# Branch & Bound

- Used to find optimal solution to many optimization problems.
- Systematic enumerates all candidate solutions, by discarding in between candidate state by using upper and lower bounds.

→ Terminology:

1. Live Node: Node that is generated but whose child nodes are not yet generated completely.
2. E-Node: is a live node that is currently being explored.
3. Dead Node: is a generated node that is not to be expanded or explored any further.

Branch & Bound (BB):

All state space search methods in which all children of E-node are generated before any other live node can become the E-node.

→ General Method

- Both BFS and DFS generalized to BB strategies

1) BFS - FIFO search
   In terms of live node
   - List of live node is a queue

2) DFS ← search - LIFO Search
   In terms of live node
   - List of livenode is a stack

- Like Backtracking, BB uses bounding function to avoid generating subtrees that do not contain an answer node.

THAKUMAL SHAHANI

PAGE No.
DATE / / 2019

- FIFO - Branch & Bound :- Search the tree using BFS with ranking fn
- LIFO - Branch & Bound : Search the tree as a BFS, but replace the FIFO queue with a stack. With ranking fn - also known as D-Search.

- LC (Least Cost) Branch & Bound : Replace the FIFO queue with priority queue.
  - The priority of a node 'p' in the queue is based on an estimate of the likelihood that the answer node is in the subtree whose root is 'p'.

- LC Branch & Bound directs the search to parts of the space most likely to contain the answer.

- Ranking function :
  - In LC Branch & Bound, Ranking fn is the basis on which next E-node is selected.
  - In general, for a node 'x', ranking function is,

$$c(x) = \underbrace{g(x)}_{\text{cost of reaching } x \text{ from root}} + \underbrace{\hat{g}(x)}_{\text{Estimate of additional effort required to reach an answer node.}}$$

# 15- Puzzle Problem

PAGE No.
DATE / /

## [I] Problem :

- Given a 4x4 board with 15 tiles, and every tile has one number from 1 to 15 and one empty space.
- Also, initial state and goal state is given, objective is to transform from initial state to goal state using legal moves.

| | | | |
|---|---|---|---|
| 1 | 3 | 4 | 15 |
| 2 | | 5 | 12 |
| 7 | 6 | 11 | 14 |
| 8 | 9 | 10 | 13 |

Initial state (IS)

| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 |
| 13 | 14 | 15 | |

Goal state (GS)

- Legal move involves moving a tile adjacent to the empty spot E to E.
- For this initial state, four possible moves are moving tiles 2, 3, 5, 6.
- A state is reachable from the initial state iff there is a sequence of legal moves for IS to GS

## [II] Check whether GS is reachable from IS :

- Number the frame positions from 1 to 16.
- $P_i$ is the frame position containing tile numbered i
- $P_{16}$ is the position of empty spot.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

$P_{IS} = [1, 5, 2, 3, 7, 10, 9, 13, 14, 15, 11, 8, 16, 12, 4, 6]$

$P_{GS} = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]$

→ For any state, let $l_i$ be the number of files $j$ such that

(a) $j < i$ and $P_j > P_i$

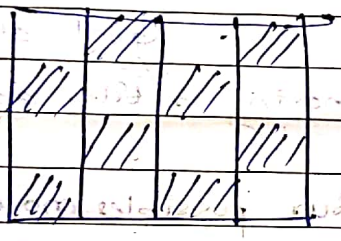
i.e $l_i$ gives the number of lower-numbered files that are to the right of file $i$ in that state.

– Example,

$l_1 = 0$ , $l_4 = 1$ , $l_{12} = 6$ , $l_{13} = 0$

→ Let $n = 1$ if the IS, the empty spot is in one of the following positions

i.e 1, 2, 4, 5, 7, 10, 12, 13, 15 ; else $n = 0$



shaded, $n = 1$

$n = 0$

Theorem:

GS is reachable from IS iff

$$\sum_{i=1}^{16} l_i + n_i \text{ is even.}$$

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $l_i$ = | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 3 | 6 | 0 | 11 | 14 |

$\sum_{i=1}^{16} l_i = 40 + 1$ , $n = 0$.

$= 40 + 0 = 40 \Rightarrow R$

if $(R \% 2 == 0) \Rightarrow$ possible

$\neq 0 \Rightarrow$ Not possible.

III Organize state space as a tree

- child of each node 'x' represents the states that are reachable from 'x' in one legal move.
- Consider, the move as move of empty space rather than the tile.
- Empty space can have 4 legal moves:- up, down, left, right.
- No node should have a child state that is same as its parent.
- let us order the move of empty space as up, right, down, left (clockwise moves).

* LIFO BB (LIFO BB) more
* DFS - takes us away from GS
  - Search of state space tree is blind.
  - taking the leftmost path from the root regardless of IS.

(FIFO-BB)
* BFS - find a GS nearest to the root.
  - It is still blind because no matter what is IS, the algo. attempts the same sequence of moves.

* Intelligent fn: (LC - Branch & Bound)
  - Also known as ranking fn / cost fn.
  - In search using ranking fn, the next E-node generated is the one with least cost.

- For 15 puzzle, we assume that moving one tile in any direction will have 1 unit cost.

- Cost fn of node(x) $\Rightarrow c(x) = g(x) + h(x)$

where, $g(x)$ = length of path from root to x (No. of moves so far)

$h(x)$ = Number of non-blank tiles not in their goal position (No. of misplaced tiles).

Initial state

| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 5 | 6 | | 8 |
| 9 | 10 | 7 | 11 |
| 13 | 14 | 15 | 12 |

Goal state

| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 |
| 13 | 14 | 15 | |

(1)

| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 5 | 6 | | 8 |
| 9 | 10 | 7 | 11 |
| 13 | 14 | 15 | 12 |

up → (2) c = 1+4

| | | | |
|---|---|---|---|
| 1 | 2 | | 4 |
| 5 | 6 | 3 | 8 |
| 9 | 10 | 7 | 11 |
| 13 | 14 | 15 | 12 |

right → (3) c = 1+4

| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 5 | 6 | 8 | |
| 9 | 10 | 7 | 11 |
| 13 | 14 | 15 | 12 |

down → (4) c = 1+2

| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |
| 9 | 10 | | 11 |
| 13 | 14 | 15 | 12 |

left → (5) c = 1+4

| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 5 | | 6 | 8 |
| 9 | 10 | 7 | 11 |
| 13 | 14 | 15 | 12 |

(backtracking)

right → (6) c = 2+1

| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | |
| 13 | 14 | 15 | 12 |

(a)

down → (7) c = 2+3

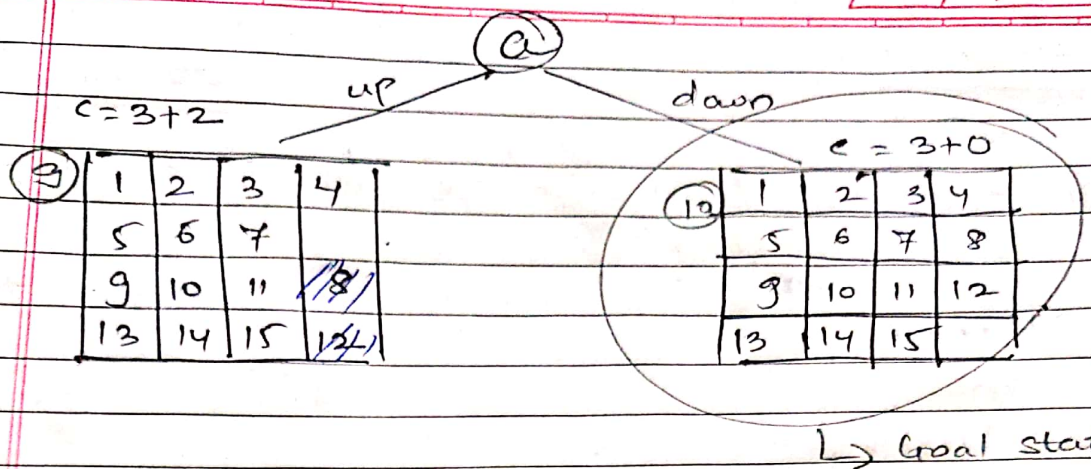
| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 |
| 13 | 14 | | 12 |

(b)

left → (8) c = 2+3

| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |
| 9 | | 10 | 11 |
| 13 | 14 | 15 | 12 |

(c)




Algorithm:

| Iteration | LiveNodes (min priority Queue) | E-node |
|---|---|---|
| 0. | c(1) | Node 1 |
| 1. | c(2) = 5, c(3) = 5, c(4) = 3, c(5) = 5 | Node 4 |
| 2. | c(2) = 5, c(3) = 5, c(5) = 5, c(6) = 3, c(7) = 5, c(8) = 5 | Node 6. |
| 3. | c(2) = 5, c(3) = 5, c(5) = 5, c(7) = 5, c(8) = 5, c(9) = 5, c(10) = 3 (GS) | Goal State |