Final Examination Revision AI 2024

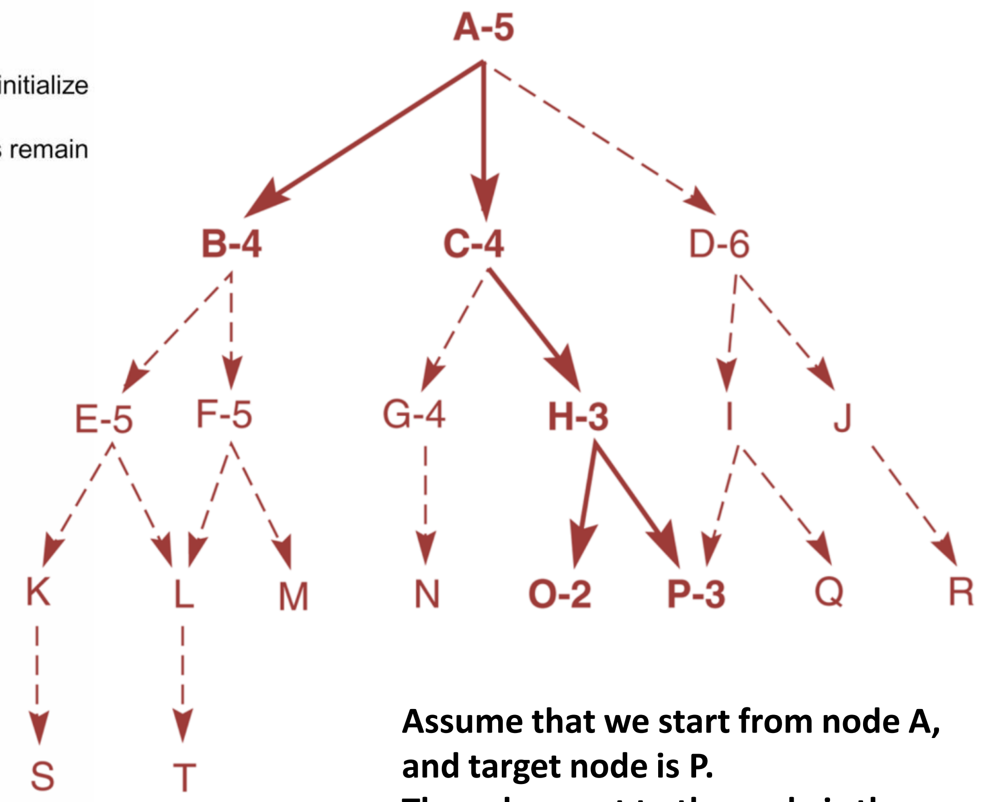
Heuristic Search Algorithm: Best First Search

- It is a general algorithm for heuristically searching any state space graph
- Supports a **variety of heuristic evaluation functions**
- **Better and flexible Algorithm** for heuristic search
- **Avoid local maxima**, dead ends; has open and close lists
- Selects the most promising state
- Apply heuristic and **sort the 'best' next state** in front of the list (priority queue) can jump to any level of the state space
- If **lead to incorrect path**, it **may retrieve the next best state**

Best First Search

```
function best_first_search;

begin
   open := [Start];                                            % initialize
   closed := [ ];
   while open ≠ [ ] do                                         % states remain
      begin
         remove the leftmost state from open, call it X;
         if X = goal then return the path from Start to X
         else begin
                  generate children of X;
                  for each child of X do
                  case
                        the child is not on open or closed:
                           begin
                                 assign the child a heuristic value;
                                 add the child to open
                           end;
                        the child is already on open:
                           if the child was reached by a shorter path
                           then give the state on open the shorter path
                        the child is already on closed:
                           if the child was reached by a shorter path then
                              begin
                                 remove the state from closed;
                                 add the child to open
                              end;
                  end;                                         % case
                  put X on closed;
                  re-order states on open by heuristic merit (best leftmost)
               end;
return FAIL                                                    % open is empty
end.
```



Assume that we start from node A, and target node is P.
The value next to the node is the heuristic value.

Best First Search – The List

open = [A5]; closed = [ ]

evaluate A5; open = [B4,C4,D6]; closed = [A5]

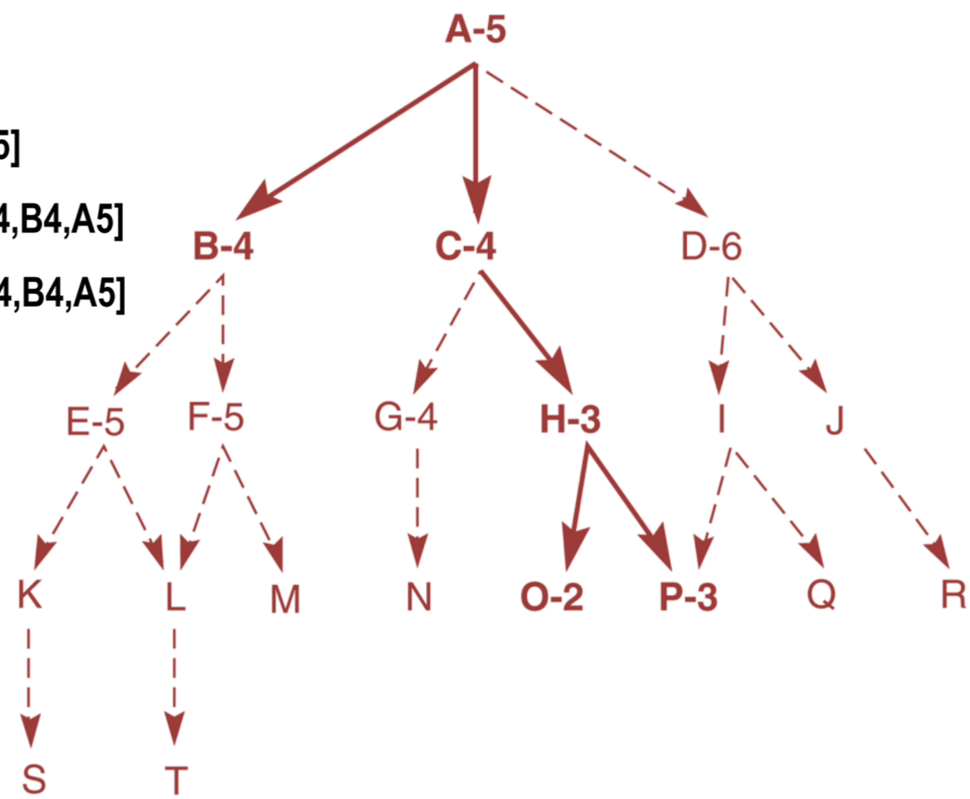
evaluate B4; open = [C4,E5,F5,D6]; closed = [B4,A5]

evaluate C4; open = [H3,G4,E5,F5,D6]; closed = [C4,B4,A5]

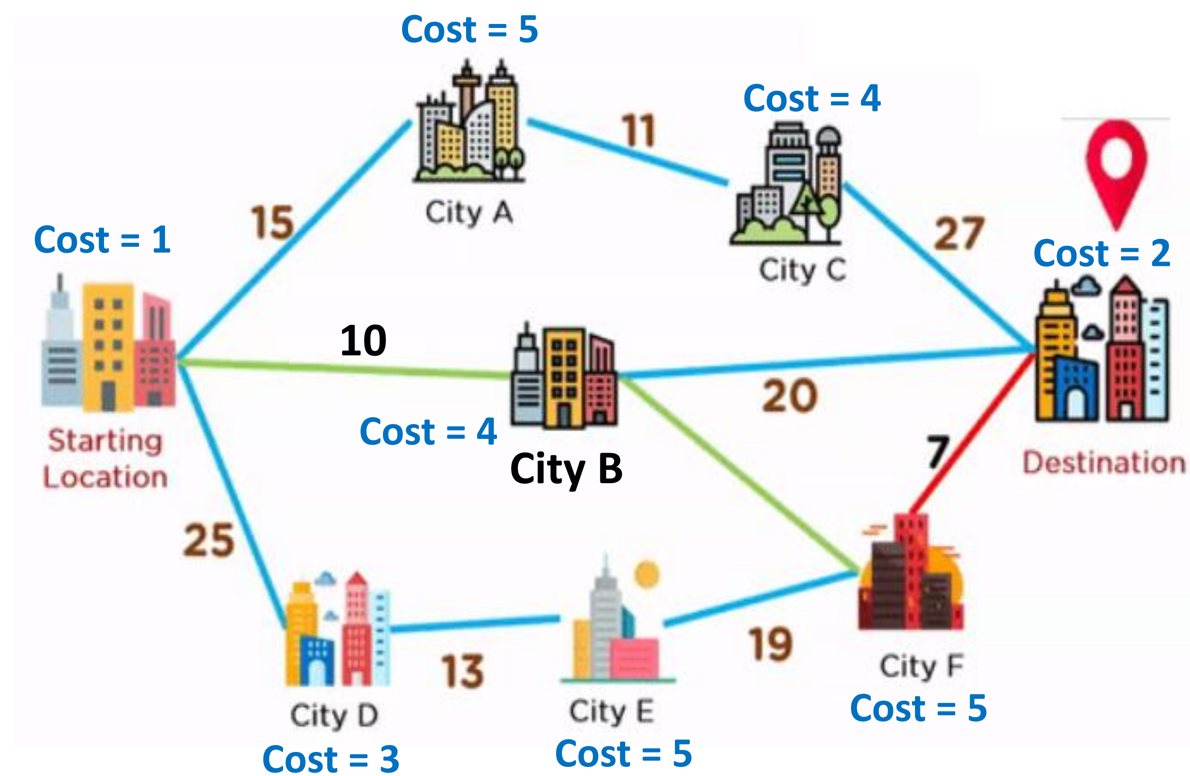
evaluate H3; open = [O2,P3,G4,E5,F5,D6]; closed = [H3,C4,B4,A5]

evaluate O2; open = [P3,G4,E5,F5,D6]; closed = [O2,H3,C4,B4,A5]

evaluate P3; the solution is found!



Revision (1): Heuristic Search Algorithm



Consider the route-finding problem of selecting the best path from **Starting Location** to **Destination**. The heuristic value and the edge value are given based on the legend.

(a) Draw the search tree using Best First Search algorithm in order to find the minimum travelling cost. (please label all nodes with their *h* values).

(b) Identify the list of visited locations found by this algorithm in (a).

Legend:

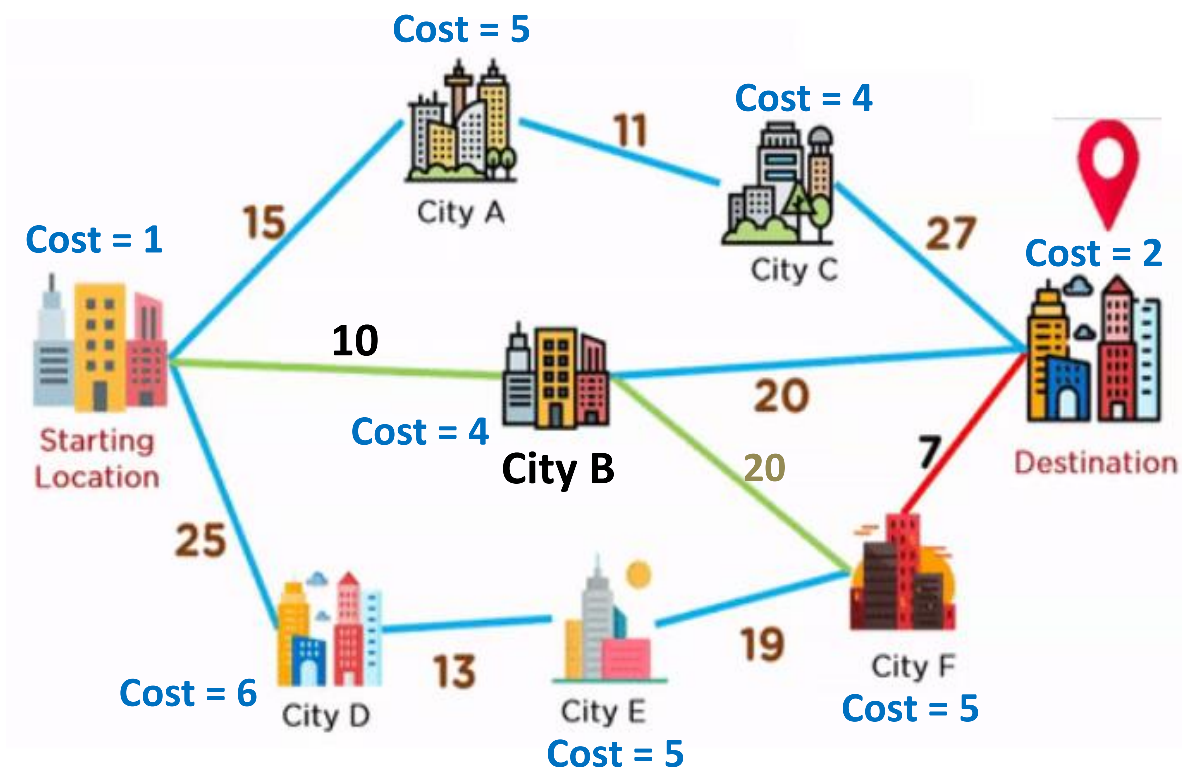
Cost: Heuristic value of every locations

Edges/Path/Arc: Distance between two locations

Heuristic Search Algorithm: A* (Heuristic Evaluation Function f(n))

- To evaluate performances of heuristics for solving a problem
- Devise good heuristic using limited information to make intelligent choices
- To better heuristic, **f(n)=g(n)+h(n)**, where h(n) distance from start to n, **g(n) is the depth measure**
- Eg. 8 puzzle, **heuristics** h(n) could be:
  - No. of tiles in wrong position
  - No. of tiles in correct position
  - Sum of distances out of place
- In other case, **heuristics** h(n) could be:
  - Traveling cost
  - Fuel consumption
  - Budget

Revision (2): Heuristic Search Algorithm



Consider the route-finding problem of selecting the best path from **Starting Location** to **Destination**. The heuristic value and the edge value are given based on the legend.

(a) Draw the search tree using A* algorithm in order to find the minimum travelling cost. (please label all nodes with their *h* values).

(b) Identify the list of visited locations found by this algorithm in (a).

Legend:

Cost: Heuristic value of every locations

Edges/Path/Arc: Distance between two locations

Revision (3) – Intelligent Agent

Case Study: Smart Waste Management System

Imagine a city implementing a Smart Waste Management System that employs IoT-enabled trash bins equipped with sensors. The system aims to optimize waste collection by monitoring the fill levels of the bins in real-time and dynamically planning collection routes for garbage trucks. The primary goals are to reduce operational costs, minimize environmental impact, and improve the overall efficiency of waste management.

i) What are the key percepts and sensors involved in the Smart Waste Management System?

ii) Analyze the actions performed by the IoT-enabled trash bins and the waste collection trucks in the Smart Waste Management System. How does it optimize waste collection routes?

iii) Propose strategies for enhancing the adaptability of the Smart Waste Management System to changing conditions, such as variations in waste generation patterns or unexpected events.

Heuristics in Games : Alpha-Beta Pruning

- To improve search efficiency in two-person games (compared to minimax that always pursues all branches in state space)
- Alpha-beta search in depth-first fashion
- Alpha(α) and beta(β) values are created
- α associates with MAX-never decrease
- β associates with MIN-never increase
- How?
  - Expand to full-ply
  - Apply heuristic evaluation to a state and its siblings
  - Back-up to the parent
  - The value is offered to grandparent as a potential α or β cutoff

Revision (4): Heuristics in Games

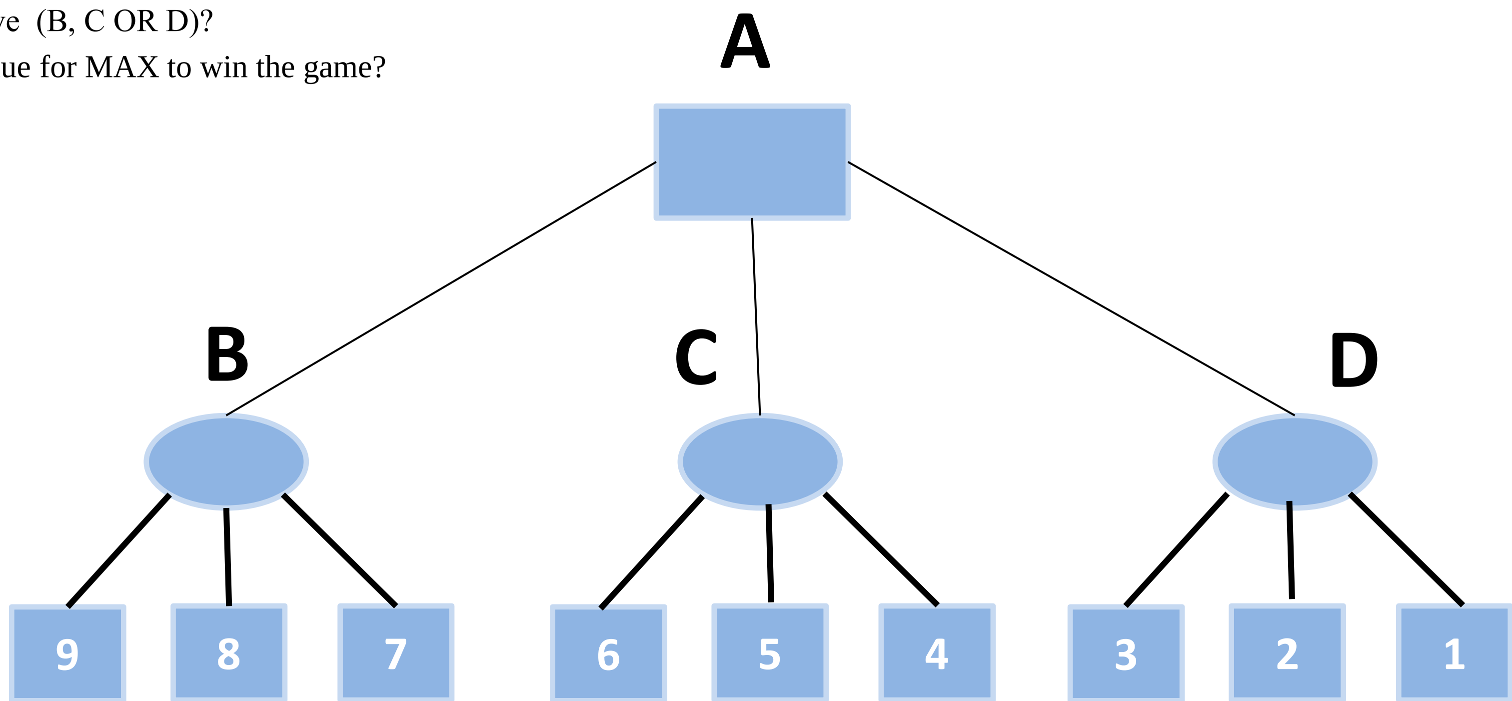
Alpha beta pruning improves search efficiency in two-person games

a) Perform Mini-Max with alpha beta pruning search and calculate the value of each node.
b) Examine and select the nodes that will not be evaluated.
c) Calculate the α and β final values for node C ?
d) What is MAX's best move (B, C OR D)?
e) Calculate the optimal value for MAX to win the game?

At MAX, only update **α**
At MIN, only update **β**

Two rules to stop alpha-beta searching:
For MIN node, if β <= α of MAX ancestor

For MAX node, if α >= β of MIN ancestors



Revision (5): Goal Driven Production System

Table 1

Letter	Fact
H	You are hot
N	You are not hot
W	Window open
O	Open the window
D	Thermostat down
CW	Close window
L	Window closed
C	You are cold

The following rules are knowledge base of a production system. Use this rule set and facts in Table 1 to answer questions (a) and (b).

Rule 1: IF you are hot THEN thermostat down.

Rule 2: IF you are not hot AND window open THEN you are cold

Rule 3: IF thermostat down AND you are cold THEN open the window

(a) Use **GOAL DRIVEN SEARCH / BACKWARD CHAINING** to describe the production system table (Table 1) including its working memory, conflict set, and rule fired.

(b) Assume that the goal is **Open the window (O)**, derive all facts that can be found.

Revision (7): Machine Learning

Naïve Bayesian can be used as a supervised learning algorithm. Figure 1 shows diabetes dataset with actual and predicted class (after Naïve Bayesian classification). Below are the details attributes of diabetes dataset that were extracted from Pima Indians Diabetes Database.

Attributes:

1. Preg - Number of times pregnant
2. Plas - Plasma glucose concentration a 2 hours in an oral glucose tolerance test
3. Pres - Diastolic blood pressure (mm Hg)
4. Skin - Triceps skin fold thickness (mm)
5. Insu - 2-Hour serum insulin (mu U/ml)
6. Mass - Body mass index (weight in kg/(height in m)^2)
7. Pedi - Diabetes pedigree function
8. Age - Age (years)
9. Tested_P - Actual tested result
10. Predicted_P - Predicted result

Revision (7): Machine Learning

Preg	Plas	Pres	Skin	Insu	Mass	Pedi	Age	Tested_P	Predicted_P
1	85	66	29	0	26.6	0.351	31	tested_positive	tested_positive
1	89	66	23	94	28.1	0.167	21	tested_positive	tested_negative
5	116	74	0	0	25.6	0.201	30	tested_positive	tested_negative
10	115	0	0	0	35.3	0.134	29	tested_negative	tested_negative
4	110	92	0	0	37.6	0.191	30	tested_negative	tested_negative
10	139	80	0	0	27.1	1.441	57	tested_negative	tested_negative
1	103	30	38	83	43.3	0.183	33	tested_positive	tested_positive
3	126	88	41	235	39.3	0.704	27	tested_positive	tested_positive
8	99	84	0	0	35.4	0.388	50	tested_negative	tested_negative
1	97	66	15	140	23.2	0.487	22	tested_positive	tested_positive
13	145	82	19	110	22.2	0.245	57	tested_positive	tested_positive
5	117	92	0	0	34.1	0.337	38	tested_negative	tested_positive
5	109	75	26	0	36	0.546	60	tested_negative	tested_positive
3	88	58	11	54	24.8	0.267	22	tested_negative	tested_negative
6	92	92	0	0	19.9	0.188	28	tested_negative	tested_negative
10	122	78	31	0	27.6	0.512	45	tested_positive	tested_positive
4	103	60	33	192	24	0.966	33	tested_positive	tested_positive
11	138	76	0	0	33.2	0.42	35	tested_negative	tested_positive
3	180	64	25	70	34	0.271	26	tested_negative	tested_negative
7	133	84	0	0	40.2	0.696	37	tested_negative	tested_positive

Figure 1

Based on the dataset given in Figure 1, answer the following questions:

a) Quantify the amount of True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) for the classification result

b) Using your answers in a), calculate the accuracy, precision and recall for this classification