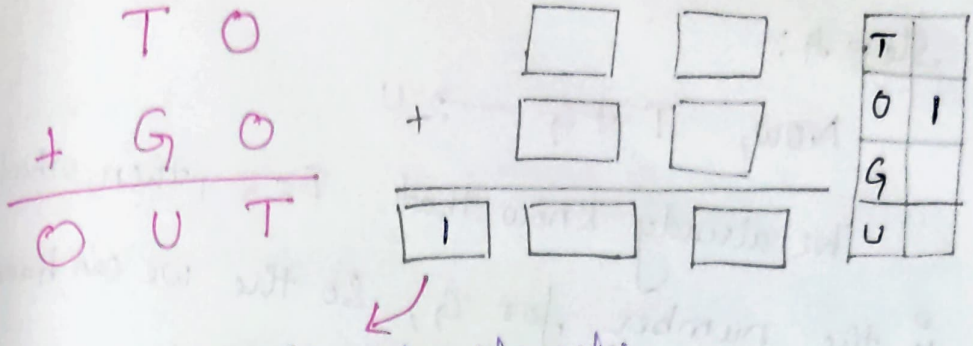
(eg) 1. TO + GO = OUT

T	
O	
G	
U	

Given,

Variable: T, O, G, U

Domains: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

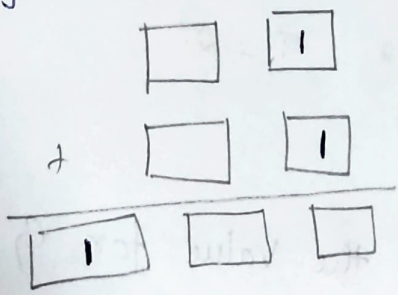


Step 1:

If two numbers added, then carry will be '1'. So write '1' in that box and ~~add the other~~ Fill the table at O with '1'.

Step 2:

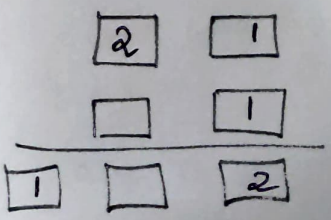
If there are any other O in the other boxes then fill it ~~with~~ also with '1'.



Step 3:

Add 1+1 → 2, Assign 2 to T in box and fill it in the tabular column corresponding to T.

If there are any other 'T' in the word, then fill is also with 2.



T	2
O	1
G	
U	

Step 4:

Now, $T + G \longrightarrow U$

We already know that $T = 2$, then what is the number for G, so the we can have the carry from ~~the~~ as two digit number

$T = 2 \quad G = ?$ $\quad T + G \longrightarrow U$

option 1 If $G = 8$ $\quad 2 + 8 = 10$ 1-carry 0 for U

option 2 If $G = 9$ $\quad 2 + 9 = 11$ 1-carry 1 for U

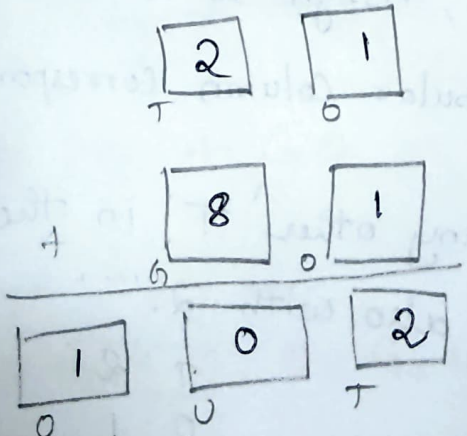
option 2 is not possible, because then both the letter O and U will be having "1"

Hence, Take option 1,

$G = 8 \quad T = 2$

Step 5:

Now, fill the value for G in the box as well as the tabular column.



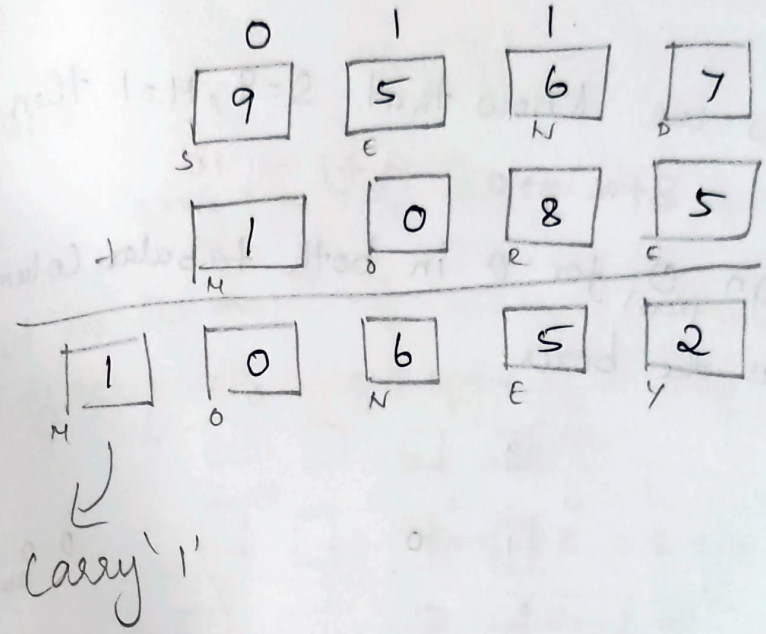
T	2
O	1
G	8
U	0

(eg 2) SEND + MORE = MONEY

Given,

variables: S E N D M O R Y

Domain : 0 - 9



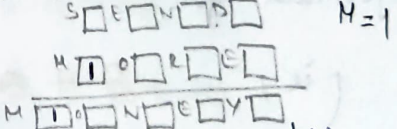
S	9
M	1
E	5
O	0
N	6
R	8
D	7
Y	2

carry '1'

Step 1:

There is a extra character in result, then carry is '1'.

So Assign, M = 1 in tabular column as well as in all the boxes where it is M.



Step 2:- S + M → O

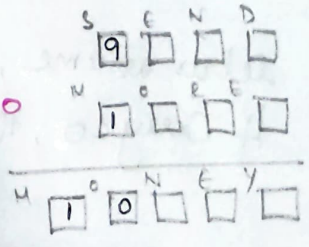
What number should be added with M=1, so that we would get two digit number with carry as 1.

∴

Option: S = 9 M = 1 so

9 + 1 → 10 → assign to O

assign To M



Step 3:-

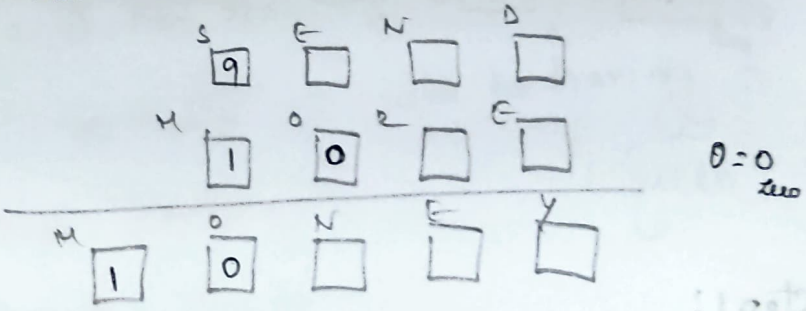
So write 9 at S, in tabular column and replace all the boxes for 's' with 9.

Step 4:-

Now we know that S=9, M=1 then

$S+M \rightarrow O \quad 9+1 = 10$

Assign O (zero) for 0 in both tabular columns as well as the boxes.



Step 5:-

Now, we know that O = 0, form a formula for next boxes,

(ie) $E + O \rightarrow N$

$Carry + E + O \rightarrow N$

We know that O=0, Carry = 0 or 1

Let us assume,

if Carry=0, then $O + E + O = N$

$E = N$ - not possible, two characters can't have same number