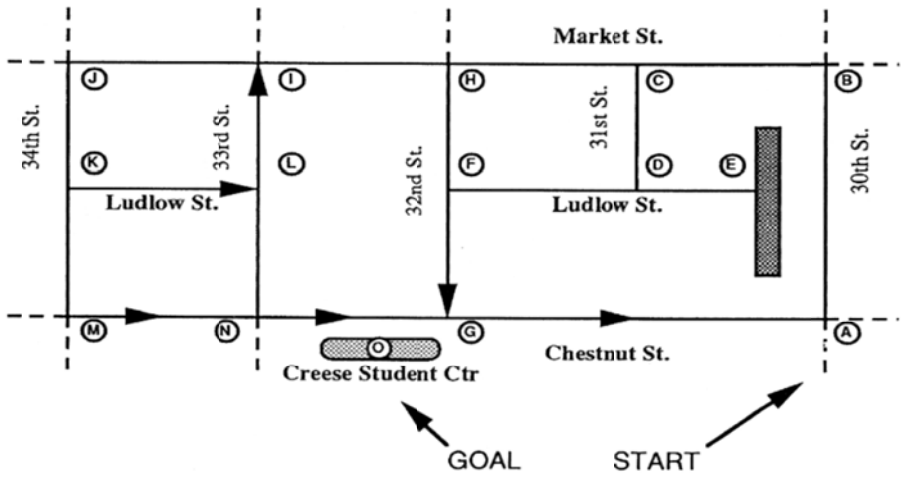
Graph search on an explicit graph

Depth First Search

Open	Closed

Breadth First Search

Open	Closed




```
function GRAPH-SEARCH( problem, fringe) returns a solution, or failure
1    closed ← an empty set
2    fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
3    loop do
4        if fringe is empty then return failure
5        node ← REMOVE-FRONT(fringe)
6        if GOAL-TEST(problem, STATE[node]) then return node
7        if STATE[node] is not in closed then
8            add STATE[node] to closed
9            fringe ← INSERTALL(EXPAND(node, problem), fringe)
10   end
```

(This is figure 3.19 from your book)

Fringe is also called "OPEN"

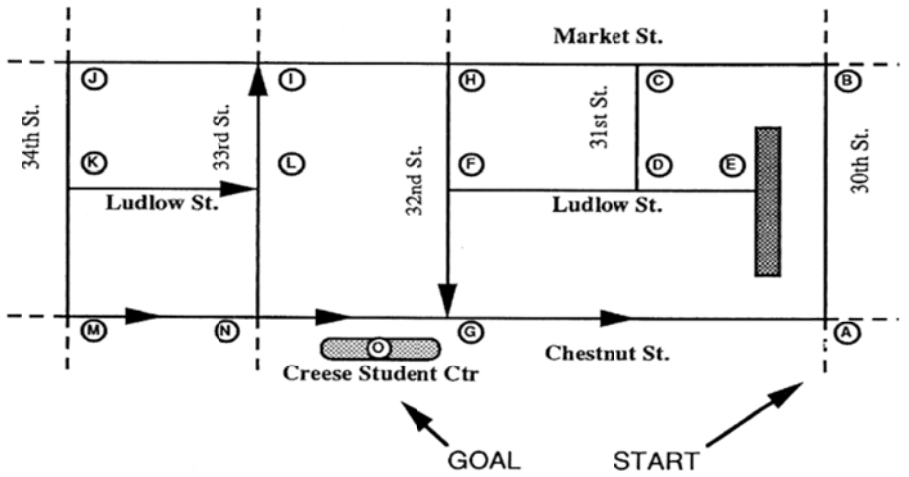
Graph search on an explicit graph

Depth First Search

	Open	Closed
1	**A**	
2	B	**A**
3	C	AB
4	DH	ABC
5	EFH	ABCD
6	FH	ABCDE
7	GH	ABCDEF
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		
18		

Breadth First Search

Open	Closed
A	
B	**A**
C	AB
DH	ABC
HEF	ABCD
EFI	ABCDH




```
function GRAPH-SEARCH( problem, fringe) returns a solution, or failure
1   closed ← an empty set
2   fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
3   loop do
4       if fringe is empty then return failure
5       node ← REMOVE-FRONT(fringe)
6       if GOAL-TEST(problem, STATE[node]) then return node
7       if STATE[node] is not in closed then
8           add STATE[node] to closed
9           fringe ← INSERTALL(EXPAND(node, problem), fringe)
10  end
```

(This is figure 3.19 from your book)

Fringe is also called "OPEN"