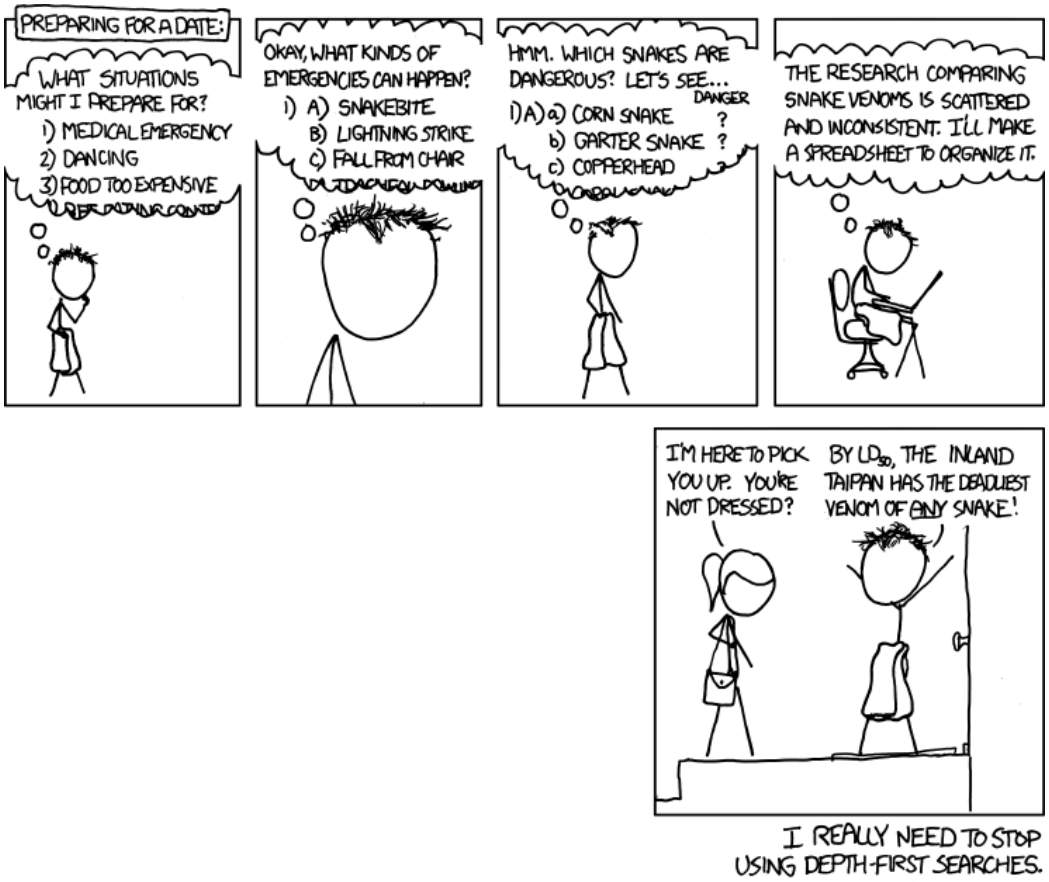
CITS3001 Mid-semester Test 2015

Fifty minutes

Answer all four questions

Total marks 60



When showing the operation of an algorithm, include enough detail to make it clear that you understand how *the algorithm* solves the problem.

Question 1: Dynamic Prog. for the 0-1 Knapsack Problem (15 marks)

Briefly describe the principles, operation, and performance issues of dynamic programming for solving the 0-1 Knapsack Problem.

An instance of 0-1 Knapsack with n items has the form

$$(\{w_1, \ldots, w_n\}, \{v_1, \ldots, v_n\}, W)$$

where w_i and v_i are the weight and value of item i, respectively. W is the knapsack capacity.

Illustrate the process of using dynamic programming on the following problem instance.

$$(\{1, 2, 3\}, \{2, 3, 4\}, 5)$$

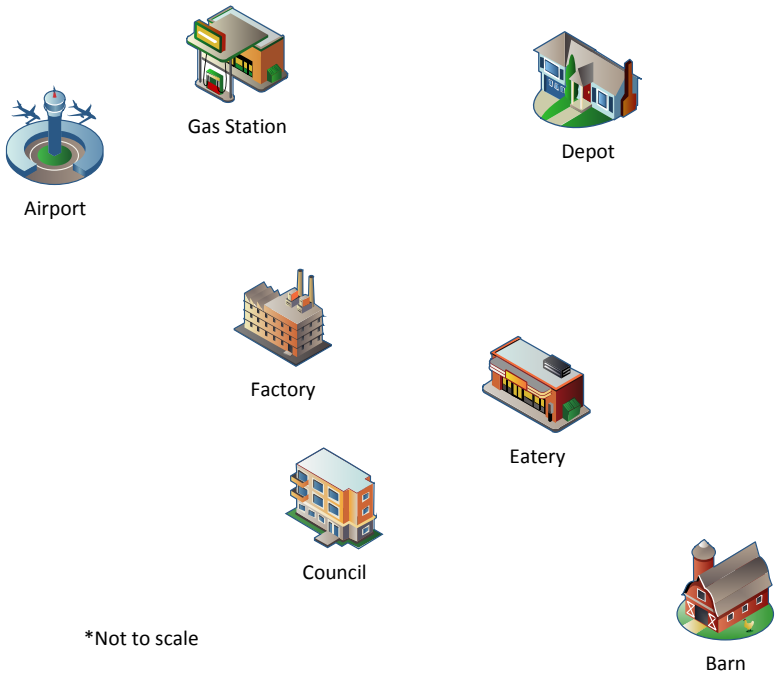
Make sure that you show

- the recursive rule; and
- how the rule is applied to fill in a value table. Each entry of the value table, $V(m,w)$, represents the total value of optimal solution where we choose from the first m items with capacity w.

Question 2: the Travelling Salesman Problem (15 marks)

Briefly describe the principles, operation, and performance issues of any *one* commonly-used approximation algorithm for the Travelling Salesman Problem.

Illustrate your answer using the distance in Fig. 1 for all places that a delivery van needs to visit in a city.



	A	B	C	D	E	F	G
A	--	35	16	28	15	14	5
B	35	--	17	26	9	15	31
C	16	17	--	25	10	4	21
D	28	26	25	--	15	17	14
E	15	9	10	15	--	7	21
F	14	15	4	17	7	--	13
G	5	31	21	15	21	13	--

Fig. 1: A set of locations in a city, and distances between them

Make sure that you show all relevant operational details of your chosen algorithm.

Question 3: Iterative Deepening Depth-First Search (15 marks)

Briefly describe the principles, operation, and performance issues of iterative deepening depth-first search. Discuss the problems associated with Breath-first and Depth-first that Iterative Deepening addresses.

Illustrate your answer using the road map in Fig. 2. Find the route from **A**irport to **B**arn that goes through the fewest intermediate cities. Do we need to use the distances listed for each road?

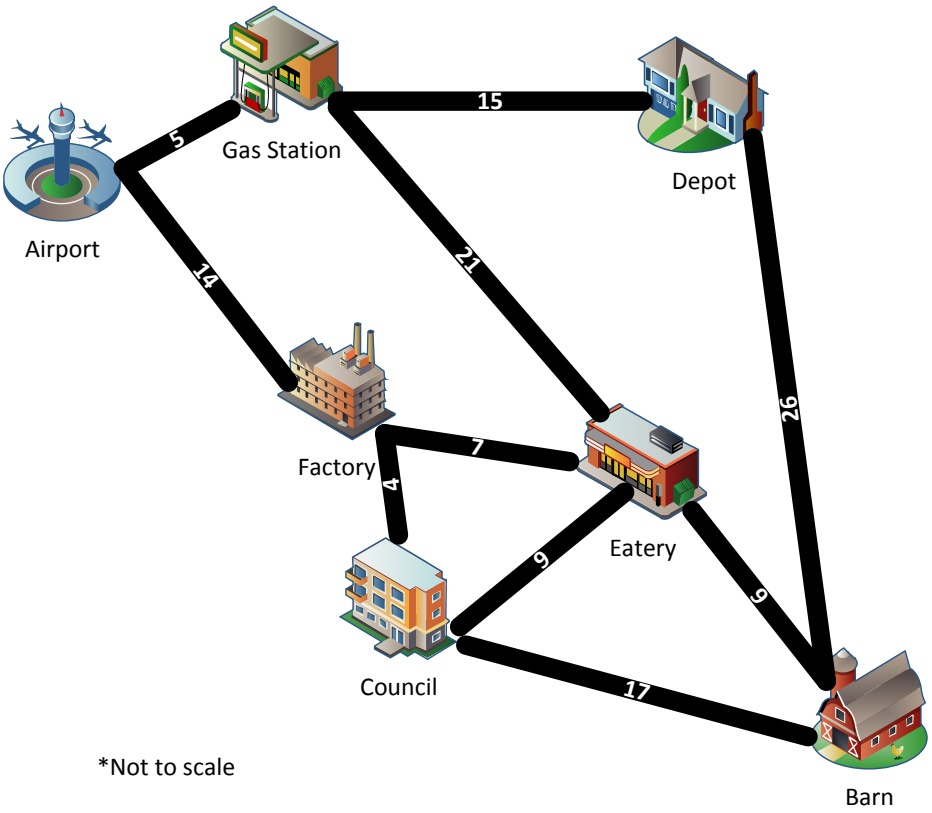


Fig. 2: A road map for locations in a city for both Question 3 and 4

Make sure that you show which nodes are expanded, in what order, and why.

Note that within an iteration, when you have expanded a node X at depth k, there is no need to expand X again at any depth $j \geq k$.

Question 4: *A** **(15 marks)**

Briefly describe the principles, operation, and performance issues of *A**.

Illustrate your answer using the road map below and the table of straight-line distances in Fig. 3. Find the route from **Airport** to **Barn** that requires the least total distance.

Airport	31	**E**atery	7
Council	16	**F**actory	19
Depot	21	**G**as Station	27
		Barn	0

Fig. 3: Straight-line distance from each location to Barn for Question 4.

Make sure that you show which nodes are expanded, in what order, and why.

Note that when you have expanded a node X at $g(X) = k$, there is no need to expand X again at any $g(X) \geq k$. We use $g(X)$ to record the actual cost of getting to node X.

Question 1

- (8 marks) DP: express the problem as one or more recurrence relations (i.e. sub-problems) and organise to minimise or eliminate the repeated work.
    - The recursive rule:
        - Let $V(m, w)$ be the value of the optimal solution where we choose from the first m items with capacity w
        - Either the m^{th} item is packed or it isn't, so

$$V(m, w) = max(v_m + V(m - 1, w - w_m), V(m - 1, w))$$

        - With the trivial base cases $V(0, w) = V(m, 0) = 0$,
          and incorporating checks for exceeding w.
- (7 marks) Table:

m \ *w*	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	2	2	2	2	2
2	0	2	3	5	5	5
3	0	2	3	5	6	7

Question 2

Various solutions are possible.

Minimum spanning tree (MST): guarantees 2-optimal. The MST for this case is AG(F(CE(B))D) with the minimum length 52

A tour can be built traversing the tree depth first AGFCEBDA

Question 3

(8 marks) BFS: Optimal and Complete, but terrible Space Complexity

DFS: Good space complexity, but neither optimal nor complete. There is also the possibility of getting trapped by an infinite path.

IDS: Address the problem of BFS and DFS by iteratively going through depth from 0 to d with an increment of 1, d is the depth of the optimal solution.

(7 Marks)

Limit = 0, Airport is not a goal

Limit = 1, A*(FG)

Limit = 2, A*(

F*(CE)

G*(DE))

Limit = 3, A*(

F*(C*(Barn)E*(Barn))

G*(D*(Barn)E*(Barn)))

So the solution with the fewest cities is AFCB or AFEB or AGDB or AGEB

Question 4

A* process: good description but no working out - 8 marks; with good working out – full marks.

Note the "straight" line distance given does not meet the requirement of h(x)<h*(x) in the case of node F. h(F) is given as 19, which is greater than the F->E->B (which is 7+9 = 16). In other words, the h(x) given isn't admissible, so it may not lead to optimal answer.

c	*g(c)*	*h(c)*	*f(c)*
Airport	0	31	31

Expand A

c	*g(c)*	*h(c)*	*f(c)*
G	5	27	32
F	14	19	33

Expand G, copy F, sort by *f*

c	*g(c)*	*h(c)*	*f(c)*
E	26	7	33
F	14	19	33
D	20	21	41

Expand E, copy D and F, sort by *f*

c	*g(c)*	*h(c)*	*f(c)*
F	14	19	33
B	35	0	35
D	20	21	41
C	35	17	52

B is the goal state. AGEB

Alternatively,

c	*g(c)*	*h(c)*	*f(c)*
E	26	7	33
F	14	19	33
D	20	21	41

Expand F, copy D, sort by *f*

c	*g(c)*	*h(c)*	*f(c)*
E	21	7	28
C	18	16	34
D	20	21	41

Expand E, copy C and D, sort by f

c	*g(c)*	*h(c)*	*f(c)*
B	30	0	30
C	18	16	34
D	20	21	41

B is the goal state. AFEB