ch→6 Backtracking & Branch and Bound.

Page No.
Date :

→ Backtracking

- It uses brute force approach.
  (In this approach we will find/ solve all possible solutions)

→ It is used to find all possible solution

state space tree → used

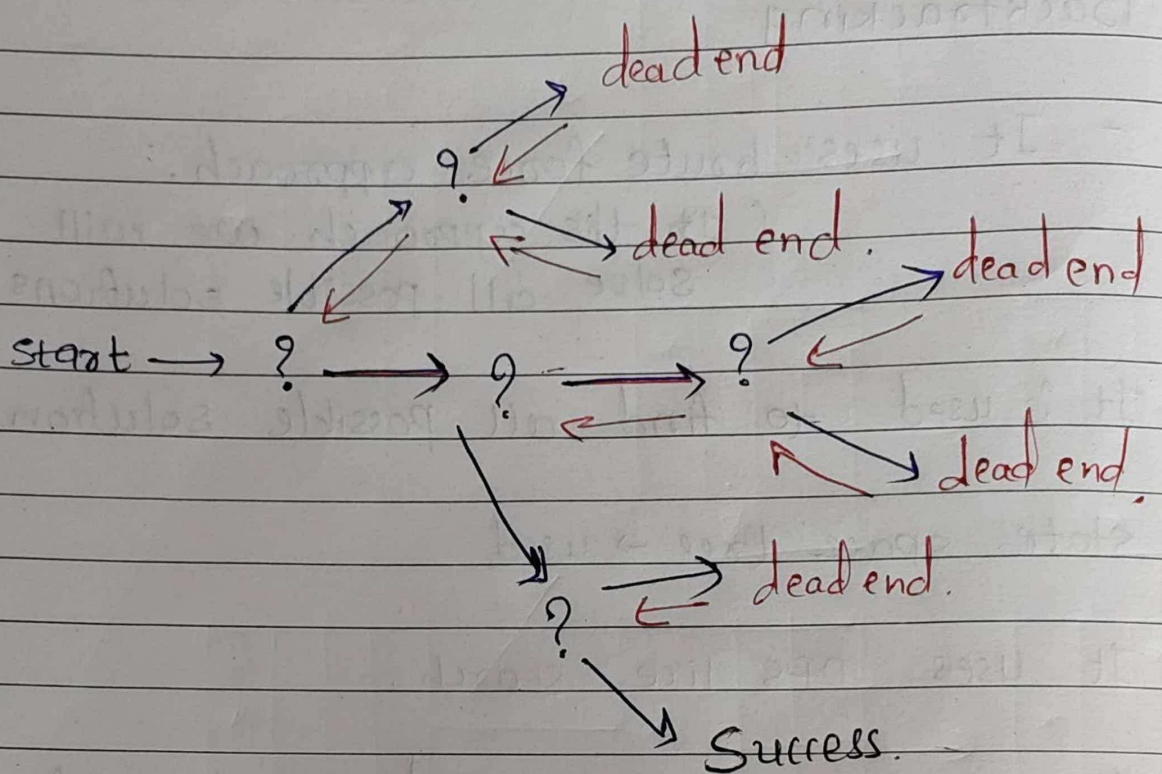
→ It uses DFS like search.

- Backtracking is also called as DFS of a tree.

→ Backtracking is the procedure whereby, after determining that a node can lead to nothing but dead node, we go back ("backtrack") to the node's parent and proceed with search on the next child.

- Node is nonpromising if when visiting the node we determine that it cannot possibly lead to a solution. otherwise, we call it promising.

→ Summary,
  → Doing DFS.
  - check whether node is promising, and, if it is nonpromising, backtrack to it's parent node.



☆ Backtracking Algorithm.

→ We Explore each node, as follows:

To explore node N:

1. If N is goal node, return "success"
2. If N is leaf node, return "Failure"
3. For each child C of N.
   3.1 Explore C
      3.1.1 If C was successful, return "suc
4. Return "Failure"

Application of Backtracking

- N-queen problem
- Coloring map
- Sum of subset
- Solving maze.
- Solving puzzle.

N-Queen Problem

Problem :- try to place N queens on a N×N board such that none of the queens can attack another queen.

- Queen can move horizontally, vertically or diagonally any distance.

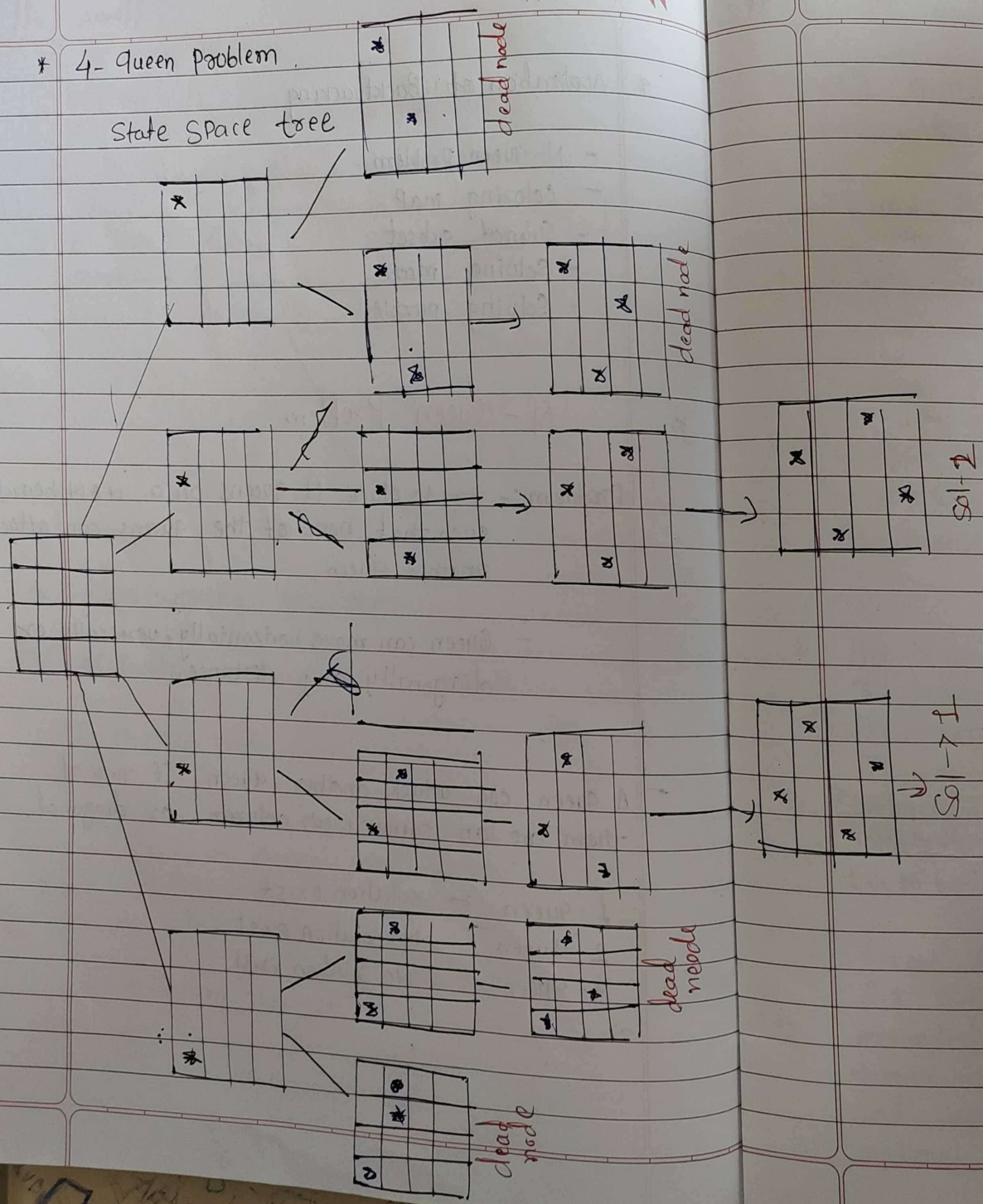
- A queen can attack another queen, if two of them are on same row, column, or diagonal.

1-queen → solution exist
2-queen - No solution exist
3-queen - No solution exist

Page No.
Date :

* 4- Queen problem.

State Space tree



Page No.
Date :

Branch and Bound.

Page No.
Date :

* B&B is used to find optimal solution (only minimization problem)

→ It uses BFS like search.

→ It uses state space tree for solution.

→ Methods For solving ~~bre~~ problem using B&B or B&B strategies
  → FIFO Branch and bound.
  → LIFO Branch and bound
  → Least-cost Branch and bound.

* BB is an algorithm design paradigm which is generally used for combinatorial optimization problems.

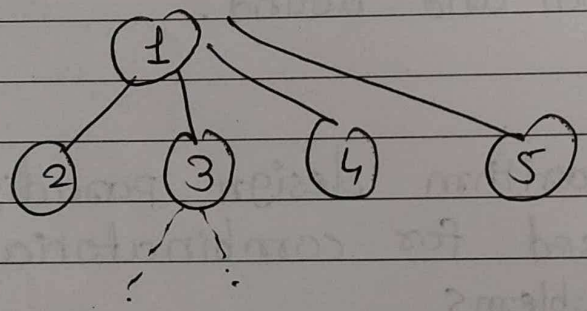
→ Combinatorial optimization problem is an optimization problem, where an optimal solution has to be identified from a finite set of solutions

- General idea of B&B is a BFS-like search for optimal solution, but not all node get expanded. Rather, a carefully selected criterion determines which node to expand and when, and another criterion tells the algorithm when an optimal solution has been found.

Live node : is a node that has been generated but whose children have not yet been generated

E-node (expanded node) : is a live node whose children are currently being explored (or, expanded)

Dead Node : is a generated node that is not to be expanded or explored any further.
All children of a dead node have already been expanded.



live node → 2, 4, 5
e-node → 3
dead node → 1

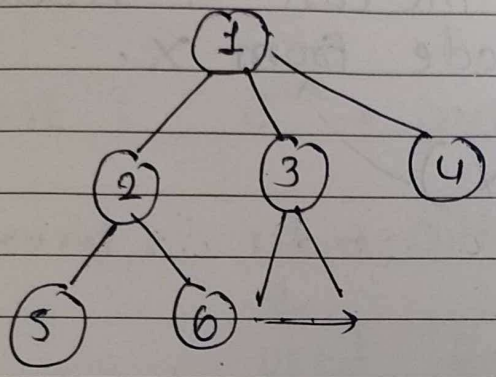
Bounding function is used to kill live node without generating all their children.

Page No.
Date :

B & B strategies

1) FIFO Branch and Bound (or BFS)

- It is an FIFO search in terms of live nodes.
- List of live nodes in queue.

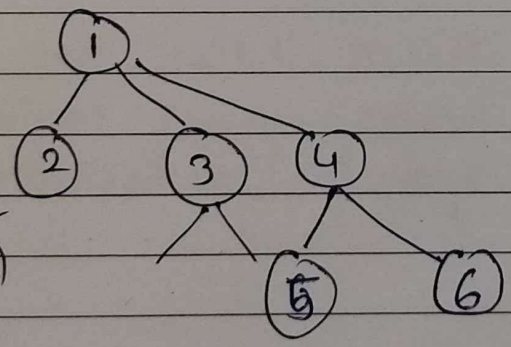


children of E-node are inserted in queue

2) LIFO Branch and Bound (or D-search)

- List of live nodes is a stack

selection of E node is blind (we don't know which E-node is good)



Children of E-node are inserted in stack

3) Least cost Branch and Bound: (LC-search)

Uses approximation cost function

→ LC uses Approximate cost Function for selecting E-node (using this we can improve search)

Cost function.

Each node x,

$c(x)$ = cost of reaching node x (E node) from root + the cost of reaching an answer node from x.

$c(x) = g(x) + h(x)$

Page No.
Date :

0/1 knapsack problem (Binary knapsack)

i/p : Weight and profit of N items.
Knapsack capacity S.

output : selection for knapsack : $\{x_1, x_2 \dots x_n\}$
where $x_i \in \{0, 1\}$

cost : $C\hat{}$ → with fractional.

Bound : ub → without Fractional
↓
used for branching

→ It is an maximization problem, but in branch and bound we can solve only minimization problem

→ If we want to solve a maximization problem, then convert problem into minimization (change the sign)

Page No.
Date :

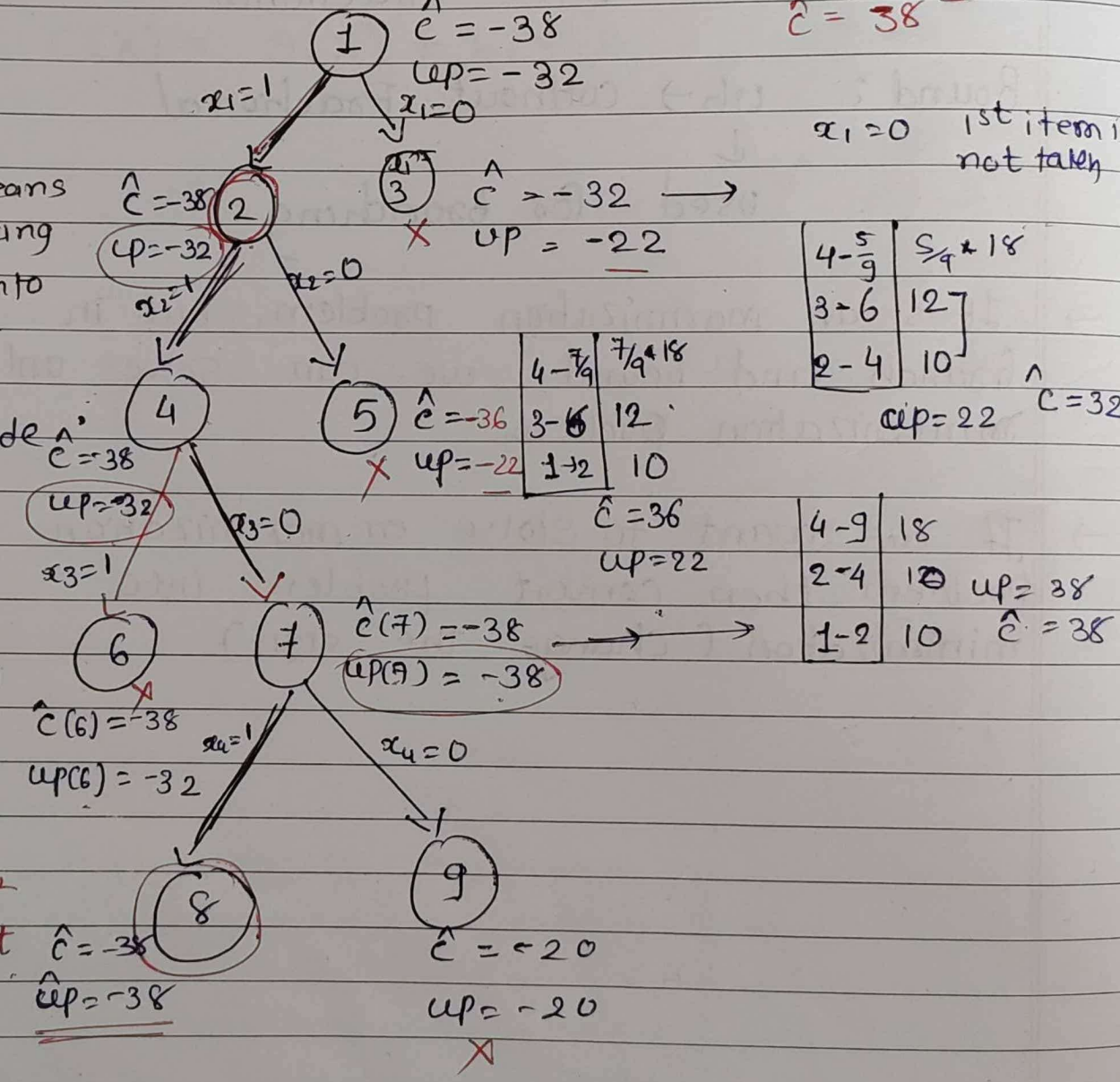
Example: $n = 4$, $m = 15$

i	1	2	3	4
P	10	10	12	18
w	2	4	8	9

Initial value of $\hat{c}$ and ub →

	p
4 → 3/9	3/9 * 18
3 → 6	12
2 → 4	10
1 → 2	10

up = 52
$\hat{c}$ = 38



maximum profit = 38

items = (1, 1, 0, 1)