Remember: Implementation of search algorithms

```
Function General-Search(problem, Queuing-Fn) returns a solution, or failure
    nodes ←  make-queue(make-node(initial-state[problem]))
    loop do
      if nodes is empty then return failure
      node ←  Remove-Front(nodes)
      if Goal-Test[problem] applied to State(node) succeeds then return node
      nodes ←  Queuing-Fn(nodes, Expand(node, Operators[problem]))
    end
```

Queuing-Fn(queue, elements) is a queuing function that inserts a set of elements into the queue and determines the order of node expansion. Varieties of the queuing function produce varieties of the search algorithm.

From session02-04.pptx, slide 83

Remember: Implementation of search algorithms

```
Function General-Search(problem, Queuing-Fn) returns a solution, or failure
   nodes ← make-queue(make-node(initial-state[problem]))
   loop do
    if nodes is empty then return failure
    node ← Remove-Front(nodes)
    if Goal-Test[problem] applied to State(node) succeeds then return node
    nodes ← Queuing-Fn(nodes, Expand(node, Operators[problem]))
   end
```

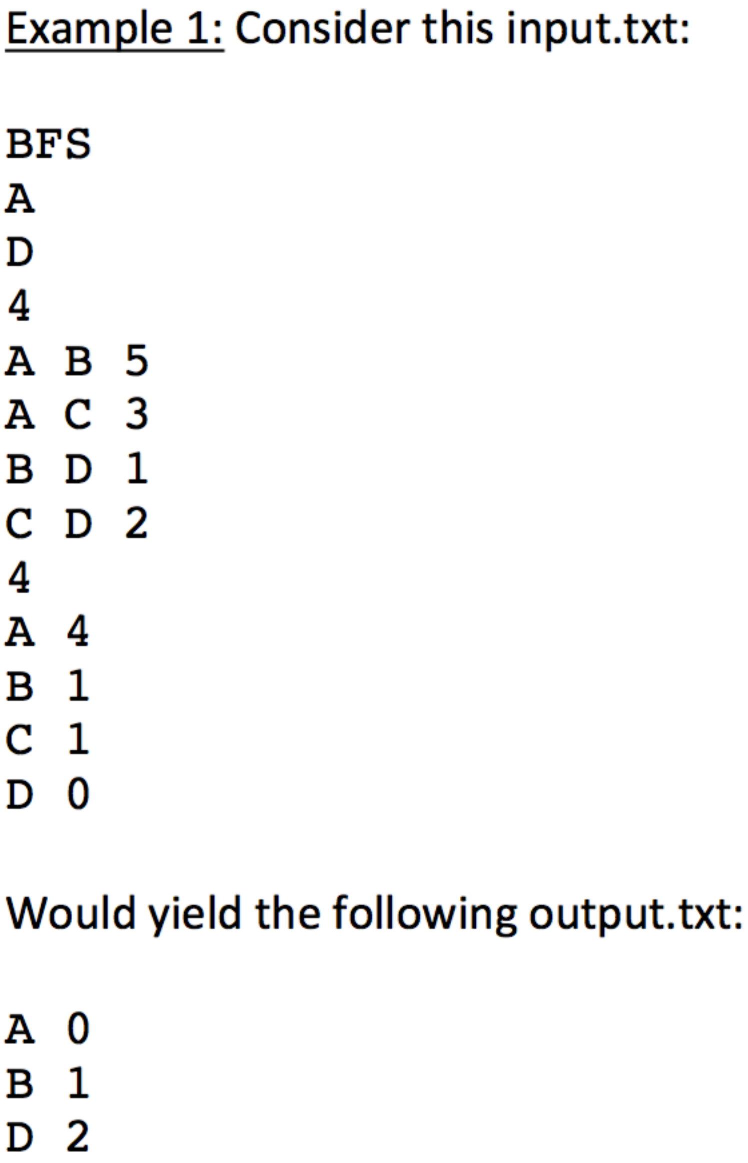
Queuing-Fn(queue, elements) is a queuing function that inserts a set of elements into the queue and determines the order of node expansion. Varieties of the queuing function produce varieties of the search algorithm.

From session02-04.pptx, slide 83

Adding tie-breaking

```
Function General-Search(problem, Queuing-Fn) returns a solution, or failure
    nodes ←   make-queue(make-node(initial-state[problem]))
    loop do
      if nodes is empty then return failure
      node ←   Remove-Front(nodes)
      if Goal-Test[problem] applied to State(node) succeeds then return node
      nodes ←   Queuing-Fn(nodes, TieBreak(Expand(node, Operators[problem])))
    end
```

TieBreak(nodeset) – homework 1 text says (p. 5):
"If all else is equal while searching routes (ties), you should explore (enqueue) multiple paths from the same intersection **in the order** in which they are listed in the **live traffic** inputs. "



Example 1: Consider this input.txt:

```
BFS
A
D
4
A B 5
A C 3
B D 1
C D 2
4
A 4
B 1
C 1
D 0
```

Would yield the following output.txt:

```
A 0
B 1
D 2
```

Order in which paths from each state are Listed:

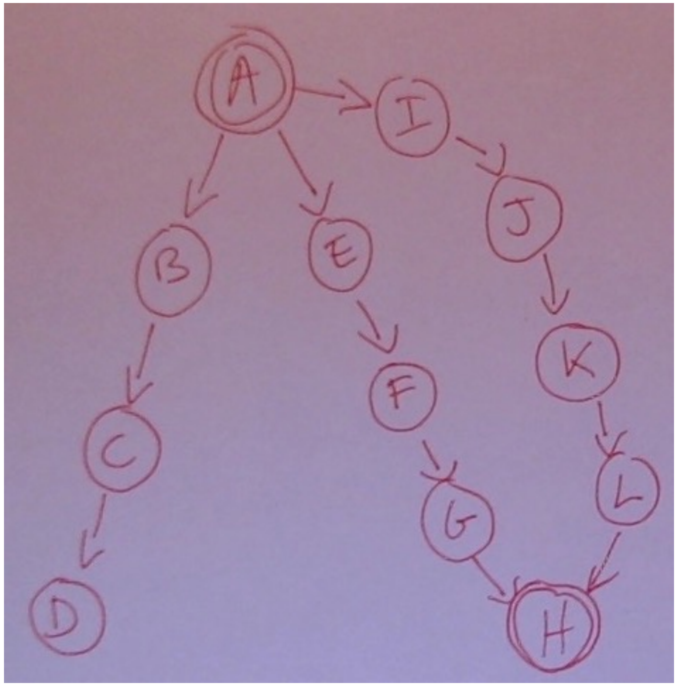
From A: A->B, A->C
From B: B->D
From C: C->D

This is why BFS returns ABD and not ACD.

Note: loop detection also contributed Here by preventing the second encountered Node with state D from being enqueued.

New hint: DFS example

```
DFS
A
H
12
A B 1
B C 1
C D 1
A E 10
E F 2
F G 3
G H 3
A I 1
I J 1
J K 1
K L 1
L H 1
[…]
```



Order in which paths from each state are listed:

From A: A->B, A->E, A->I

From B: B->C

From C: C->D

From E: E->F

From F: F->G

From G: G->H

From I: I->J

From J: J->K

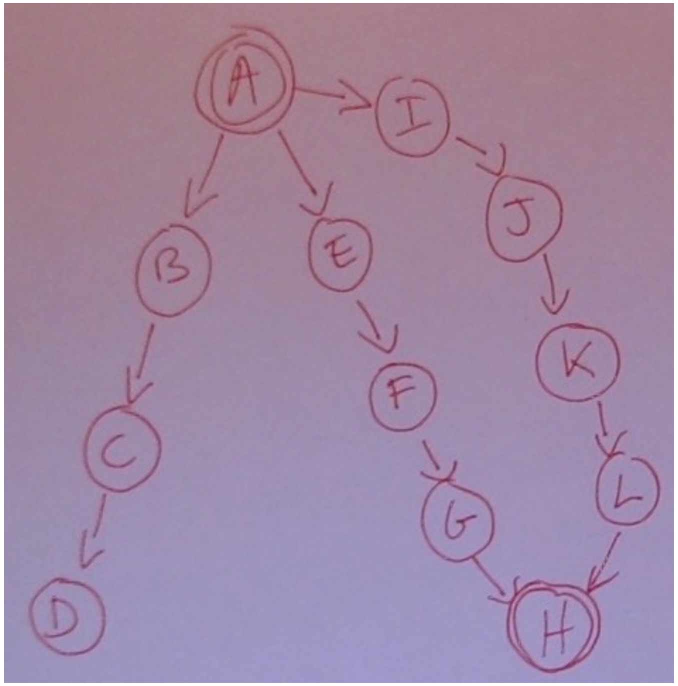
From K: K->L

From L: L->H

DFS returns?

New hint: DFS example

```
DFS
A
H
12
A B 1
B C 1
C D 1
A E 10
E F 2
F G 3
G H 3
A I 1
I J 1
J K 1
K L 1
L H 1
[…]
```



Order in which paths from each state are listed:

From A: A->B, A->E, A->I

From B: B->C

From C: C->D

From E: E->F

From F: F->G

From G: G->H

From I: I->J

From J: J->K

From K: K->L

From L: L->H

Node	State	Parent
1	A	NIL

New hint: DFS example

Order in which paths from each state are listed:

From A: A->B, A->E, A->I

From B: B->C

From C: C->D

From E: E->F

From F: F->G

From G: G->H

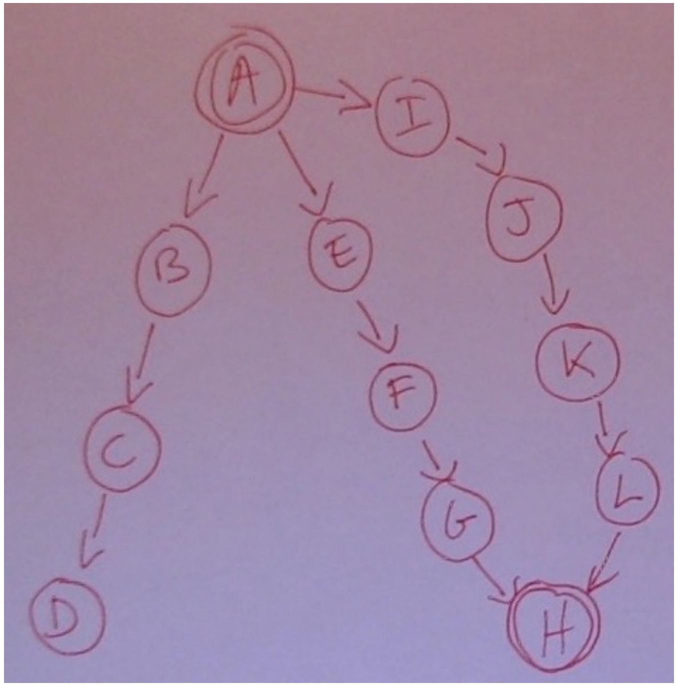
From I: I->J

From J: J->K

From K: K->L

From L: L->H

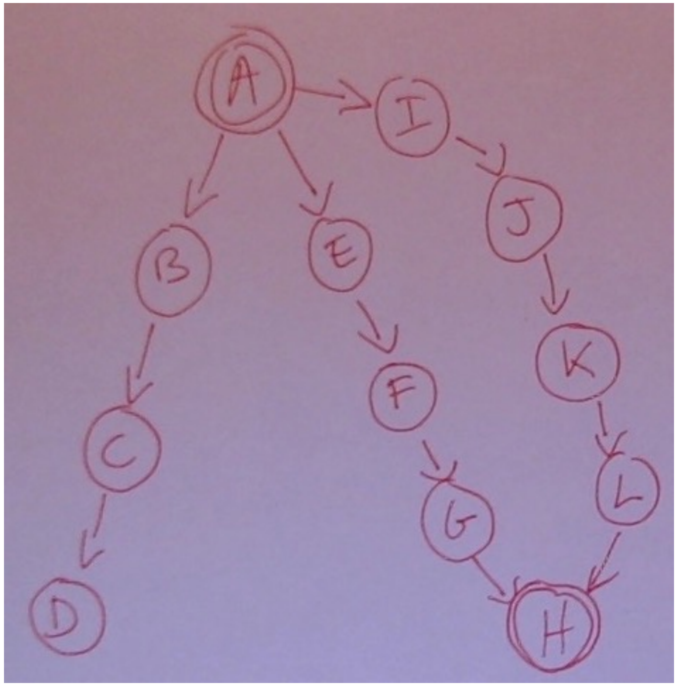
```
DFS
A
H
12
A B 1
B C 1
C D 1
A E 10
E F 2
F G 3
G H 3
A I 1
I J 1
J K 1
K L 1
L H 1
[…]
```



Node	State	Parent
2	B	1
3	E	1
4	I	1
1	A	NIL

New hint: DFS example

```
DFS
A
H
12
A B 1
B C 1
C D 1
A E 10
E F 2
F G 3
G H 3
A I 1
I J 1
J K 1
K L 1
L H 1
[…]
```



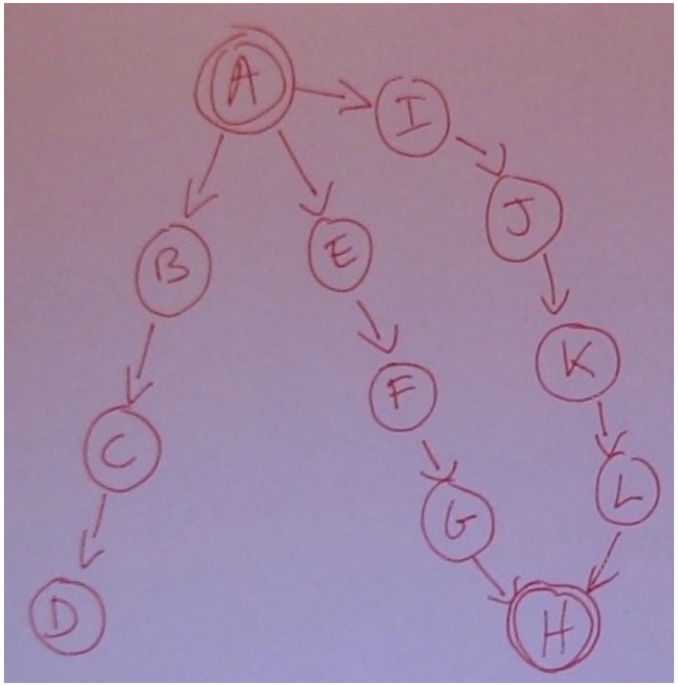
Order in which paths from each state are listed:

From A: A->B, A->E, A->I
From B: B->C
From C: C->D
From E: E->F
From F: F->G
From G: G->H
From I: I->J
From J: J->K
From K: K->L
From L: L->H

Node	State	Parent
5	C	2
2	B	1
3	E	1
4	I	1
1	A	NIL

New hint: DFS example

```
DFS
A
H
12
A B 1
B C 1
C D 1
A E 10
E F 2
F G 3
G H 3
A I 1
I J 1
J K 1
K L 1
L H 1
[…]
```



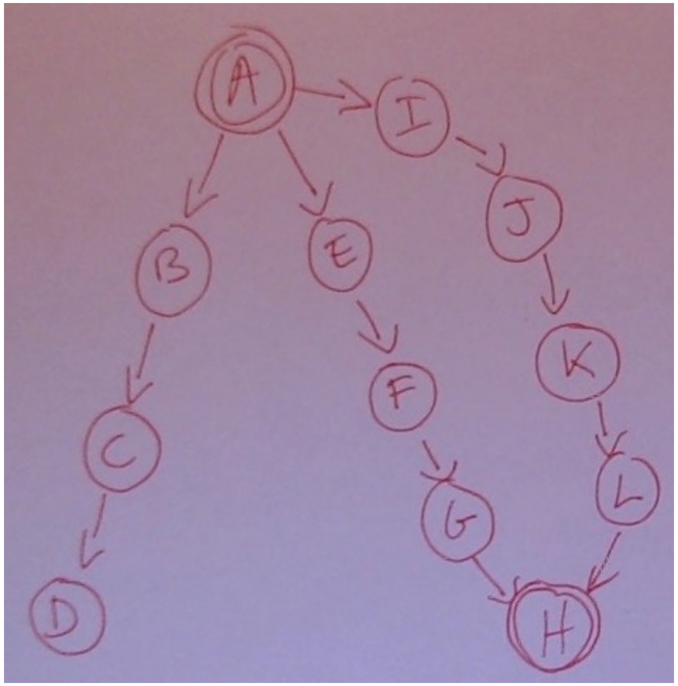
Order in which paths from each state are listed:

From A: A->B, A->E, A->I
From B: B->C
From C: C->D
From E: E->F
From F: F->G
From G: G->H
From I: I->J
From J: J->K
From K: K->L
From L: L->H

Node	State	Parent
6	D	5
5	C	2
2	B	1
3	E	1
4	I	1
1	A	NIL

New hint: DFS example

```
DFS
A
H
12
A B 1
B C 1
C D 1
A E 10
E F 2
F G 3
G H 3
A I 1
I J 1
J K 1
K L 1
L H 1
[…]
```



Order in which paths from each state are listed:

From A: A->B, A->E, A->I
From B: B->C
From C: C->D
From E: E->F
From F: F->G
From G: G->H
From I: I->J
From J: J->K
From K: K->L
From L: L->H

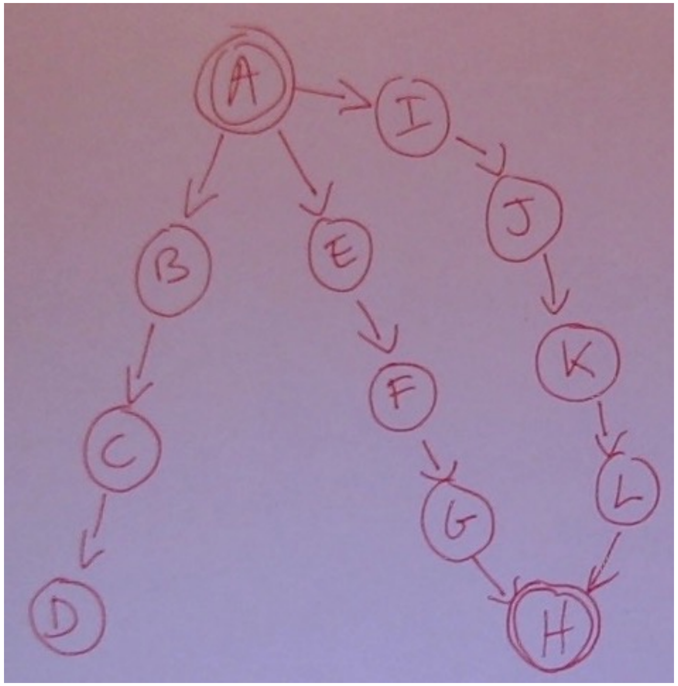
Node	State	Parent
6	D	5
5	C	2
2	B	1
3	E	1
4	I	1
1	A	NIL

New hint: DFS example

Order in which paths from each state are listed:

From A: A->B, A->E, A->I
From B: B->C
From C: C->D
From E: E->F
From F: F->G
From G: G->H
From I: I->J
From J: J->K
From K: K->L
From L: L->H

```
DFS
A
H
12
A B 1
B C 1
C D 1
A E 10
E F 2
F G 3
G H 3
A I 1
I J 1
J K 1
K L 1
L H 1
[…]
```



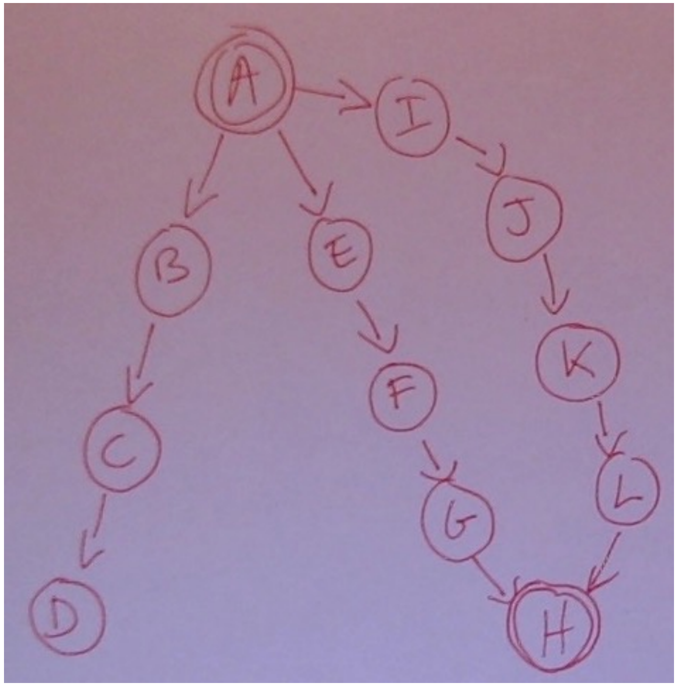
Node	State	Parent
7	F	3
6	D	5
5	C	2
2	B	1
3	E	1
4	I	1
1	A	NIL

New hint: DFS example

Order in which paths from each state are listed:

From A: A->B, A->E, A->I
From B: B->C
From C: C->D
From E: E->F
From F: F->G
From G: G->H
From I: I->J
From J: J->K
From K: K->L
From L: L->H

```
DFS
A
H
12
A B 1
B C 1
C D 1
A E 10
E F 2
F G 3
G H 3
A I 1
I J 1
J K 1
K L 1
L H 1
[…]
```



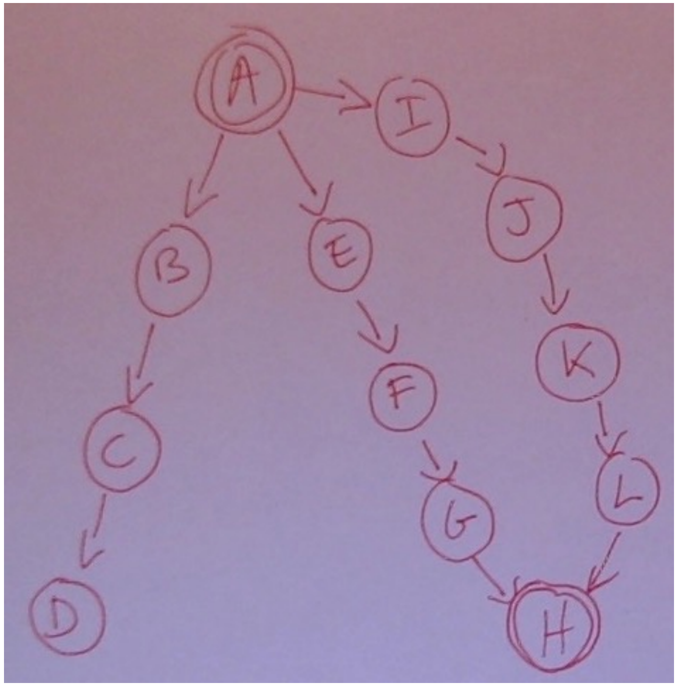
Node	State	Parent
8	G	7
7	F	3
6	D	5
5	C	2
2	B	1
3	E	1
4	I	1
1	A	NIL

New hint: DFS example

Order in which paths from each state are listed:

From A: A->B, A->E, A->I
From B: B->C
From C: C->D
From E: E->F
From F: F->G
From G: G->H
From I: I->J
From J: J->K
From K: K->L
From L: L->H

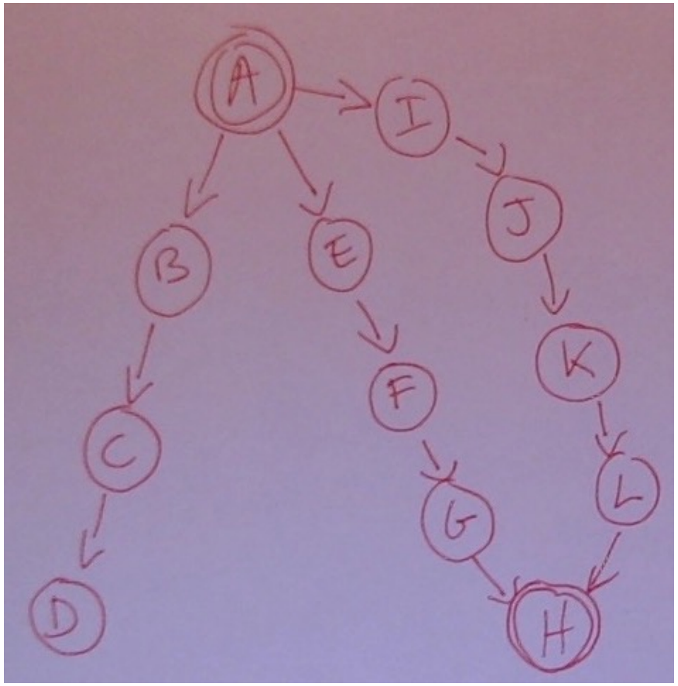
```
DFS
A
H
12
A B 1
B C 1
C D 1
A E 10
E F 2
F G 3
G H 3
A I 1
I J 1
J K 1
K L 1
L H 1
[…]
```



Node	State	Parent
9	H	8
8	G	7
7	F	3
6	D	5
5	C	2
2	B	1
3	E	1
4	I	1
1	A	NIL

New hint: DFS example

```
DFS
A
H
12
A B 1
B C 1
C D 1
A E 10
E F 2
F G 3
G H 3
A I 1
I J 1
J K 1
K L 1
L H 1
[…]
```



Order in which paths from each state are listed:

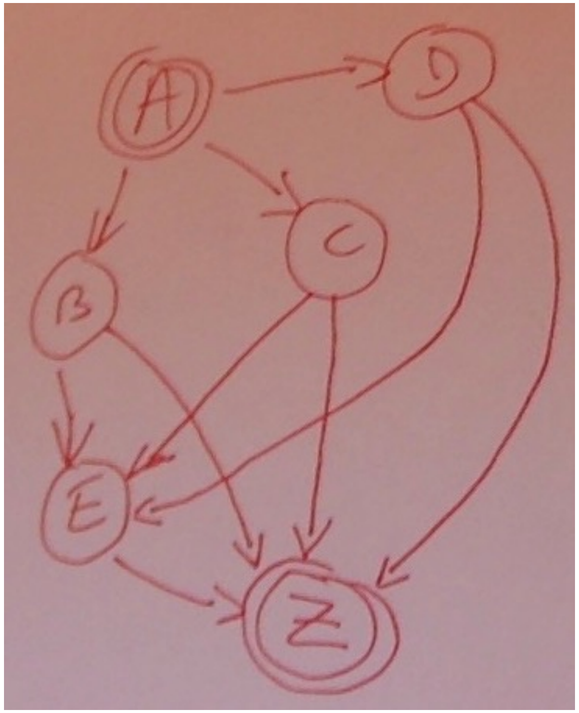
From A: A->B, A->E, A->I
From B: B->C
From C: C->D
From E: E->F
From F: F->G
From G: G->H
From I: I->J
From J: J->K
From K: K->L
From L: L->H

Node	State	Parent
9	H	8
8	G	7
7	F	3
6	D	5
5	C	2
2	B	1
3	E	1
4	I	1
1	A	NIL

Solution:
A 0
E 1
F 2
G 3
H 4

New hint: another DFS example

DFS
Andy
Zoe
10
Andy Bill 4
Andy Claire 3
Andy Daniel 2
Bill Elaine 3
Bill Zoe 1
Claire Elaine 4
Claire Zoe 2
Daniel Elaine 2
Daniel Zoe 2
Elaine Zoe 2
[…]



Order in which paths from each state are listed:

From A: A->B, A->C, A->D
From B: B->E, B->Z
From C: C->E, C->Z
From D: D->E, D->Z
From E: E->Z

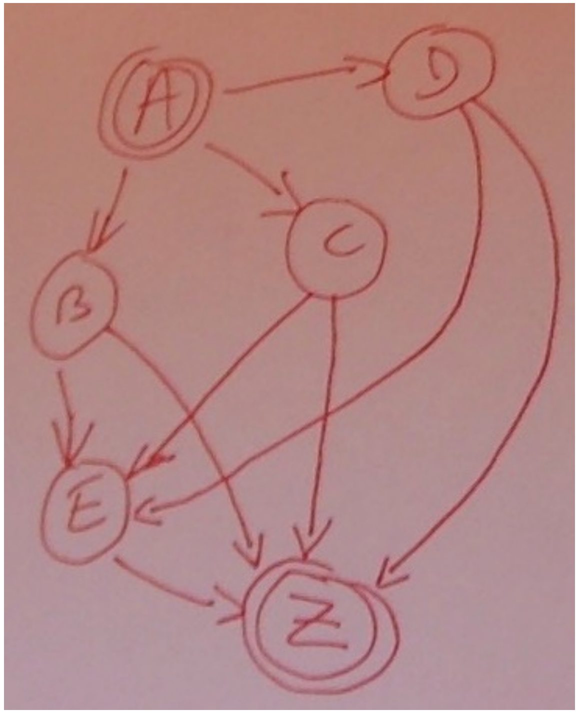
DFS returns?

New hint:
another DFS example

Order in which paths from each state are listed:

From A: A->B, A->C, A->D
From B: B->E, B->Z
From C: C->E, C->Z
From D: D->E, D->Z
From E: E->Z

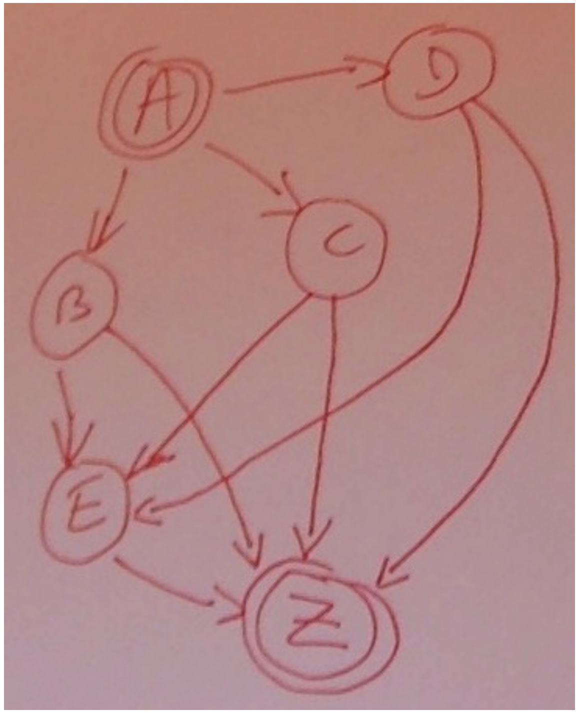
DFS
Andy
Zoe
10
Andy Bill 4
Andy Claire 3
Andy Daniel 2
Bill Elaine 3
Bill Zoe 1
Claire Elaine 4
Claire Zoe 2
Daniel Elaine 2
Daniel Zoe 2
Elaine Zoe 2
[…]



Node	State	g	Parent
1	A	0	NIL

New hint:
another DFS example

DFS
Andy
Zoe
10
Andy Bill 4
Andy Claire 3
Andy Daniel 2
Bill Elaine 3
Bill Zoe 1
Claire Elaine 4
Claire Zoe 2
Daniel Elaine 2
Daniel Zoe 2
Elaine Zoe 2
[…]



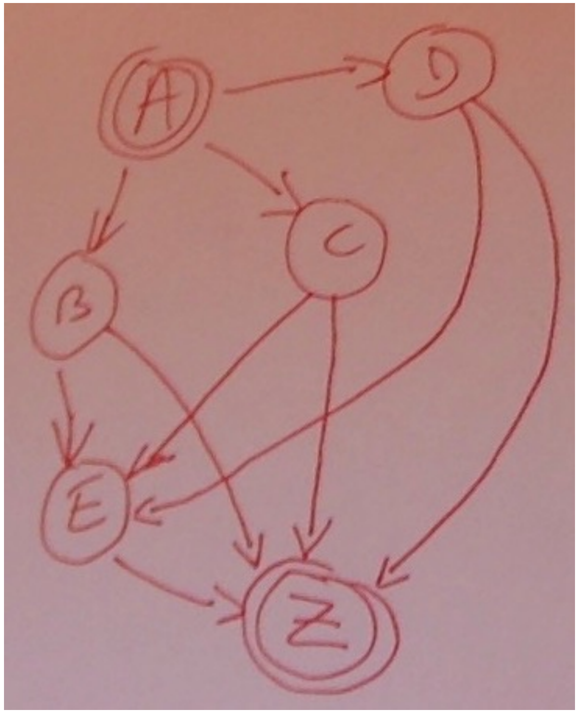
Order in which paths from each state are listed:

From A: A->B, A->C, A->D
From B: B->E, B->Z
From C: C->E, C->Z
From D: D->E, D->Z
From E: E->Z

Node	State	g	Parent
2	B	1	1
3	C	1	1
4	D	1	1
1	A	0	NIL

New hint: another DFS example

DFS
Andy
Zoe
10
Andy Bill 4
Andy Claire 3
Andy Daniel 2
Bill Elaine 3
Bill Zoe 1
Claire Elaine 4
Claire Zoe 2
Daniel Elaine 2
Daniel Zoe 2
Elaine Zoe 2
[…]



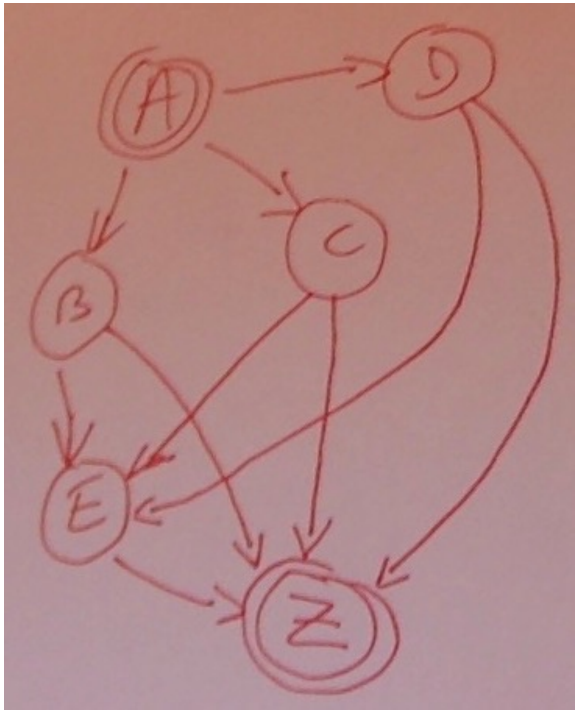
Order in which paths from each state are listed:

From A: A->B, A->C, A->D
From B: B->E, B->Z
From C: C->E, C->Z
From D: D->E, D->Z
From E: E->Z

Node	State	g	Parent
5	E	2	2
6	Z	2	2
2	B	1	1
3	C	1	1
4	D	1	1
1	A	0	NIL

New hint: another DFS example

DFS
Andy
Zoe
10
Andy Bill 4
Andy Claire 3
Andy Daniel 2
Bill Elaine 3
Bill Zoe 1
Claire Elaine 4
Claire Zoe 2
Daniel Elaine 2
Daniel Zoe 2
Elaine Zoe 2
[…]



Order in which paths from each state are listed:

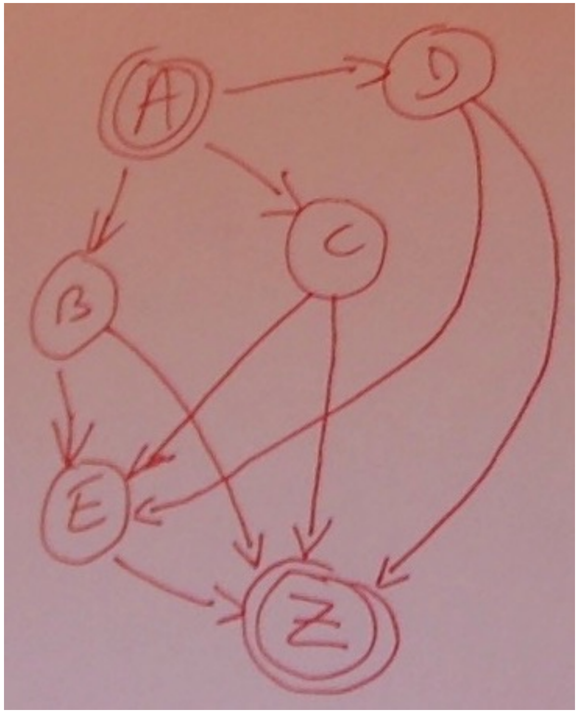
From A: A->B, A->C, A->D
From B: B->E, B->Z
From C: C->E, C->Z
From D: D->E, D->Z
From E: E->Z

Node	State	g	Parent
5	E	2	2
6	Z	2	2
2	B	1	1
3	C	1	1
4	D	1	1
1	A	0	NIL

Note: here we do not enqueue a node with state Z again since we already have one with lower cost in the open queue (loop detection).

New hint:
another DFS example

DFS
Andy
Zoe
10
Andy Bill 4
Andy Claire 3
Andy Daniel 2
Bill Elaine 3
Bill Zoe 1
Claire Elaine 4
Claire Zoe 2
Daniel Elaine 2
Daniel Zoe 2
Elaine Zoe 2
[…]



Order in which paths from each state are listed:

From A: A->B, A->C, A->D
From B: B->E, B->Z
From C: C->E, C->Z
From D: D->E, D->Z
From E: E->Z

Node	State	g	Parent
5	E	2	2
6	Z	2	2
2	B	1	1
3	C	1	1
4	D	1	1
1	A	0	NIL

Solution:
A 0
B 1
Z 2

Another hint / clarification

From homework 1 text, p. 3:

“Your program should write in output.txt the list of intersections/locations traveled over in your solution path, including the starting and finishing locations and the **accumulated** time from start to that intersection/location, in order of travel.”

Hint: For DFS, or for A* using a non-admissible heuristic: this may not be the optimal path.

Adding tie-breaking

```
Function General-Search(problem, Queuing-Fn) returns a solution, or failure
    nodes ←  make-queue(make-node(initial-state[problem]))
    loop do
      if nodes is empty then return failure
      node ←  Remove-Front(nodes)
      if Goal-Test[problem] applied to State(node) succeeds then return node
      nodes ←  Queuing-Fn(nodes, TieBreak(Expand(node, Operators[problem])))
    end
```

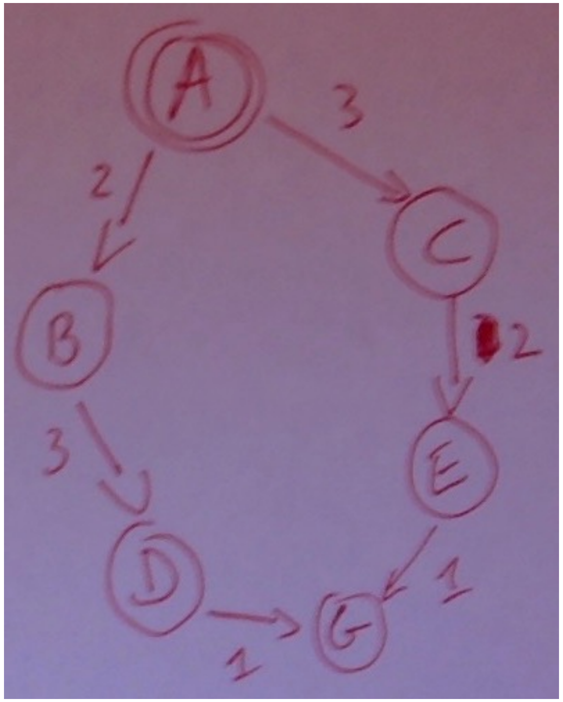
TieBreak(nodeset) – homework 1 text says (p. 5):
“If all else is equal while searching routes (ties), you should explore (enqueue) multiple paths from the same intersection **in the order** in which they are listed in the **live traffic** inputs. “

This can sometimes be under-specified (see next slide). Add this (with lower priority than the above rule):

“if all else is still equal, newly expanded nodes should be enqueued after (farther in the queue from the queue's front) older ones that were already in the queue.”

New hint: a UCS example

UCS
A
G
6
A B 2
A C 3
B D 3
C E 2
D G 1
E G 1
[…]



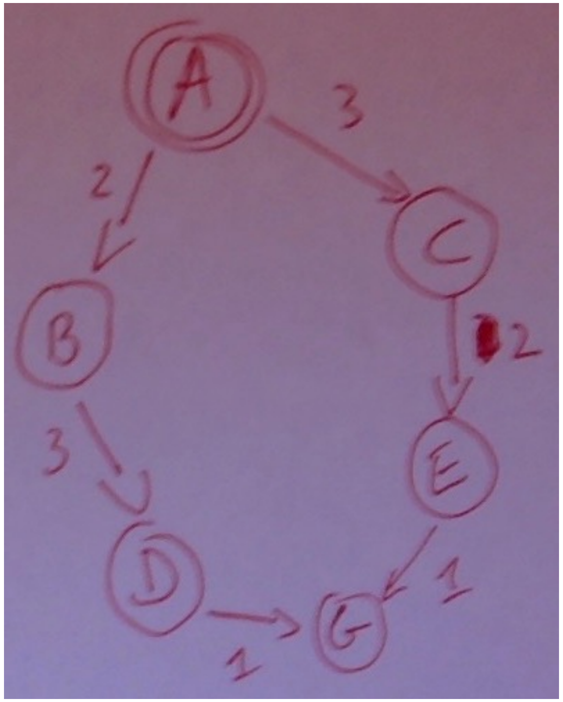
Order in which paths from each state are Listed:

From A: A->B, A->C, A->D
From B: B->D
From C: C->E
From D: D->G
From E: E->G

UCS returns?

New hint: UCS example

UCS
A
G
6
A B 2
A C 3
B D 3
C E 2
D G 1
E G 1
[...]



Order in which paths from each state are listed:

From A: A->B, A->C, A->D
From B: B->D
From C: C->E
From D: D->G
From E: E->G

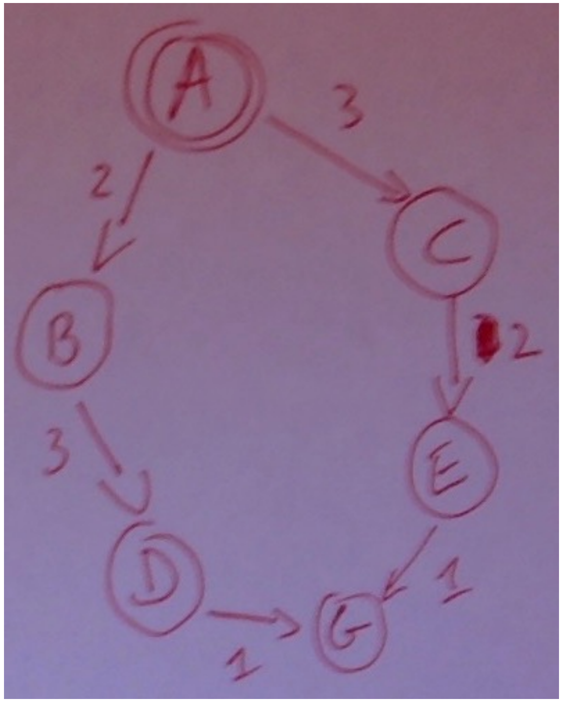
Node	**State**	**g**	**Parent**
1	A	0	NIL

New hint: UCS example

Order in which paths from each state are listed:

From A: A->B, A->C, A->D
From B: B->D
From C: C->E
From D: D->G
From E: E->G

```
UCS
A
G
6
A B 2
A C 3
B D 3
C E 2
D G 1
E G 1
[…]
```



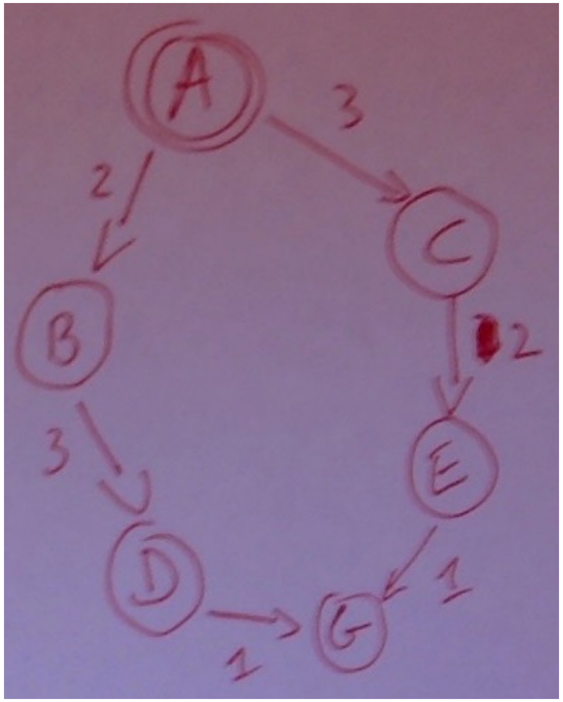
Node	State	g	Parent
1	A	0	NIL
2	B	2	1
3	C	3	1

New hint: UCS example

Order in which paths from each state are listed:

From A: A->B, A->C, A->D
From B: B->D
From C: C->E
From D: D->G
From E: E->G

```
UCS
A
G
6
A B 2
A C 3
B D 3
C E 2
D G 1
E G 1
[…]
```



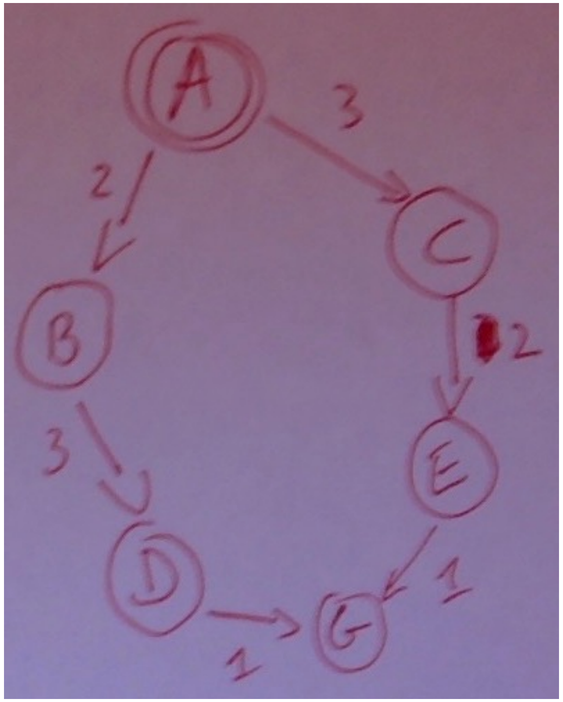
Node	State	g	Parent
1	A	0	NIL
2	B	2	1
3	C	3	1
4	D	5	2

New hint: UCS example

Order in which paths from each state are listed:

From A: A->B, A->C, A->D
From B: B->D
From C: C->E
From D: D->G
From E: E->G

```
UCS
A
G
6
A B 2
A C 3
B D 3
C E 2
D G 1
E G 1
[…]
```



Node	State	g	Parent
1	A	0	NIL
2	B	2	1
3	C	3	1
4	D	5	2
5	E	5	3

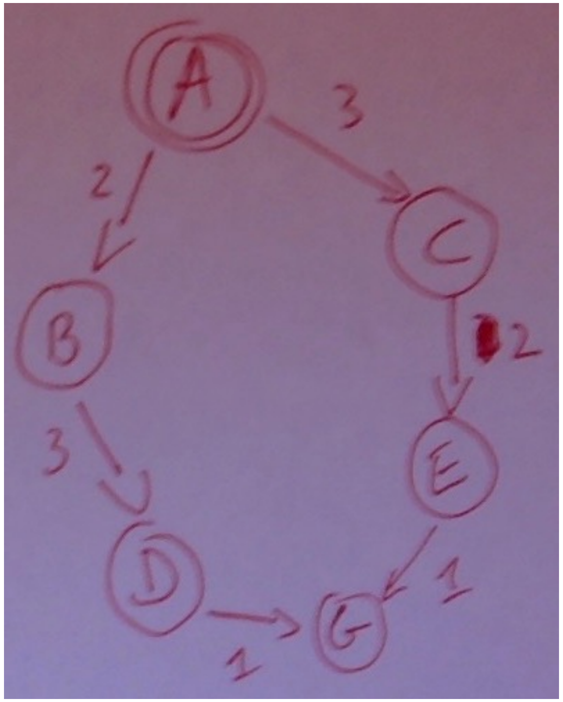
“if all else is still equal, newly expanded nodes should be enqueued after (farther in the queue from the queue's front) older ones that were already in the queue.”

New hint: UCS example

Order in which paths from each state are listed:

From A: A->B, A->C, A->D
From B: B->D
From C: C->E
From D: D->G
From E: E->G

UCS
A
G
6
A B 2
A C 3
B D 3
C E 2
D G 1
E G 1
[…]



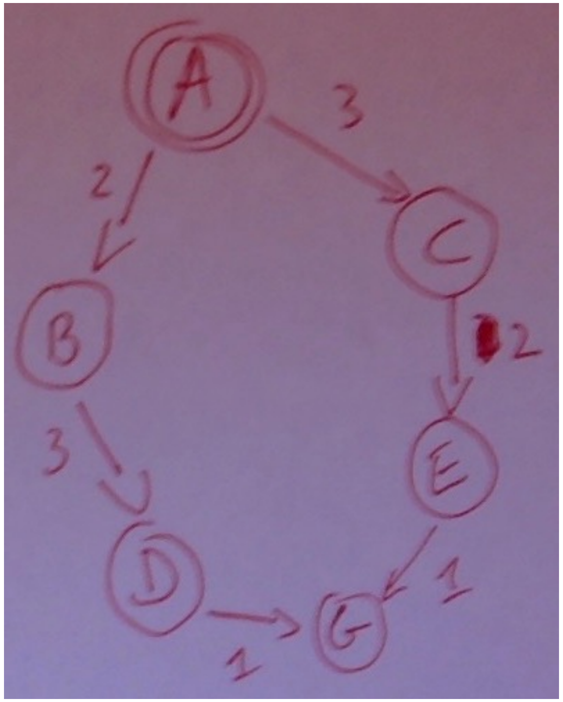
Node	State	g	Parent
1	A	0	NIL
2	B	2	1
3	C	3	1
4	D	5	2
5	E	5	3
6	G	6	4

New hint: UCS example

Order in which paths from each state are listed:

From A: A->B, A->C, A->D
From B: B->D
From C: C->E
From D: D->G
From E: E->G

```
UCS
A
G
6
A B 2
A C 3
B D 3
C E 2
D G 1
E G 1
[…]
```



Node	State	g	Parent
1	A	0	NIL
2	B	2	1
3	C	3	1
4	D	5	2
5	E	5	3
6	G	6	4

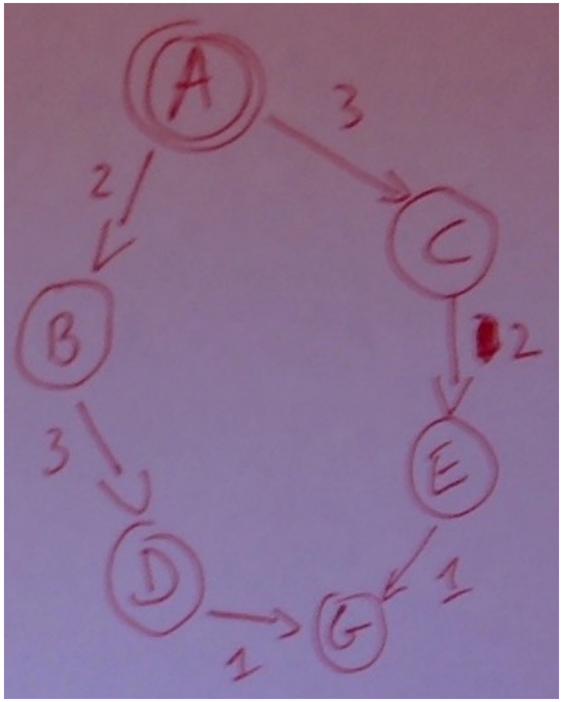
Note: here we do not enqueue a node with state G again since we already have one with same cost 6 in the open queue (loop detection).

New hint: UCS example

Order in which paths from each state are listed:

From A: A->B, A->C, A->D
From B: B->D
From C: C->E
From D: D->G
From E: E->G

UCS
A
G
6
A B 2
A C 3
B D 3
C E 2
D G 1
E G 1
[...]



Node	State	g	Parent
1	A	0	NIL
2	B	2	1
3	C	3	1
4	D	5	2
5	E	5	3
6	G	6	4

Solution:
A 0
B 2
D 5
G 6