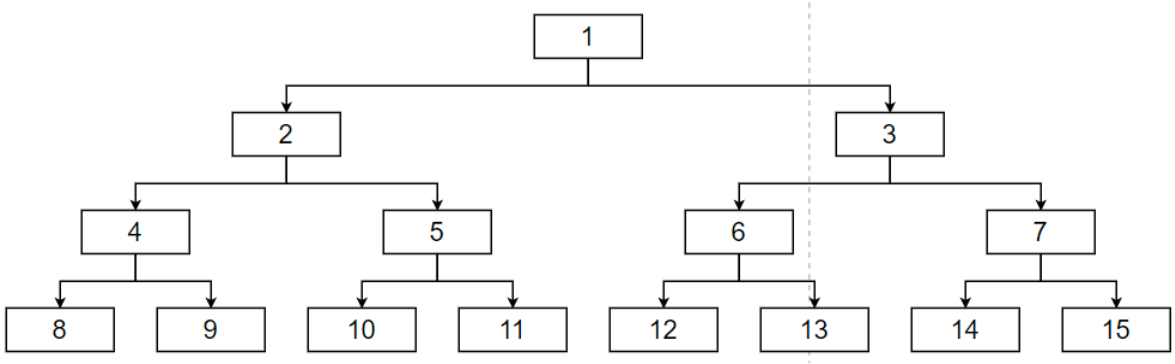
CS303A Homework 1

11811721 庄湛

Q1 Consider a state space where the start state is number 1 and each state k has two successors: numbers 2k and 2k + 1.

1. Draw the portion of the state space for states 1 to 15



2. Suppose the goal state is 11. List the order in which nodes will be visited for breadth first search, depth-limited search with limit 3, and iterative deepening search

> Breadth-first: 1 2 3 4 5 6 7 8 9 10 11
>
> Depth-limited search with limit 3: 1 2 4 8 9 5 10 11
>
> Iterative deepening: 1 1 2 3 1 2 4 5 3 6 7 1 2 4 8 9 5 10 11

3. How well would bidirectional search work on this problem? What is the branching factor in each direction of the bidirectional search?

Bidirectional search do well in this work on this problem. In this search tree, the state of the target can be represented in binary. For example, the binary of 11 is 1011, look at the last three bits from the last one to the third one, if this bit is 1, the target is the right child of its parent, otherwise the target is the left child of its parent.

4. Does the answer to (c) suggest a reformulation of the problem that would allow you to solve the problem of getting from state 1 to a given goal state with almost no search?

Sure. As long as you always use the reverse search, through binary coding and the above rules, you can directly find the right path.

5. Call the action going from k to 2k Left, and the action going to 2k + 1 Right. Can you find an algorithm that outputs the solution to this problem without any search at all?

Shown as **Fig.1**:

Algorithm 1 f(n)

```
1: if n = 1 then
2:     return
3: else if n % 2 = 0 then
4:     Print(Left)
5:     return f(n/2)
6: else
7:     Print(Right)
8:     return f((n-1)/2)
9: end if
```

Fig.1

Q2 The traveling salesperson problem (TSP) can be solved with the minimum-spanning- tree (MST) heuristic, which estimates the cost of completing a tour, given that a partial tour has already been constructed. The MST cost of a set of cities is the smallest sum of the link costs of any tree that connects all the cities.

a. Show how this heuristic can be derived from a relaxed version of the TSP.

Ans: Using a minimum spanning tree would allow the salesman to repeatedly reach the same city and calculate the total distance only by calculating the distance of the first segment, which is a relaxed version of the TSP。

b. Show that the MST heuristic dominates straight-line distance from the current city back to the start city.

Ans: According to the *Cut property* (Let S be any subset of vertices, and let e be the min cost edge with exactly one endpoint in S. Then the MST contains e.), we can split the two cities of each node of the path, so that the edge connecting the two cities must be the least costly edge between the two sets of points. The line segment between two points is the shortest, so the MST heuristic dominates straight-line distance from the current city back to the start city.

c. Design a hill-climbing algorithm and write the procedure to solve the TSP.

The Hill-climbing algorithm is shown as **Fig.2**:

Algorithm 1 Hill-climbing

```
0: Number the cities from 1 to n
0: Initialize res as [1,2,...,n] // The order in which cities are visited
0: Eval ⇐ getTotalDistance(res)
0: t ⇐ 0
1: for t < TriesLimit do
1:     t ⇐ t + 1
1:     Flip the index of two cities at random
1:     TmpEval ⇐ getTotalDistance(res)
2:     if TmpEval < Eval then
2:         Update the optimal route
3:     end if
4: end for
```

Fig.2

d. Repeat part (a) using a genetic algorithm instead of hill climbing.

The GA algorithm is shown as **Fig.3**:

Algorithm 2 GeneticAlgorithm

```
0: Number the cities from 1 to n
0: Initialize N random arrays res[N], each one of them is defined as [1,2,...,n]
   // The order in which cities are visited
0: Eval ⇐ getTotalDistance(res) // Calculate each total distance separately
   and assign to Eval array
0: t ⇐ 0
1: for t < TriesLimit do
1:    t ⇐ t + 1
1:    Select two best candidate routes r1, r2 from res based on Eval
1:    Reproduce two routes n1, n2 from the r1, r2
1:    Add the n1, n2 to the res
1:    Eval ⇐ getTotalDistance(res) // Calculate each total distance sepa-
      rately and assign to Eval array
1:    Remove two worst candidate routes from res based on Eval
1:    Update the optimal route with the best one in Eval
2: end for
```

Fig.3

For the reproduce method, which can be chosed in many way, for example:

Step 1: Randomly select a fixed length interval of r_1 and r_2, such as [i, j].

Step 2: Swap the values of the two arrays r_1 and r_2 in the interval [i, j].

Step 3: Establishes a one-to-one correspondence F between the missing and redundant elements of the two new arrays n_1, n_2, respectively.

Step 4: According to the correspondence F, modify the values outside the fixed interval [i, j] of the two arrays, respectively.

Step 5: Complete the construction of the new array n_1, n_2

Q3 Consider the two-player game described in Figure 1:

Figure 1 The starting position of a simple game. Player A moves first. The two players take turns moving, and each player must move his token to an open adjacent space in either direction. If the opponent occupies an adjacent space, then a player may jump over the opponent to the next open space if any. (For example, if A is on 3 and B is on 2, then A may move back to 1.) The game ends when one player reaches the opposite end of the board. If player A reaches space 4 first, then the value of the game to A is +1; if player B reaches space 1 first, then the value of the game to A is −1.

a. Draw the complete game tree, using the following conventions:

• Write each state as (s_A, s_B), where s_A and s_B denote the token locations.

• Put each terminal state in a square box and write its game value in a circle.

• Put *loop states* (states that already appear on the path to the root) in double square boxes.

Since their value is unclear, annotate each with a "?" in a circle.

Ans:

Shown as **Fig.4**:

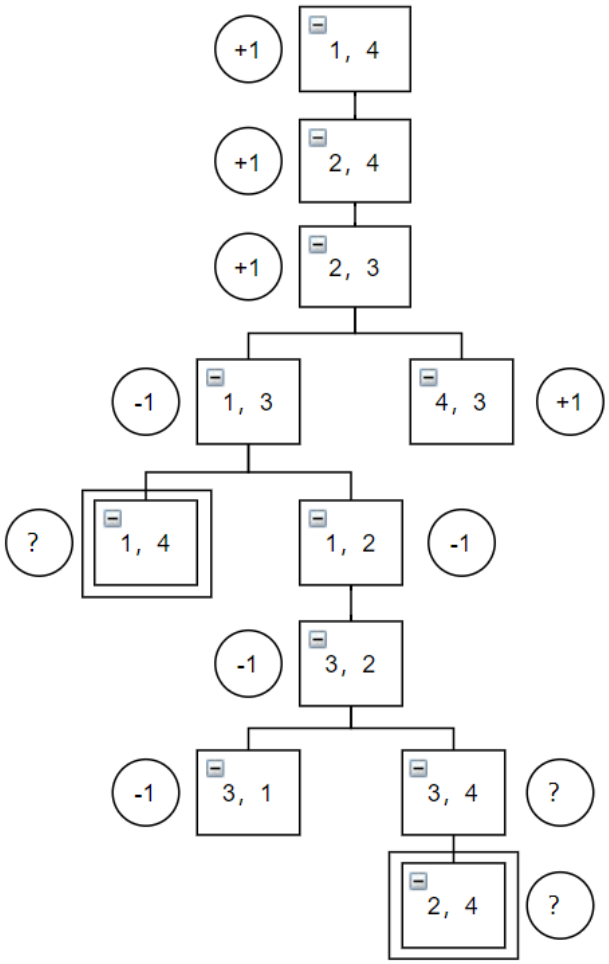


Fig.4

b. Now mark each node with its backed-up minimax value (also in a circle). Explain how you handled the "?" values and why

Ans:

Shown as **Fig.5**:

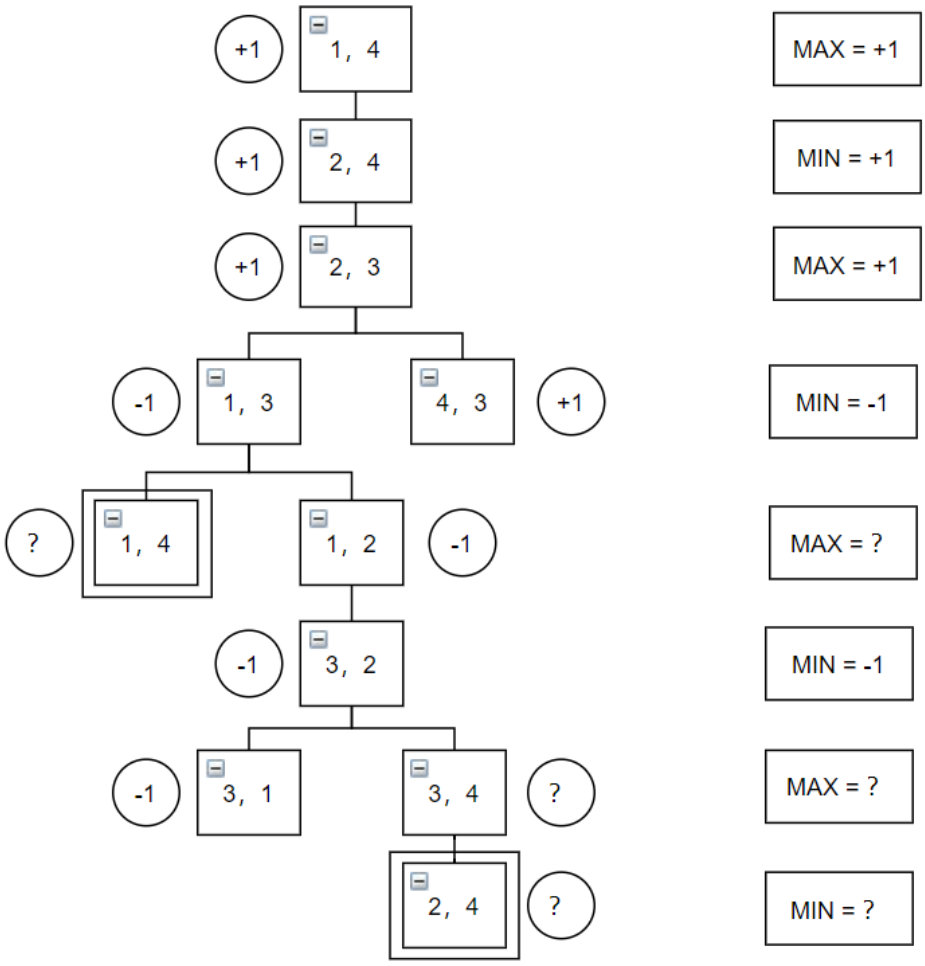


Fig.5

Because "?" means that the value is unclear, so we can treat it larger than -1 and less than +1.

Q4 Consider the graph with 8 nodes A_1, A_2, A_3 A_4, H, T, F_1, F_2. A_i is connected to A_{i+1} for all i, each A_i is connected to H, H is connected to T, and T is connected to each F_i. Find a 3-coloring of this graph by hand using the following strategy: backtracking with conflict-directed backjumping, the variable order A_1, H, A_4, F_1, A_2, F_2, A_3, T, and the value order R, G, B.

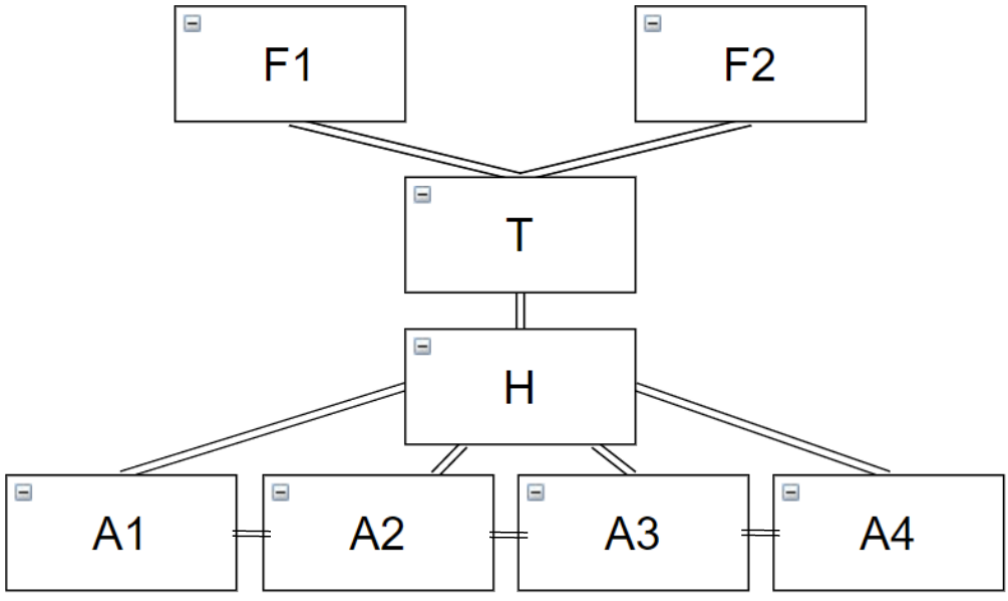


Fig.6

The question is shown in the **Fig.6**

In the first six rounds, there was no conflict, and the result orderly was expressed as:

$\vec{a}_6 = \{red, green, red, red, blue, red\}$

When A_3 is assigned, it conflicts with A_4 if it is red, H if it is green, and A_2 if it is blue.

Because A_4 is an internal dead-end whose connected set is only {H, A_3}, the jumpback set of A_4 includes just H, and the algorithm jumps again, this time back to H. Therefore, algorithm conflict-directed backjumping jumps to A_4. Then A_4 turns to be blue, and then A_3 turns to be red.

Finally T is assigned to be blue smoothly and reasonably. So the result can be expressed as:

$\vec{a}_8 = \{red, green, blue, red, blue, red, red, blue\}$ which means:

A_1 is red

H is $green$

A_4 is $blue$

F_1 is red

A_2 is $blue$

F_2 is red

A_3 is red

T is $blue$