The University of British Columbia
Department of Computer Science
Midterm Examination - Fall 2003

Computer Science 322
(Introduction to Artificial Intelligence)

Total marks: 48
Time: 60 minutes

Name:

Student Number:

Question 1 [8 marks]

Consider the following knowledge base, KB:

1. $a(nil, A, A)$.
2. $a(cons(H,T), L, cons(H,R)) \leftarrow a(T, L, R)$.

Give a successful top-down derivation with substitutions for the query

$?a(cons(d, cons(o, nil)), cons(g, nil), S)$

State which clause is chosen from the knowledge base and the substition set used at each step.

Answer

$yes(S) \leftarrow a(cons(d, cons(o, nil)), cons(g, nil), S)$.

choose clause 2, substitution $\{H/d, T/cons(o, nil), L/cons(g, nil), S/cons(d, R_1), R/R_1\}$

$yes(cons(d, R_1)) \leftarrow a(cons(o, nil), cons(g, nil), R_1)$.

choose clause 2, substitution $\{H/o, T/nil, L/cons(g, nil), R_1/cons(o, R_2), R/R_2\}$

$yes(cons(d, cons(o, R_2))) \leftarrow a(nil, cons(g, nil), R_2)$.

choose clause 1, substitution $\{A/cons(g, nil), R_2/cons(g, nil)\}$

$yes(cons(d, cons(o, cons(g, nil)))) \leftarrow$

The answer is $S = cons(d, cons(o, cons(g)))$.

You got 1/2 a mark for each appropriate choice of clause, 1 and 1/2 marks for each correctly formed subsitition set, and 1/2 a mark for each correctly formed clause (including the initial clause and the final clause).

Question 2 [10 marks]

Consider the following knowledge base, KB:

$a(X, Y) \leftarrow b(X) \wedge c(Y).$
$a(X, Y) \leftarrow c(Y) \wedge e(X).$
$b(X) \leftarrow d(X).$
$b(X) \leftarrow e(X).$
$c(m).$
$c(n).$
$d(p).$

(a) [3 marks] What does the equation $KB \models g$ mean? What are the conditions required for this to be true?
(b) [3 marks] Give 3 atoms that are *not* logical consequences of KB.
(c) [4 marks] Give the final consequence set produced by a bottom-up proof procedure from KB.

Answer

(a) $KB \models g$ means that g is a logical consequence of KB. In order for this to be the case, for every interpretation I that is a model of KB, g must also be true in I. An interpretation I is a model of KB if every clause in KB is true in I.
1 mark for saying what the equation stands for, 2 marks for the conditions.

(b) Any three from $b(m), b(n), c(p), d(m), d(n), e(m), e(n), e(p)$ or any combinations using $a(X, Y)$ apart from $a(p, m)$ or $a(p, n)$. Or something not mentioned in KB at all...
1 mark for each correct answer.

(c) $C = c(m), c(n), d(p), b(p), a(p, m), a(p, n)$
2 marks for $c(m), c(n)$ and $d(p)$. 1 mark for $b(p)$. 1 mark for $a(p, m)$ and $a(p, n)$.

Question 3 [16 marks]

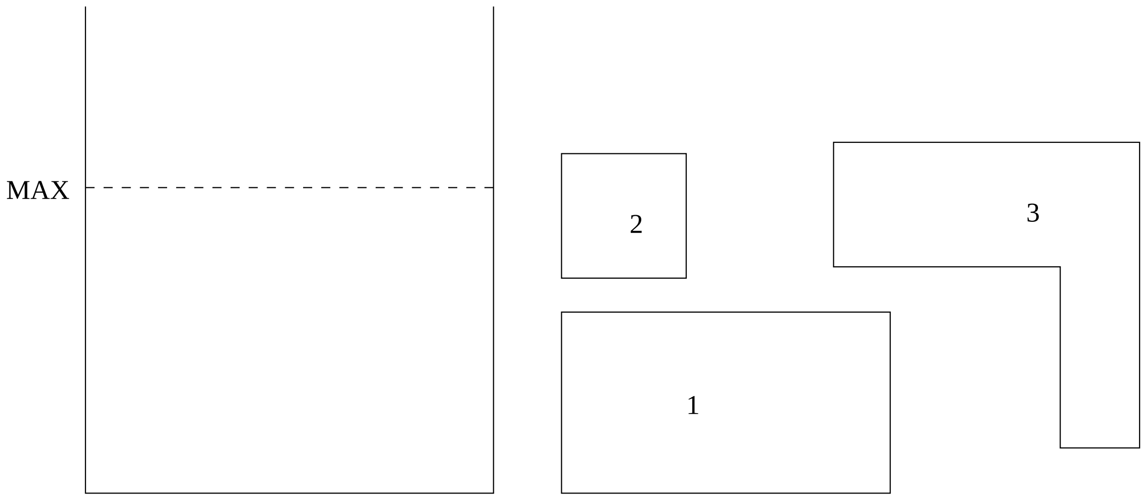


Figure 1: Bin Packing domain

This is a real-world problem, actions which are physically impossible are to be ignored.

Consider the large bin and collection of numbered blocks shown in Figure 1. If the blocks are placed into the bin in one of two correct orders then they will all fit with no part of any block above the MAX line indicated. The task is to search for this correct order. The two possible actions (both of which have a cost of 1) are:

1. add a block to the bin,
2. remove a block from the bin

The blocks *cannot* be rotated in any way. Blocks can only be removed from the bin in the reverse of the order in which they were added (e.g. the last one added must be removed first).

For each of the following search algorithms answer these questions with regards to the search problem described:

(a) [2 marks] Assuming a solution exists within the search space, is the algorithm guaranteed to find it? Why/why not?
(b) [1 mark] What effect would multiple-path pruning have on the performance of the algorithm?
(c) [5 marks] Draw the search tree up to and including the current frontier after the first three nodes have been examined/expanded by the algorithm. Include the current state of the bin within each node. Ensure that the state representation indicates the order in which the blocks have been added.

(a) Depth First

1. Depth first is not guaranteed to return the answer because it could loop forever on this problem by continuously adding and removing the same block.

1 mark for saying that it isn't guaranteed to find the answer, 1 mark for why.

2. Multiple-path pruning would remove the cycles in the search graph, thus stopping the algorithm from looping forever and enabling it to (eventually) return an answer. It would also increase the amount of space required to run the algorithm from linear to exponential.

I was primarily looking for the first reason, but the second reason is also true and so would get the mark. (no bonus marks for both I'm afraid).

3.

Your bin contents may differ depending on which node you picked to expand.

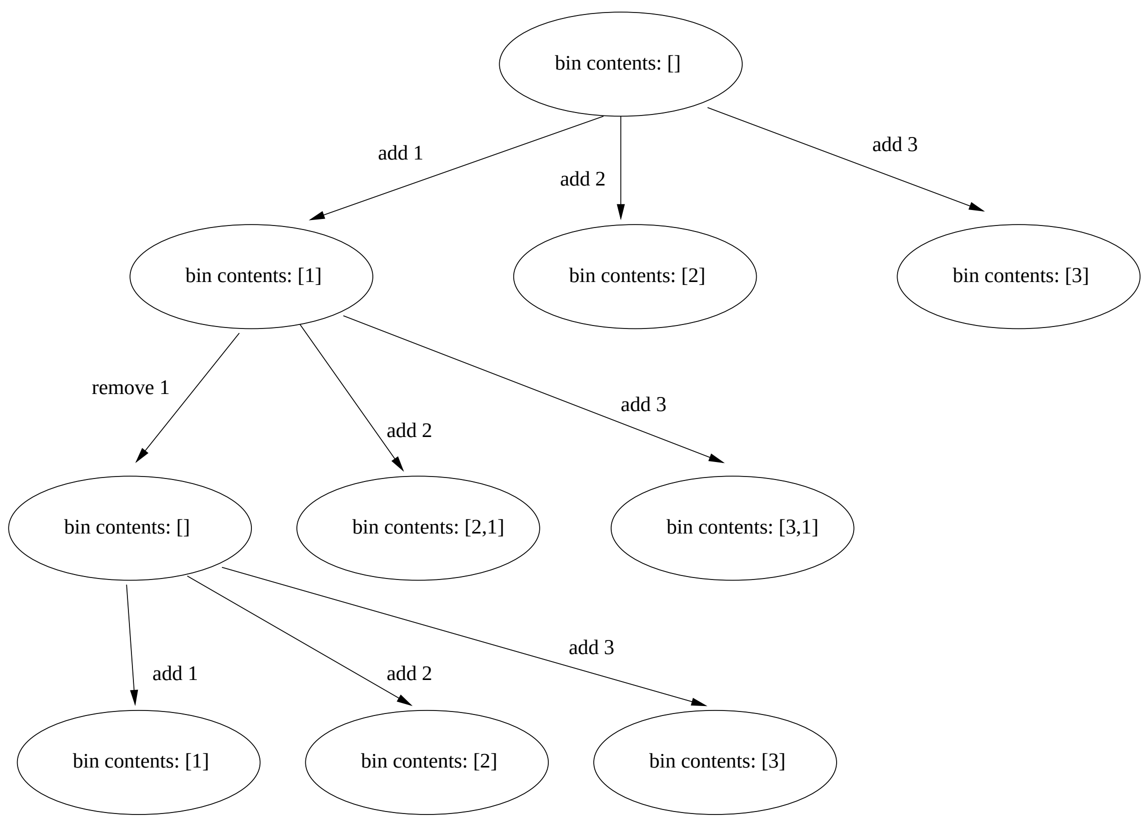


Figure 2: Top-down search tree

(b) Lowest Cost First

1. This algorithm *will* return an answer. Since it selects the path which currently has the lowest cost and expands it, and the cost of a path within a loop would keep increaseing, it would break out of the loop and look elsewhere.

1 mark for saying that it will return a solution. 1 mark for saying why.

2. Multiple path pruning may speed up the algorithm by reducing the size of the frontier, but little else.

3.

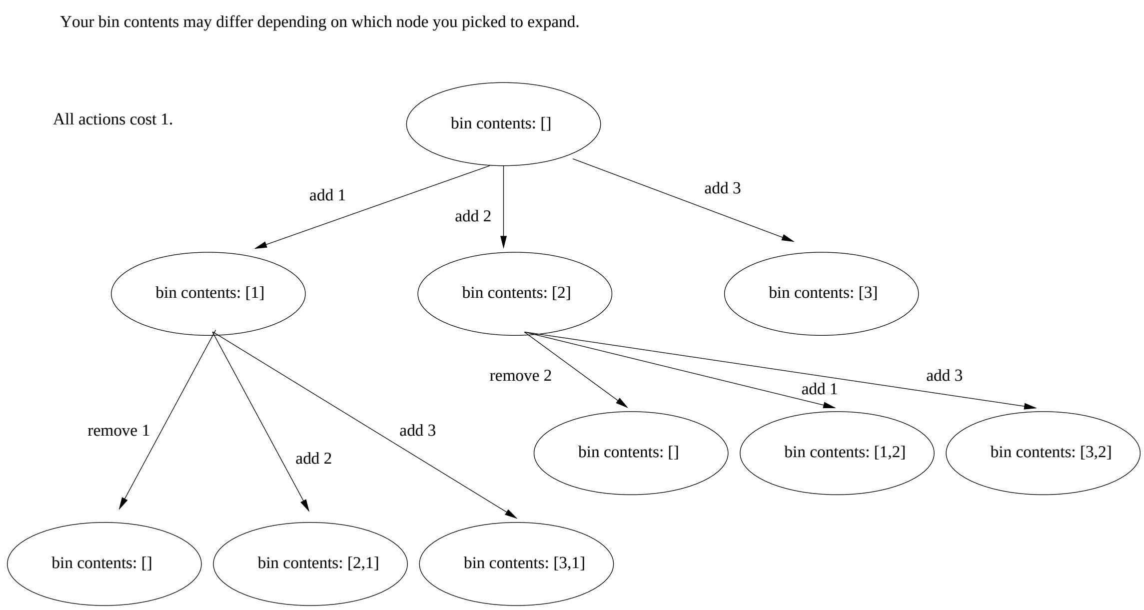


Figure 3: Lowest-cost search tree

Question 4 [8 marks]

Five variables, each with a set of possible values, and a set of constraints have been used to describe a particular creature. Use the **backtracking algorithm** to determine which of the possible creatures listed has been described.

Possible creatures: elephant, dragonfly, rhinoceros, tiger, eagle.

Variables:

Horns or Tusks(H)	values: $\{0, 1, 2\}$
Eyes(E)	values: $\{0, 2, 4, 1000\}$
Wings(W)	values: $\{0, 2, 4\}$
Legs(L)	values: $\{0, 2, 4, 6\}$
Tails(T)	values: $\{0, 1\}$

Constraints:

$E > W$	$W > 2 \times T$	$T > H$
$H = 0$	$H \neq W$	$L < E$
$L > W$	$L > T$	

Use the variable ordering: H T W L E

Answer (continue onto the back page if required)

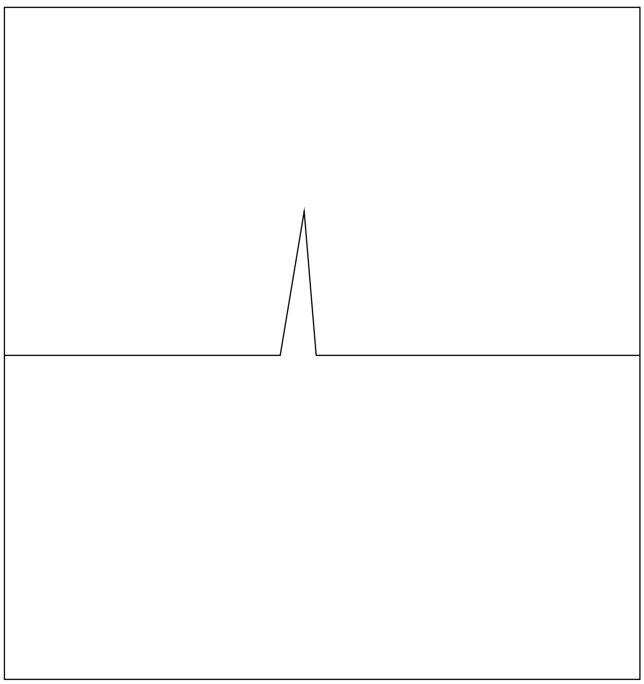
H in 0
- T in 0 fail
- T in 1
  - W in 0 fail
  - W in 2 fail
  - W in 4
    - L in 0 fail
    - L in 2 fail
    - L in 4 fail
    - L in 6
      - E in 0 fail
      - E in 2 fail
      - E in 4 fail
      - E in 1000 SOLUTION

$H = 0$, $T = 1$, $W = 4$, $L = 6$, $E = 1000$. The answer is a **dragonfly**.

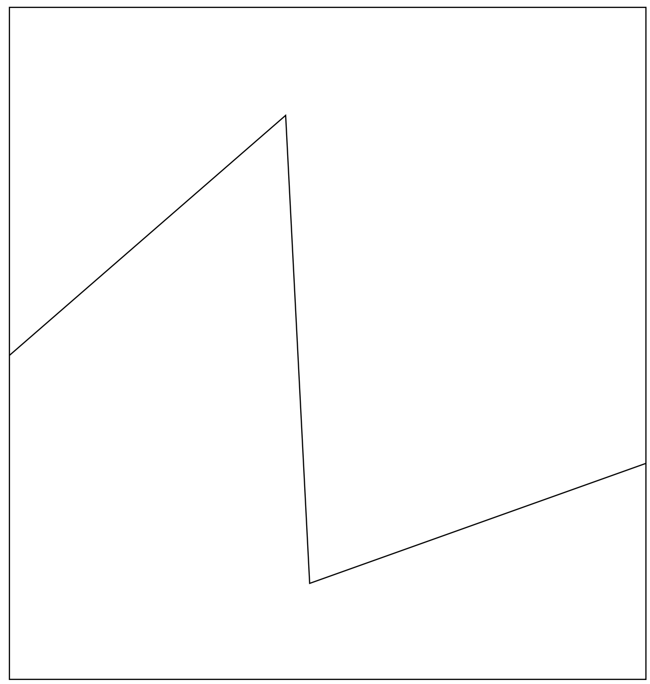
Question 5 [6 marks]

Two 1-dimensional heuristic landscapes are illustrated in Figure 4. For each landscape, say whether **random restart hill-climbing** or **beam search** would reach an acceptable solution more quickly (or if they would give roughly the same performance) and why. Assume that only the highest peak in each landscape has been classified as an acceptable solution.

Include the important details of the two methods in your answer. You may present your reasoning about both landscapes within a single answer if you wish.



Landscape a



Landscape b

Figure 4: Two 1-dimensional landsapes of heuristic values

Answer (continue onto the back page if required)

Landscape (a) Simple Hill Climbing (SHC) will move to the neighbour of its current position that has the best increase in heuristic value. Once it has reached a local maxima then it will remain there. This particular landscape has only one maxima and the rest of the landscape is completely flat (i.e. a plateau). If we are lucky then the start point will be on the side of the peak, but there is a high probability that the algorithm will start elsewhere. However, we are using **random restart hill-climbing** (RRHC) on this occasion. This means that each time the hill climbing method cannot progress then it will randomly restart somewhere else. If we let it keep restarting then eventually it will find the peak.

Beam search will start with k points, generate neighbours for each of those points and then choose the k best neighbours to be the new set of points. Again, if we are lucky (or we have a very large k), then we will have one of those starting points on the peak. This would be all that beam search would need to find the solution. However, if none of the points starts on the peak, and none of the neighbours of the starting points is located on the peak then beam search also cannot progress and find the solution.

Both assessments have been made with the assumption that the implementation of the algorithm will halt if it cannot find a 'better' answer than the one it already has. If we were to assume that the algorithms would pick one of the neighbours that have the *same* heuristic level (either arbitrarily or according to some directive given by the user) then both would find the peak. RRHC would tend to be more efficient as we are only maintaining one point rather than several, however it is possible for it to start, for example, to the left of the peak and just keep going left when no improvement is found. Therefore there would be no definate answer as to which would be more efficient.

Landscape (b) RRHC has approximately a 50 percent change of starting on the side of the peak we are looking for. Simple hill climbing would get stuck on local maxima, but because we can then start again somewhere else, this is not a problem. If we restart enough times we will eventually find the peak. Beam search, starting from many places, is more likely to start somewhere on the peak. However, if it *doesn't* then it will not find the solution at all because it has no means to rescue itself from local maxima. It does not make random choices of neighbours or do a random restart.

Neither is clearly the best choice here either.

The whole question had 6 marks to be given for a good argument that demonstrated knowledge of the two methods being compared (although you did not need to do it in the level of detail that has been given above, and the assumption about how to handle plateaus is also up to you). The marks were not divided into 3 per part as you may have demonstrated more knowledge in the first answer.