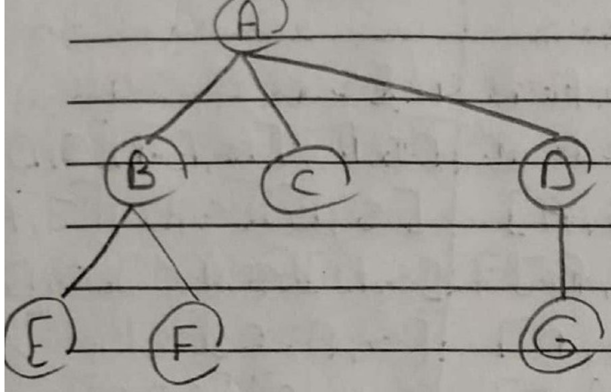
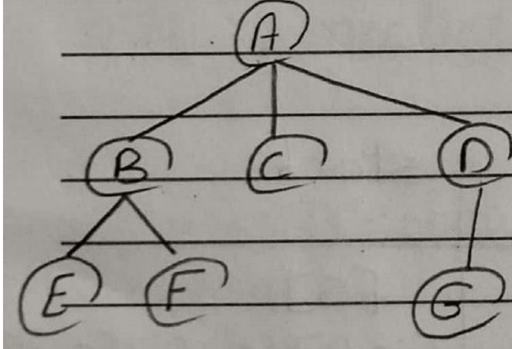
Date / / Object Lec 2

* Breadth First traversal : (BFt)



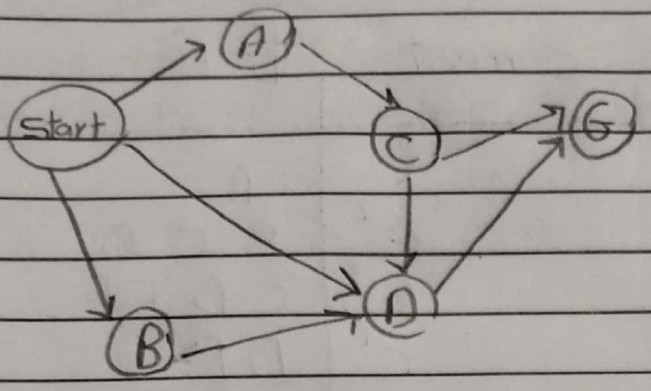
current	queue
	A
A	B C D
B	C D E F
C	D E F
D	E F G
E	F G
F	G
G	

* DePth First traversal:(DFt)



current	stack
	A
A	B C D
D	B C G
G	B C
C	B
B	E F
F	E
E	—

Breadth
* ~~Depth~~ first search: (BFS)



goal

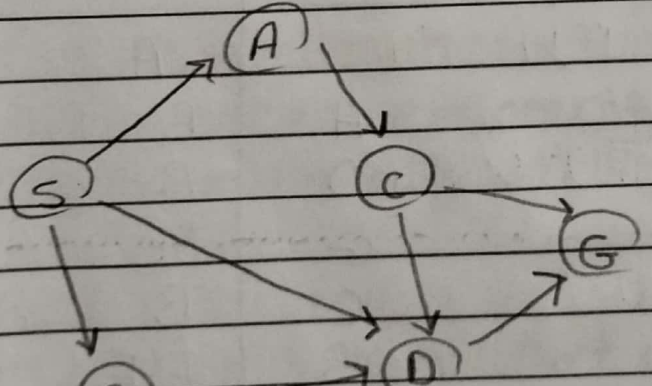
(S → D → G)

Path	queue
	[S]
[S]	[S,A] [S,B] [S,D]
[S,A]	[S,B] [S,D] [S,A,C
[S,B]	[S,D] [S,A,C] [S,B,D]
[S,D]	[S,A,C] [S,B,D] [S,A
[S,A,C]	[S,B,D] [S,D,G]
	[S,A,C,D] [S,A,C,G]
[S,D,G]	—

دا الحل بتاعي فخلاص مش مكمل → [S,D,G]

لما جاي انزل (S,B,D) هلاقي اني نزلت ال D قبل كدا فهمسحها واسيبها

* Depth first search: (DFS)



(S → D → G)

current	stack
	[S]
[S]	[S,A] [S,B] [S,D]
[S,D]	[S,A] [S,B] [S,D,G]
[S,D,G]	—

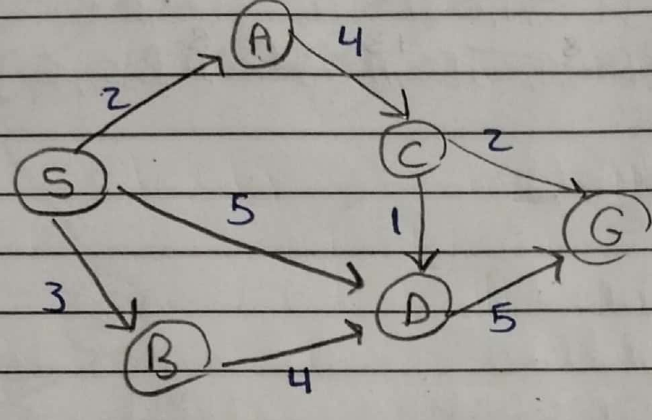
→ goal

Date / / Object

1- complete
2- optimal
3- Time & space complexity
$= o(b^{\text{ceilling}\backslash \varepsilon})$

uniform cost search:

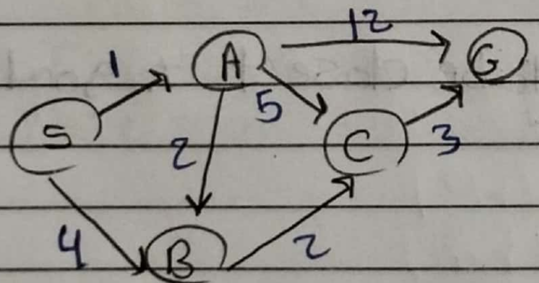
1. use a priority queue (least cost first)
2. pop element with least cost
3. if two element have same cost use alphabetic order



current	Proirity queue
	0 [S]
[S]	[S,A][S,B] [S,D]
[S,A]	[S,B][S,D][S,A,C]
[S,B]	$[S,\overset{5}{D}]$ $[S,\overset{6}{A},C]$ $[S,\overset{7}{B},D]$
[S,D]	$[S,\overset{6}{A},C]$ $[S,\overset{7}{B},D]$ ~~$[\overset{10}{S},B,G]$~~
[S,A,C]	~~[S,B,D]~~ $[S,\overset{10}{D},G]$ ~~[S,A,C,D]~~
	$[S,A,\overset{8}{C},G]$
[S,A,C,G]	

cost = 2+4+2 = 8 ← [S,A,C,G]

A* search: goal (S → G)



Node	H
S	7
A	6
B	4
C	2
G	0

G-cost = cost to move from S → A → C = 1+5 = 6
H-cost = H value for last node in the path (C) from table = 2
F-cost = G-cost + H-cost
= 6 + 2 = 8

* combines uniform cost & gready

* A* finds the optimal path
* idea → avoid expending paths that already expensive

K.M.S
NOTEBOOK & STATIONARY

Date / / Object

F.cost يستخدم A*search في

current	Priority queue
	[$\overset{7}{S}$]
[S]	[S,$\overset{7}{A}$] [S,$\overset{8}{B}$]
[S,A]	[S,$\overset{8}{B}$] [S,$\overset{7}{A}$,B] [S,$\overset{8}{A}$,C] [S,A,$\overset{13}{G}$]
[S,A,B]	[S,$\overset{8}{B}$] [S,$\overset{8}{A}$,C] [S,$\overset{13}{A}$,G] [$\overset{7}{S}$,A,B,C]
[S,A,B,C]	~~[S,$\overset{8}{B}$]~~ ~~[S,$\overset{8}{A}$,C]~~ [S,$\overset{13}{A}$,G] [S,$\overset{8}{A}$,B,C,G]
[S,A,B,C,G]	

goal →

H.cost من الجدول

G.cost = 8
H.cost = 0
F cost = 8+0=8

Greedy best first search :.

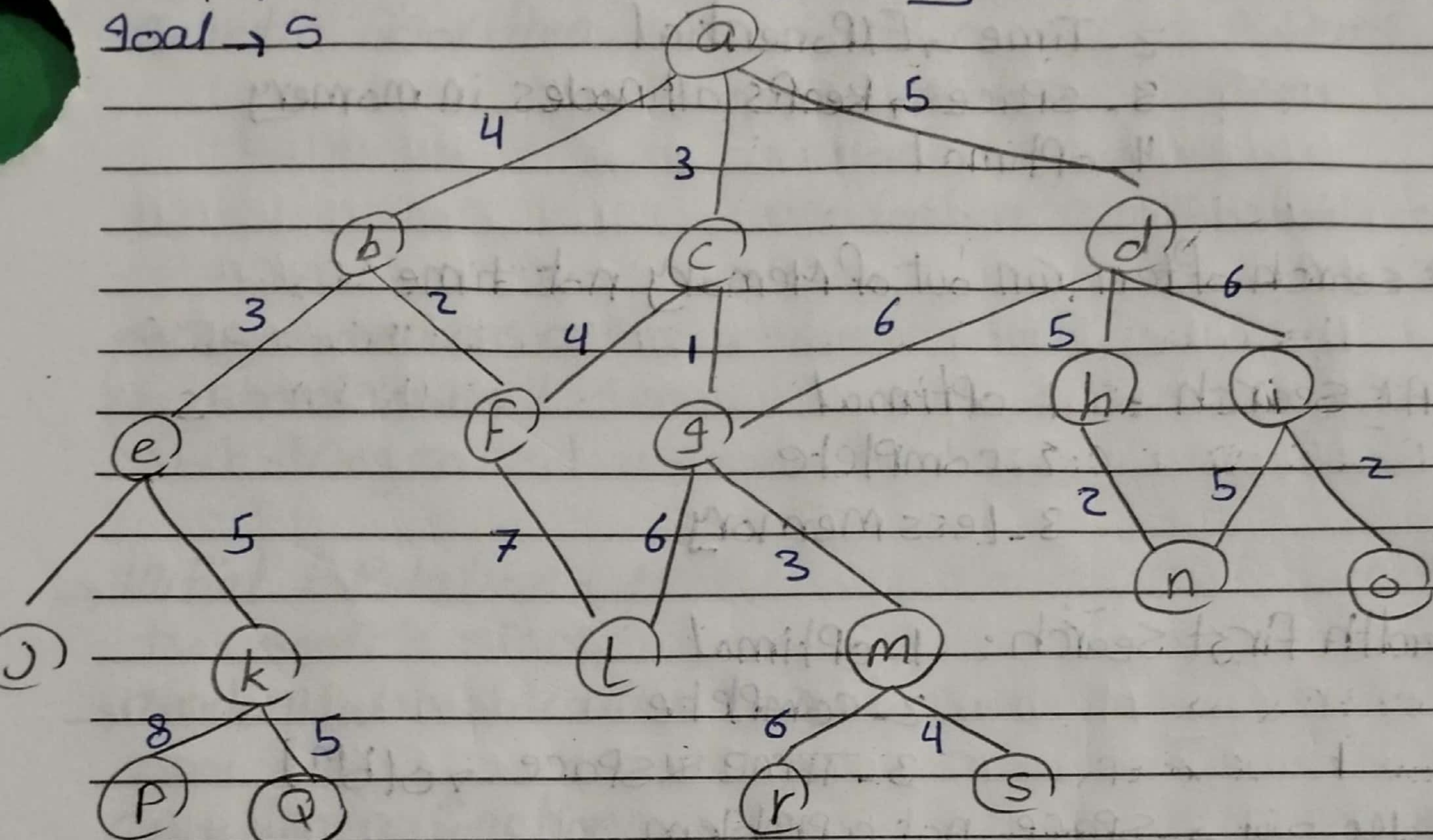
* Evaluation function f(n) = h(n) → cost from n to goal
* ignore the Path cost
* Expends the node that appears to be closest to goal

~~Last~~
1. not complete
2. not optimal
3. Time complexity → $O(b^m)$ m → Max depth of search

Date / / Object

Hill_climbing search

start → a
goal → s



[a]	$[a,\overset{4}{b}][a,\overset{3}{c}][a,\overset{5}{d}]$,
[a,c]	$[\overset{4}{a},b][\overset{3}{a},d]\,[a,c,\overset{1}{g}]\,[a,c,\overset{4}{F}]$
[a,c,g]	$[a,b][a,d][a,c,F][a,c,g,\overset{6}{L}][a,c,g,\overset{3}{m}$
[a,c,g,m]	[a,c,g,m,s][a,c,g,m,r]
[a,c,g,m,s]	

↓
goal

بعمل sort للاجيال الجديدة فقط وبكنسل خلاص القديمة

* لو الجول هو النود الاعلى يبقا التانت وبعدين أكمل back tracking وأرجع للجول

similar to Greedy search in h()

A* search : 1- Complete
2- Time → Exponential
3- space → keeps all nodes in momery
4- optimal

* A* search often run out of Memory not time

IDA* search : 1- optimal
2- complete
3- Less memory

Breadth first Search : 1- optimal
2- complete
3- Time & space → $O(b^m)$

4- بنستخدمها → - space not a problem
- find solution with fewest arcs
- there are shallow solutions
- infinite paths

Depth First search : 1 not optimal
2. not complete
3- time → $O(b^m)$
4. space → $O(bm)$

5- بنستخدمها → - many solution with long path length
- space is restricted محدودة

6- Poor method when → - graph cycles
- sparse solution at shallow path

- there huristic knowledge indicating when one path is Better than another

Genetic Algorithm: uses 1- Defficult search Problems
2- optimization Problems
3- Machine learning
4- adaptive rule ~~Passbat~~ bases

string sometimes called chromosomes
↳ letters called genes
↳ bit string called individuals

→ initial Population: Must be arepresentative sample of the search space

→ random initialization can be agood idea if sample is large

GA: fitness function evaluates each solution & decide it will be in next generation of solution

*Common selection Methods used in GA serch:
1- fitness Proportionate selection
2- Rank selection
3- tournament selection

Rank selection: All individual are sorted according To their fitness, Each individual is then assigned a Probability of being selected from some Prior Probability density.