Each service can be categorised by the quality of service. (QOS)
↳ reliable
↳ ~~non~~ unreliable

reliable service : applications like digitized voice traffic.

Applications like emails do not require connection (but it is reliable)

AI DEEPAK KHEMANI

8-Puzzle :

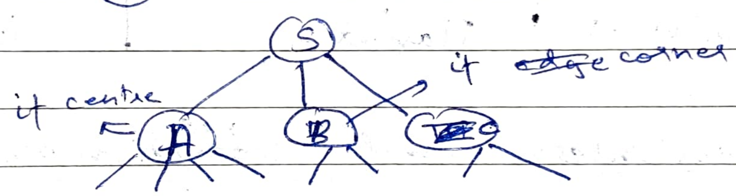
3	1	2
5	4	
6	7	8

S

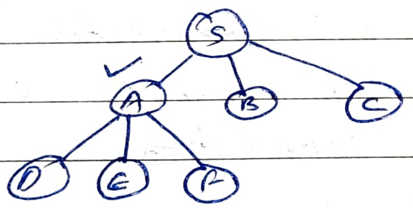
→

1	2	3
8		4
7	6	5

↳ Goal



OPEN SET : = {S} → {A, B, C} → {B, C, D, E, F}



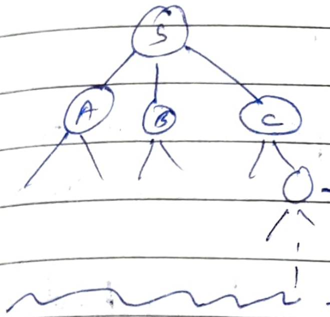
→ state space

1) OPEN ← {S}
2) WHILE OPEN NOT EMPTY :
3) { Pick any node n from OPEN : ~~OPEN ← OPEN~~
CLOSED = CLOSED ∪ {n}
OPEN = OPEN − {n}
if (n == GOAL) return TRUE
else
OPEN ← OPEN ∪ {neighbours (n) − Closed}
}
4) RETURN FAILURE

Page No.
Date

2) DFS
3) BFS

time complexity $\sim O(|\text{closed set}|)$



→ depth = 1
→ depth = 2
→ depth = 'd'

b = branching factor (avg. branches)

DFS time $= \frac{1}{2}\left[\text{Goal at first node at depth 'd'} + \text{Goal at last node at depth 'd'}\right]$

$$= \frac{1}{2}\left[d + 1 + b + b^2 + \dots + b^d\right]$$

$$= \frac{1}{2}\left[d + \frac{b^{d+1}-1}{b-1}\right] \approx \frac{b^d}{2} \quad \text{(Exponential)}$$

BFS time $= \frac{1}{2}\left[\text{all nodes till } (d-1)^{\text{th}} \text{ depth} + \text{Goal at } 1^{st} \text{ node} + \text{Goal at last node}\right]$

$$= \frac{1}{2}\left[\frac{b^d-1}{b-1} + \frac{b^{d+1}-1}{b-1}\right]$$

$$= \frac{b^d}{2}\left(\frac{1}{b}+1\right)$$

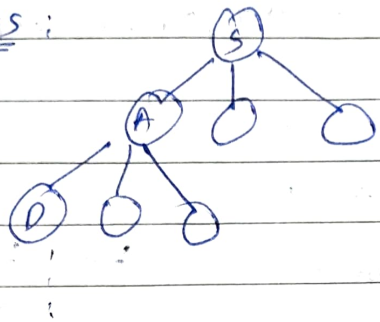
$$= \frac{b^d}{2}\left(\frac{1+b}{b}\right) \quad \text{(Exponential)}$$

⇒ DFS is better than BFS

$$\boxed{\frac{\text{BFS}}{\text{DFS}} = \frac{1+b}{b}}$$

Space Complexity → examine by open set.

DFS:



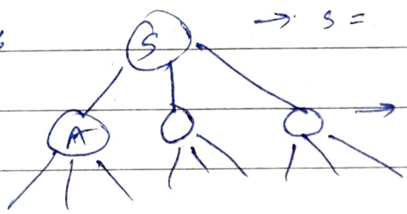
(b- 1)

$(b-1)$ space complexity = $(d-1)(b-1)$
(at random lvl d)

d^{th} lvl

$\approx O(b.d)$
(linear)

BFS:



→ s = b^0

→ b^1

→ b^2

at depth d → b^d
(Exponential)

∴ DFS is better than BFS in space complexity.

* Quality of solⁿ : BFS >> DFS
BFS gives optimal solⁿ.

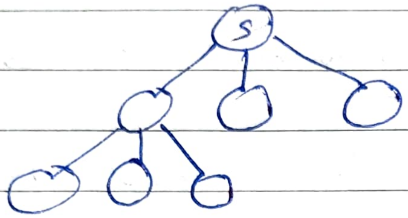
* **Completeness :**
⇒ Both will be able us to reach goal state. ∴ both are complete.

X—X

AI

* **DB-DFS**: Depth bound DFS

⇒ Set up a bound on depth (like d = 3, 4, etc)
DFS will only go till depth.

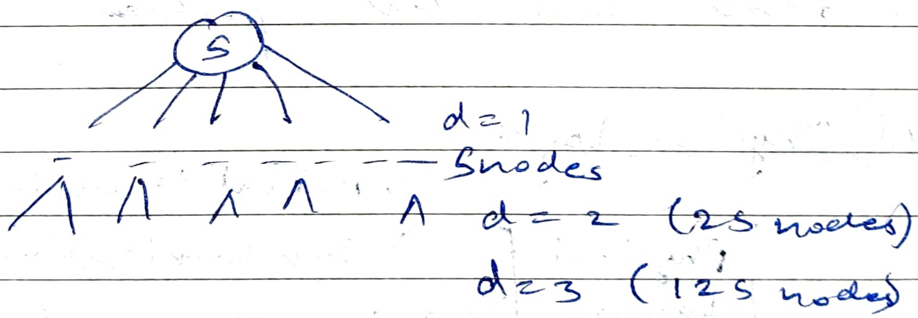


⇒ good quality of solⁿ
⇒ completeness ✗.

* **DFID**: Depth First iterative Deepening.

⇒ here depth bound ↑es ~~good~~ as we do not get the solⁿ
⇒ good quality of solⁿ
⇒ completeness ✓
⇒ Time and space complexity as DFS.

⇒ The tree gets deleted as we increse d.



⇒ To ~~see~~ check at d = 3 we need to check all nodes at d = 2

∴ ratio = $\frac{1+5+25}{125} = \frac{31}{125}$ (not that significant work)

∴ we can delete and regenerate the whole tree again.

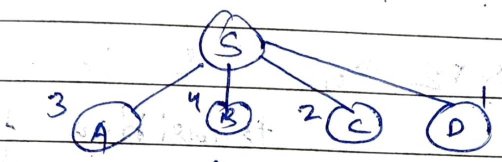
Page No.
Date

BFS, DFS, DB-DFS, DFID are blind algos as we don't know how close we are to the goal.

* **Heuristic Algorithms** : (we use heuristic f^ns to determine the node)

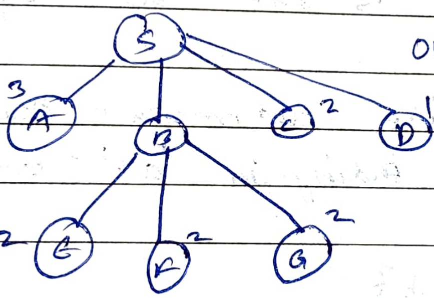
* **Best First Search** :

$h_1(n)$ = no. of correct nodes



OPEN = {A, B, C, D}
CLOSED = {S}

⇒ select B (B has max $h_1(n)$ in OPEN set)



OPEN = {A, C, D, E, F, G}
CLOSED = {S, B}

and so on...

⇒ Best way to ~~do~~ maintain open set is in max heap. (node with highest $h_1(n)$ will be at root)

⇒ time and space complexity depends on heuristic f^n.

⇒ the heuristic algo we used is complete.

x—x

AI

* **Genetic Algorithm:**

a. selection b. cross-over c. Mutation → uses random soln.

e.g.

x		eval f^n (x^2)	Prob (4 outcomes)
13	01101	169	$4 \times 169/1170 \approx 0.8$ — 1
24	11000	576	$4 \times 576/1170 \approx 1.67$ — 2
8	01000	64	$4 \times 64/1170 \approx 0$ — x
19	10011	361	$4 \times 361/1170 \approx 1.2$ — 1
		1170	↑ selection

$\boxed{\begin{array}{l} 13 \rightarrow 01101 \\ 24 \rightarrow 11000 \end{array}}$ $\boxed{\begin{array}{l} 24 \rightarrow 11000 \\ 19 \rightarrow 10011 \end{array}}$ → cross over (single point)

01100 (12) 11001 (25) → 2 11011 (27) → 2 10000 (16)

$\boxed{\begin{array}{l} 25 \rightarrow 11001 \\ 27 \rightarrow 11011 \end{array}}$ $\boxed{\begin{array}{l} 27 \rightarrow 11011 \\ 25 \rightarrow 11001 \end{array}}$ → 3rd bit in both candidate is zero. so we will never reach the answer. ∴ we use mutation

mutation: change one bit in order to get to the answer.

$\boxed{\begin{array}{l} 11\underline{1}11 \\ 11\underline{1}01 \end{array}}$

* **TSP:**

using Gen. Algo.

Page No.
Date

P_1 = 2 4 7 5 6 1 8 9 3 } path representation
P_2 = 6 2 8 3 9 1 4 5 7

1) Partially Mapped Crossover :

P_1 = 2 4 7 | 5 6 1 8 | 9 3
P_2 = 6 2 8 | 3 9 1 4 | 5 7

C_1 = 5 6 1 8 → mapping (if value already there then use the mapped value)
C_2 = 3 9 1 4

C_1 = 9 2 4 5 6 1 8 → 4 as 8 → 4 (6 2 8)
↳ 6 → 9 ∴ 9 from 6 2 8

∴ C_1 = $\underbrace{9\ 2\ 4}_{P_2}$ $\underbrace{5\ 6\ 1\ 8}_{P_1}$ $\underbrace{3\ 7}_{P_2}$

C_2 = $\underbrace{2\ 8\ 7}_{P_1}$ $\underbrace{3\ 9\ 1\ 4}_{P_2}$ $\underbrace{6\ 5}_{P_1}$

2) Ordered Crossover :

P_1 = 2 ~~4~~ ~~7~~ | 5 6 ~~1~~ 8 | ~~9~~ ~~3~~
P_2 = ~~6~~ 2 ~~8~~ | 3 9 ~~1~~ 4 | ~~5~~ 7

C_1 = $\underbrace{5\ 6\ 1\ 8}_{P_1}$ $\underbrace{2\ 3\ 9\ 4\ 7}_{P_2}$

C_2 = $\underbrace{3\ 9\ 1\ 8}_{P_2}$ $\underbrace{2\ 7\ 5\ 6\ 8}_{P_1}$

* Ordinal Representation

↳ study on your own.

INDEX = 1 2 3 4 5 6 7 8 9
P_1 = 2 3 5 3 | 3 1 2 2 1
P_2 = 6 2 6 2 | 5 1 1 1 1

$c_1 = \underbrace{2\ 3\ 5\ 3}_{p_1}\ .\ \underbrace{5\ 1\ 1\ 1\ 1}_{p_2}$
$c_2 = \underbrace{6\ 2\ 6\ 2}_{p_2}\ \ \underbrace{3\ 1\ 2\ 2\ 1}_{p_1}$
} using single pt. crossover

$c_1 = 2\ 4\ 7\ 5\ 9\ 1\ 3\ 6\ 8$ } path rep.

TOC

* Thm: For every NDFA, $\exists$ a DFA which simulates the behaviour of NDFA. Alternatively if L is the set accepted by NDFA, then $\exists$ a DFA that accepts L

Proof:
Let $M = (Q, \Sigma, \delta, q_0, F)$ be a NDFA accepting L. We ~~construct~~ construct a DFA M' as:

$M' = (Q', \Sigma, \delta', q_0', F')$ where:

(i) $Q' = 2^Q$ (any state in Q' is denoted by $[q_1, q_2, \ldots q_i]$ where $q_1, q_2, \ldots\ q_i \in Q$)
(ii) $q_0' = [q_0]$
(iii) F' is the set of all subsets of Q containing an element of F.

Before defining δ', we look at the construction of Q', q_0' and F. M is initially at q_0. But on application of an i/p symbol, say 'a', M can reach any of the states $\delta(q_0, a)$

To describe M, just after the i/p symbol 'a',