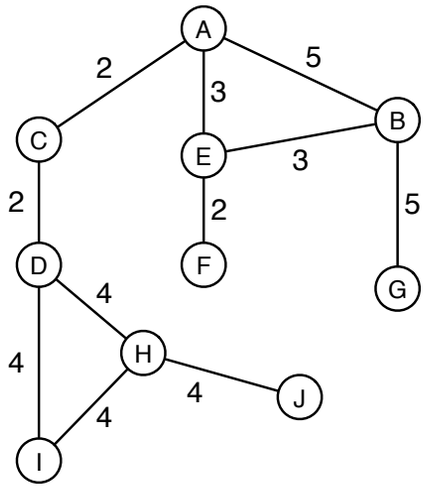
CS255: Artificial Intelligence
Seminar Sheet 1 — Uninformed Search

1. Consider the state space below where nodes represent valid states and edges represent valid transitions from one state to the next. The edges are labelled with their cost of traversal. The start state is labelled A and the goal state is labelled J.



   Using a lowest-cost-first *tree search* (i) show the resulting search tree, (ii) give the state of the queue as the search progresses, (iii) state the sequence in which nodes are selected for expansion and how many nodes are expanded, and (iv) state the route found and its cost.

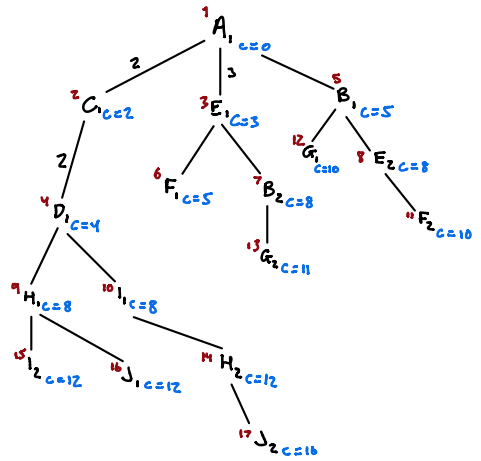
   When searching the state space, if two nodes are equally desirable then the selection should be done in alphabetic order. When expanding a node n, if n has a child node n' that is also an ancestor of n within the search tree then n' can be omitted from the expansion of n.

2. Using a lowest-cost-first *graph search* in the state space from Question 1, (i) give the state of the frontier at each step of the search, (ii) give the state of the closed set at each step, (iii) list any paths that are pruned, and (iv) state the route found and its cost. You should apply cycle checking and multiple-path pruning.

   When selecting paths from the frontier, if two paths are equally desirable then the selection should be done aphabetically according to the nodes at the end of the paths. If there is a tie-break, then you should select the path that has been in the frontier the longest.

3. In a tree search, what would be the result of using $f(n) = -depth(n)$ for inserting nodes into the queue?

1)



Route: A, C, D, H, J

Cost: 12

Nodes expanded: 17

Expanded:	Queue:
A_1	C_1^2, E_1^3, B_1^5
C_1	E_1^3, D_1^4, B_1^5 ← D, before B, as $c(D_1) = 4$, vs. $c(B_1) = 5$
E_1	D_1^4, B_1^5, F_1^5, B_2^6
D_1	B_1^5, F_1^5, B_2^6, H_1^8, I_1^8
B_1	F_1^5, B_2^6, E_2^8, H_1^8, I_1^8, G_1^{10}
F_1	B_2^6, E_2^8, H_1^8, I_1^8, G_1^{10}
B_2	E_2^8, H_1^8, I_1^8, G_1^{10}, G_2^{11}
E_2	H_1^8, I_1^8, F_2^{10}, G_1^{10}, G_2^{11}
H_1	I_1^8, F_2^{10}, G_1^{10}, G_2^{11}, I_2^{12}, J_1^{12}
I_1	F_2^{10}, G_1^{10}, G_2^{11}, H_2^{12}, I_2^{12}, J_1^{12}
F_2	G_1^{10}, G_2^{11}, H_2^{12}, I_2^{12}, J_1^{12}
G_1	G_2^{11}, H_2^{12}, I_2^{12}, J_1^{12}
G_2	H_2^{12}, I_2^{12}, J_1^{12}
H_2	I_2^{12}, J_1^{12}, J_2^{16}
I_2	J_1^{12}, J_2^{16}
J_1	J_2^{16}
J_2	-

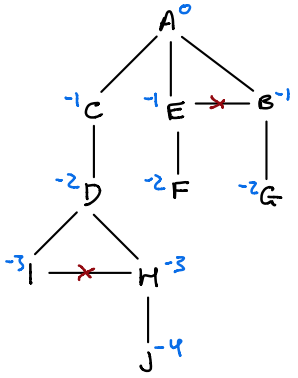
2)

Frontier:	Closed list:	Status:
$\langle A \rangle_0$	-	
$\langle A,C \rangle_2$, $\langle A,E \rangle_3$, $\langle A,B \rangle_5$	A^0	$\langle A \rangle$: Explored
$\langle A,E \rangle_3$, $\langle A,C,D \rangle_4$, $\langle A,B \rangle_5$	A^0, C^2	$\langle A,C \rangle$: Explored
$\langle A,C,D \rangle_4$, $\langle A,B \rangle_5$, $\langle A,E,F \rangle_5$, $\langle A,E,B \rangle_6$	A^0, C^2, E^3	$\langle A,E \rangle$: Explored
$\langle A,B \rangle_5$, $\langle A,E,F \rangle_5$, $\langle A,E,B \rangle_6$, $\langle A,C,D,H \rangle_8$, $\langle A,C,D,I \rangle_8$	A^0, C^2, E^3, D^4	$\langle A,C,D \rangle$: Explored
$\langle A,E,F \rangle_5$, $\langle A,E,B \rangle_6$, $\langle A,B,E \rangle_8$, $\langle A,C,D,H \rangle_8$, $\langle A,C,D,I \rangle_8$, $\langle A,B,G \rangle_{10}$	A^0, C^2, E^3, D^4, B^5	$\langle A,B \rangle$: Explored
$\langle A,E,B \rangle_6$, $\langle A,B,E \rangle_8$, $\langle A,C,D,H \rangle_8$, $\langle A,C,D,I \rangle_8$, $\langle A,B,G \rangle_{10}$	$A^0, C^2, E^3, D^4, B^5, F^5$	$\langle A,E,F \rangle$: Explored
$\langle A,B,E \rangle_8$, $\langle A,C,D,H \rangle_8$, $\langle A,C,D,I \rangle_8$, $\langle A,B,G \rangle_{10}$	$A^0, C^2, E^3, D^4, \underline{B^5}, F^5$	$\langle A,E,B \rangle$: Pruned
$\langle A,C,D,H \rangle_8$, $\langle A,C,D,I \rangle_8$, $\langle A,B,G \rangle_{10}$	$A^0, C^2, \underline{E^3}, D^4, B^5, F^5$	$\langle A,B,E \rangle$: Pruned
$\langle A,C,D,I \rangle_8$, $\langle A,B,G \rangle_{10}$, $\langle A,C,D,H,I \rangle_{12}$, $\langle A,C,D,H,J \rangle_{12}$	$A^0, C^2, E^3, D^4, B^5, F^5, H^8$	$\langle A,C,D,H \rangle$: Explored
$\langle A,B,G \rangle_{10}$, $\langle A,C,D,I,H \rangle_{12}$, $\langle A,C,D,H,I \rangle_{12}$, $\langle A,C,D,H,J \rangle_{12}$	$A^0, C^2, E^3, D^4, B^5, F^5, H^8, I^8$	$\langle A,C,D,I \rangle$: Explored
$\langle A,C,D,I,H \rangle_{12}$, $\langle A,C,D,H,I \rangle_{12}$, $\langle A,C,D,H,J \rangle_{12}$	$A^0, C^2, E^3, D^4, B^5, F^5, H^8, I^8, G^{10}$	$\langle A,B,G \rangle$: Explored
$\langle A,C,D,H,I \rangle_{12}$, $\langle A,C,D,H,J \rangle_{12}$	$A^0, C^2, E^3, D^4, B^5, F^5, \underline{H^8}, I^8, G^{10}$	$\langle A,C,D,I,H \rangle$: Pruned
$\langle A,C,D,H,J \rangle_{12}$	$A^0, C^2, E^3, D^4, B^5, F^5, H^8, \underline{I^8}, G^{10}$	$\langle A,C,D,H,I \rangle$: Pruned
	$A^0, C^2, E^3, D^4, B^5, F^5, H^8, I^8, G^{10}, J^{12}$	$\langle A,C,D,H,J \rangle$: Explored

ACDHJ, Cost: 12

Paths expanded: 14

3)



$f(n) = -\text{depth}(n)$

↳ Depth first search

function TREE-SEARCH(*problem*) **returns** a solution, or failure
  initialize the frontier using the initial state of *problem*
  loop do
    if the frontier is empty **then return** failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state **then return** the corresponding solution
    expand the chosen node, adding the resulting nodes to the frontier

---

function GRAPH-SEARCH(*problem*) **returns** a solution, or failure
  initialize the frontier using the initial state of *problem*
  initialize the explored set to be empty
  loop do
    if the frontier is empty **then return** failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state **then return** the corresponding solution
    add the node to the explored set
    expand the chosen node, adding the resulting nodes to the frontier
      only if not in the frontier or explored set

Figure 3.7 An informal description of the general tree-search and graph-search algorithms. The parts of GRAPH-SEARCH marked in bold italic are the additions needed to handle repeated states.