Massachusetts Institute of Technology

6.034 Artificial Intelligence

Handout 6034 Item # **18**

Problem 1 Search (Winston)

Part A

Were it not for the middle nodes in the lower part of the network of streets leading to I, you would have a binary tree of depth nine, with $2^9 = 512$ leaf nodes, each of which would generate a path terminating at I.

Many people failed to note that the middle nodes do not branch and viewed the network as a combination of a network with branching factor 2 with a network of branching factor 3. There would be 2^5 leaf nodes at the bottom of the binary tree, and for each of those, there would be 3^4 leaf nodes, each of which would generate a path terminating at I. That is, $2^5 \times 3^4 = 2592$. We gave full credit for this answer.

Those who did note that the middle nodes do not branch proceeded to the correct answer along several routes. One approach was to note that the network is equivilent to a nine-level binary tree plus the contributions of the middle node. Consider for example, the highest of the middle nodes. It receives inputs from half of the leaf nodes of a binary tree six-levels deep, thus contributing $\frac{1}{2} \times 2^6 = 2^5 = 32$ paths. Thus, the total number of leaf nodes is $2^9 + 2^5 + 2^6 + 2^7 + 2^8 = 992$. We gave a point of extra credit for this answer.

Part B

Because all the nodes in the network leading to I are closer to the goal than any of the nodes leading to J, all the paths in that network are explored, just as if ordinary depth-first search were used. We gave full credit to 2592, 992, and any answer that was the same as the answer offered in part A.

Part C

Because dynamic programming is used, only one path leading to any particular node is extended. Thus, only one path leading to each of the three nodes just above I are extended. Accordingly, there are just 3 paths terminating at place I.

Alternatively, an extremely careful reading of the definition of dynamic programming provided in the problem could lead to a construction that produces an answer of 5, because the shortest paths to the middle and right nodes just above I are not the first paths produced to those nodes.

Partial credit was awared for an answer of one, if consistent with the answer to part E, for those who misread or overlooked the meaning of the word *generate*.

Part D

Beam search only extends the w best scoring paths at any level. Hence there are w paths at any level. Hence in this problem, 2 paths terminate at place I. Without backtracking, no path from S to G would be found. The point distribution was 3 for the first part and 2 for the second.

Part E

Because the path leading to the penultimate node on the correct path is so long, all paths to the nodes just above I, not otherwise terminated by dynamic programming, are considered. Because of dynamic programming, there are just three such paths, each of which generates a path to I. Thus, the answer is the same as the answer in part C, and points were awarded for such consistency.

Similarly, all paths to the nodes just before J, not otherwise terminated by dynamic programming, are considered. Because of dynamic programming, there are just two such paths, each of which generates a path to J.

The point distribution was 3 for the first part and 2 for the second.

Partial credit was awared for those who answered one path in both cases, if consistent with the answer to part C, for those who misread or overlooked the meaning of the word *generate*.

Problem 2 Games (Davis)

Part A

See the figure for the official answer.

Part B

See the figure for the official answer.

Note that branch and bound searches from the least cost path. Initially that's the branch labelled "20" at the right of the tree in the figure. We need consider only two offspring of that branch (because there's no relevant node from Part A in the rightmost sub-tree). Hence we visit A19, discovering a path of cost 28, and node A23, discovering a path of cost 30.

But the middle branch of the tree indicates a path cost of only 23, so we have to push search forward from there, in case there is a path of cost less than 28 (the best path to a goal so far). In doing this we visit A11 (cost = 40) and A14 (cost = 31).

At this point we can quit, because the only remaining open path lies at the end of the branch labelled "30" at the left of the tree, and this already costs more than the path to A19.

Part C

See the figure for the official answer.

Many people made the mistake of subtracting from the static score only the cost of the last move in the path. You lost only a few points for this.

Part D

Options 1 and 2 simply ignore one or the other of the factors and hence don't work.

Options 3 and 4 will work. Option 4 is best, as it makes the Static Score dominate the scoring function, and allows Path Cost to break ties between equal Static Scores. Option 3 also works given the values in the current problem.

Option 5 puts the weights in the wrong direction, and 6 is ruled out by 3 and 4.

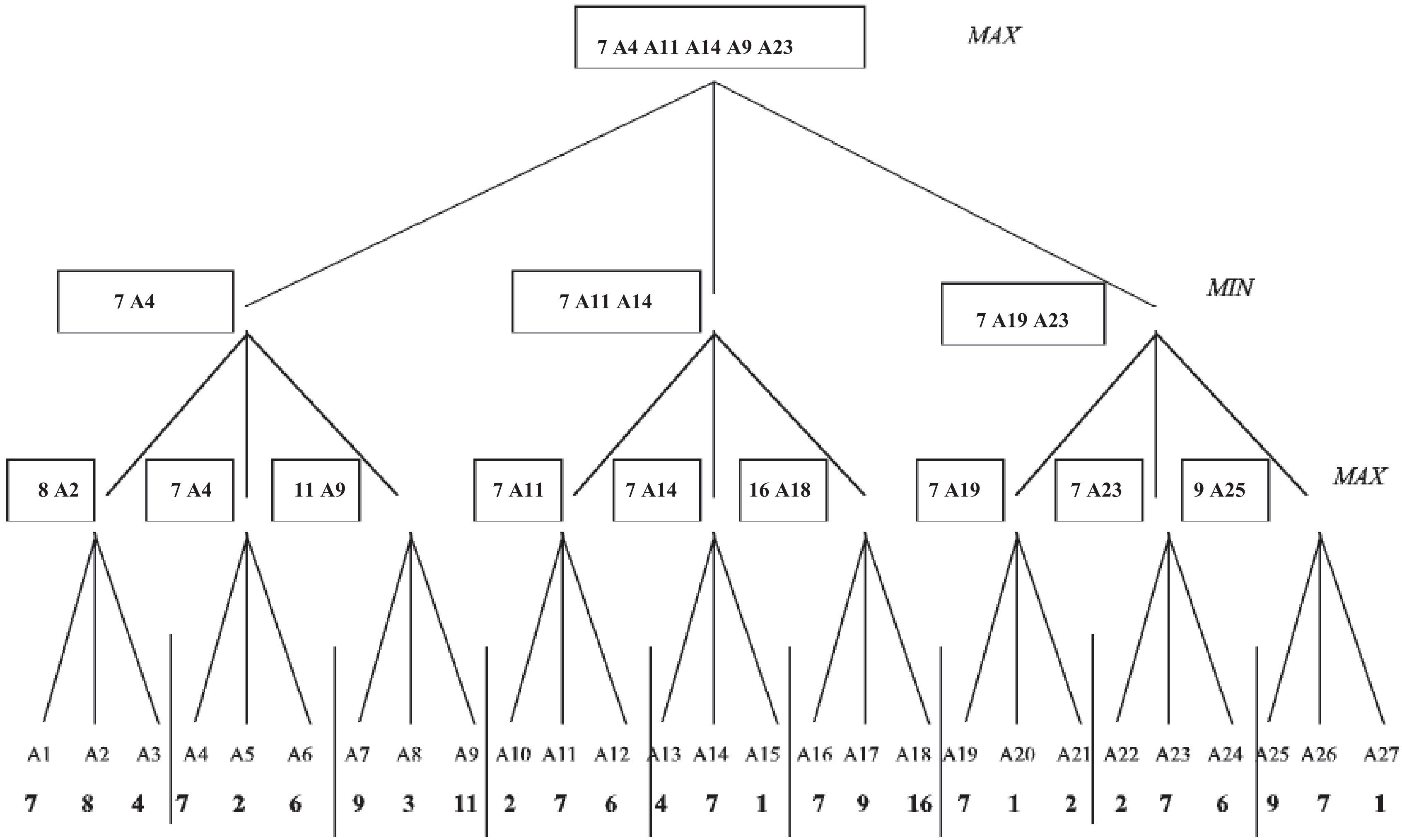

Figure 2.1: Numbers in **bold** are static evaluation scores for the node.

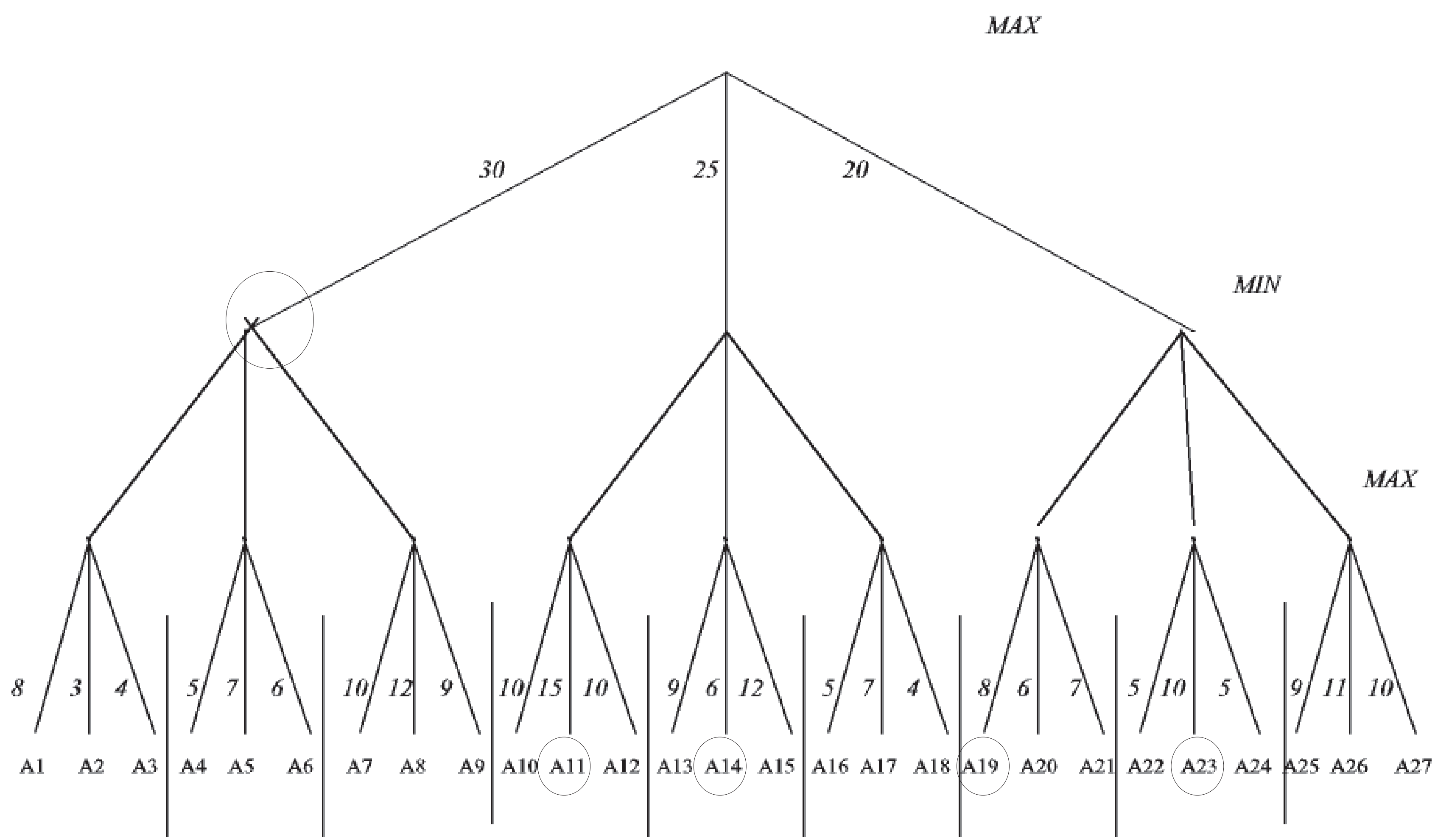

Figure 2.2: *Italic* numbers are the cost of the making the move associated with that branch.

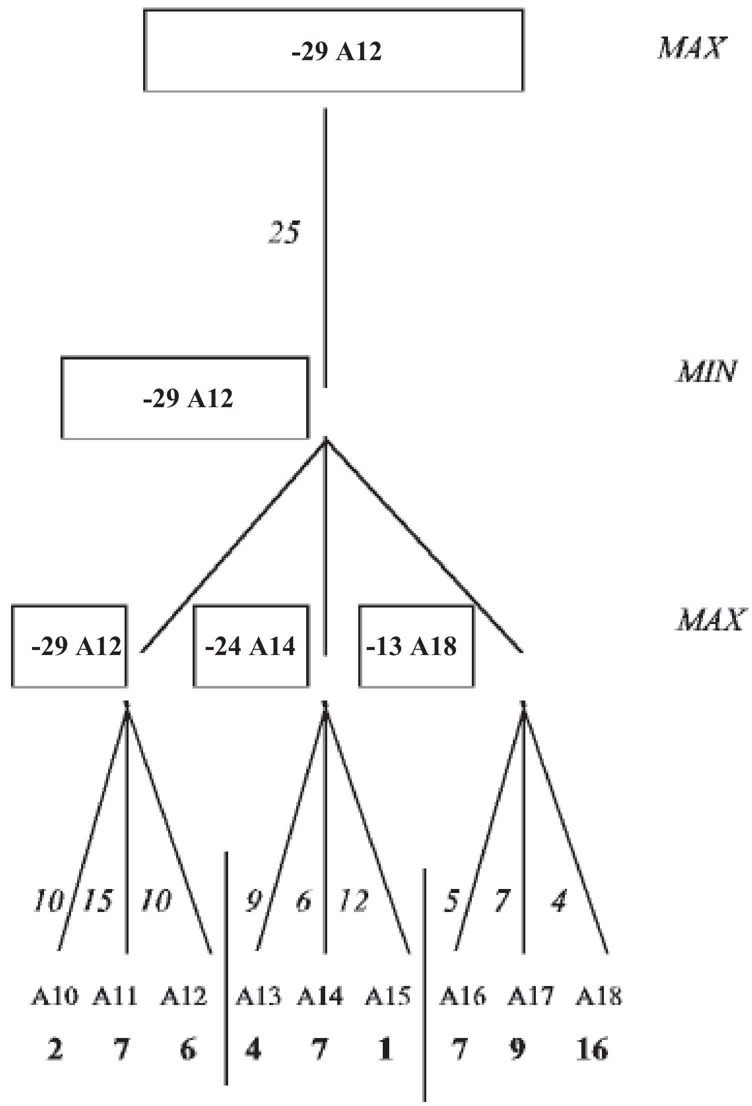

Figure 2.3: The middle segment of the same tree, showing both **static scores** and *move costs*.

Problem 3 BC Rules (Lozano-Perez)

Part A

This is simply a pattern for the input and answer. Note that the answer is unknown and therefore specified as a variable.

```
(backchain '(deriv (expt x 2) ?ans))
```

Part B

```
                 (deriv (expt x 2) ?ans)
                           |
    ---------------------------------
R1  | ?x1<-(expt x 2)           R7  | ?x7<-x
    | ?ans<-0                       | ?n7<-2
    |                               | ?ans<-(* 2 (expt x (- 2 1)) ?dx7)
    |                               |
(constant (expt x 2))          --------------
  QUESTION                     |             |
                          (deriv x ?dx7)  (constant 2)
                               |            QUESTION
                            R1 | ?x1<-x
                               | ?dx7<-0
                               |
                          (constant x)
                            QUESTION
```

Part C

Note that this fails to find the answer because we don't know the derivative of `x`. So simply assert

```
(deriv x 1)
```

Part D

One key part of this question is to notice that we are asking if we can avoid questions for derivatives that are actually covered by these rules. That is, we don't care about sines, cosines or functions of y or whatever. In that case, all we have to do is to remove R1 and the `(constant ?n7)` premise from R7 and add a default rule (at the end of the rule list) that says that treats everything as a constant.

```
R8 IF
   THEN (deriv ?x8 0)
```

Problem 4 Puzzling CSP (Lozano-Perez)

Part A

Variable	Legal Domain Values
0	A0 B2 C0
1	A3 B0 D1
2	C0
3	B3 D0

Part B

1. Shared edges must agree on the type of figure (circle, square, diamond).
2. A piece (A, B, C, D) cannot be used more than once.

Each of these would be implemented as a constraint function and would reside in the arcs. The first of these is quite similar to the constraint in line-labeling and the second is very similar to the constraint in course-scheduling (not to use courses more than once). Both of these constraints are necessary to get the CSP software to find answers. Note that other possible constraints, e.g. that the pieces match the boundary can be handled by simply eliminating from the domain values that violate them.

Although the second constraint may seem obvious or picky in the context of the puzzle, it is crucial in the context of representing the world to a computer. Knowing when not to overlook "common sense" is a typical and often difficult component of AI. (Problem 4b was one of the harder questions.)

Part C

The lines below represent two directed arcs.

```
       1,2
   Var0 ---- Var1
    | \    / |
    | 2\  /2 |
    |   \/   |
 1,2|   /\   |1,2
    |  /  \  |
    | /    \ |
   Var3 ---- Var2
       1,2
```

Part D

Variable	Legal Domain Values
0	A0 C0
3	B3 D0

Part E

	search step 1	search step 2	search step 3	search step 4
Chosen Var	2	1		
Chosen Value	C0	B0		
Var 0	A0 B2	A0		
Var 1	B0	B0		
Var 2	C0	C0		
Var 3	D0	D0		

No changes happen in the last couple of steps. Note that the C0 value for Var 0 is dropped from its domain because C0 is assigned to Var 2 and the constraint that prevents reuse of parts relates all pairs of vars.