Lecture 4

Note: Some slides and/or pictures are adapted from Lecture slides / Books of

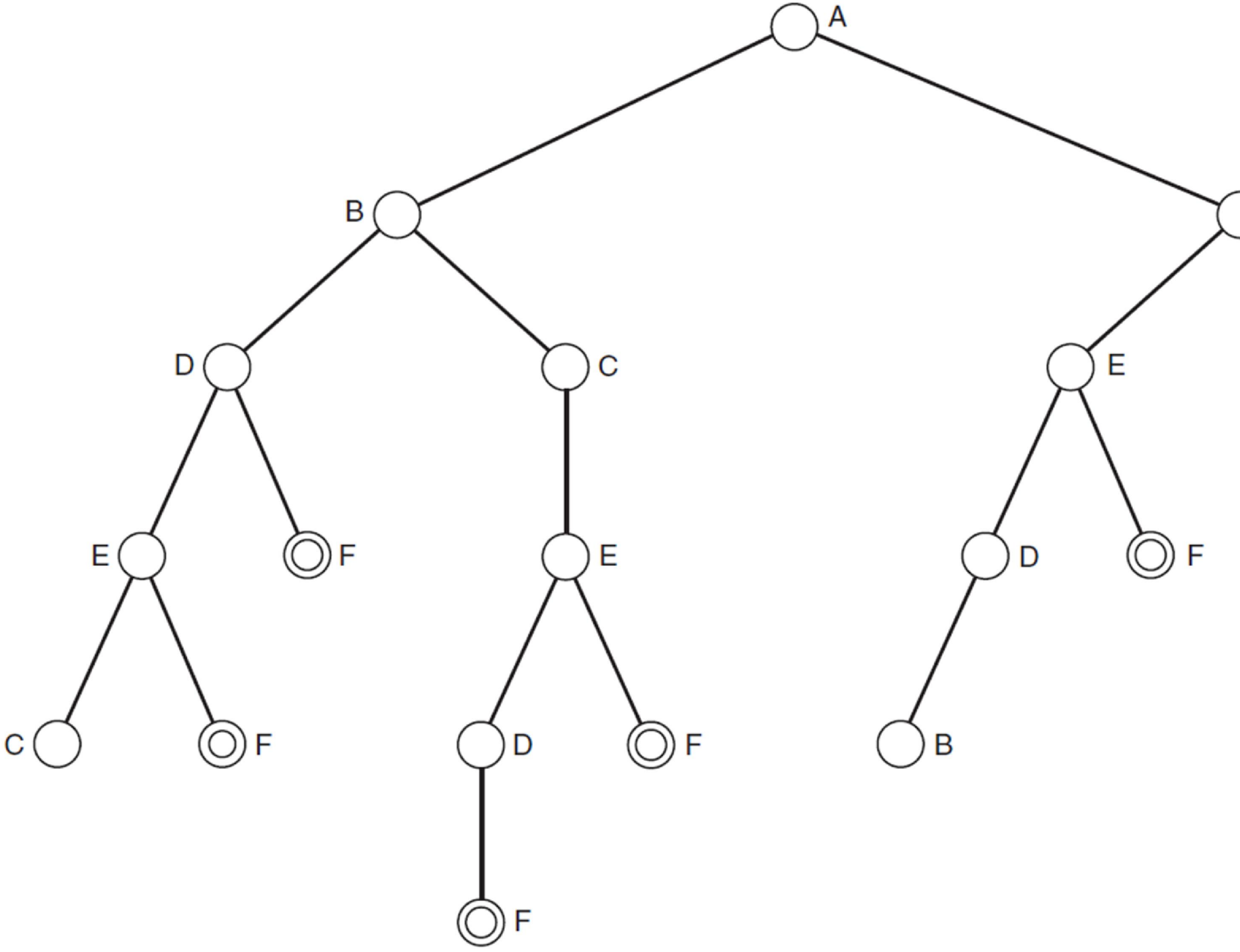
- Dr Zafar Alvi.
- Text Book - *Aritificial Intelligence Illuminated* by Ben Coppin, Narosa Publishers.
- Ref Books
  - *Artificial Intelligence- Structures & Strategies for Complex Problem Solving by* George F. Luger, 4th edition, Pearson Education.
  - *Artificial Intelligence A Modern Approach* by Stuart Russell & Peter Norvig.
  - Artificial Intelligence, Third Edition by Patrick Henry Winston

Informed vs Uninformed Methods

- A search method is **informed** if it uses additional information about nodes that have not yet been explored to decide which nodes to examine next.
- If a method is not informed, it is **uninformed, or blind.**
- In other words, search methods that use heuristics are informed, and those that do not are blind.

Heuristic

- A **heuristic evaluation function** is a function that when applied to a node gives a value that represents a good estimate of the distance of the node from the goal.
- For two nodes *m and n, and a heuristic function f, if f(m) < f(n), then it should be the case that m is more likely to be on an* ***optimal*** **path** to the goal node than *n.*
- *In other words, the lower the heuristic value* of a node, the more likely it is that it is on an optimal path to a goal and the more sensible it is for a search method to examine that node.


A
B
D
C
E
E
F
D
F
C
F
E
D
F
C
F
D
F
B

Distance Details

Cities	Distance from F
A	25
B	8
C	20
D	6
E	12
F	0

Table 4.4 Analysis of hill climbing

Step	State	Queue	Notes
1	A	(empty)	The queue starts out empty, and the initial state is the root node, which is A.
2	A	B,C	The successors of A are sorted and placed on the queue. B is placed before C on the queue because it is closer to the goal state, F.
3	B	C	
4	B	D,C,C	
5	D	C,C	
6	D	F,E,C,C	F is placed first on the queue because it is closest to the goal. In fact, it is the goal, as will be discovered in the next step.
7	F	E,C,C	SUCCESS: Path is reported as A,B,D,F.

Table 4.5 Analysis of best-first search of tree shown in Figure 4.4

Step	State	Queue	Notes
1	A	(empty)	The queue starts out empty, and the initial state is the root node, which is A.
2	A	B,C	The successors of the current state, B and C, are placed in the queue.
	A	B,C	The queue is sorted, leaving B in front of C because it is closer to the goal state, F.
3	B	C	
4	B	D,C,C	The children of node B are added to the front of the queue.
5	B	D,C,C	The queue is sorted, leaving D at the front because it is closer to the goal node than C.
6	D	C,C	Note that although the queue appears to contain the same node twice, this is just an artifact of the way the search tree was constructed. In fact, those two nodes are distinct and represent different paths on our search tree.
7	D	E,F,C,C	The children of D are added to the front of the queue.
8	D	F,E,C,C	The queue is sorted, moving F to the front.
9	F	E,C,C	SUCCESS: Path is reported as A,B,D,F.

Table 4.6 Analysis of beam search of tree shown in Figure 4.7

Step	State	Queue	Notes
1	A	(empty)	The queue starts out empty, and the initial state is the root node, which is A.
2	A	B,C	The two children of the current node are added to the back of the queue.
3	B	C	
4	B	C,D,C	The two children of B are added to the back of the queue.
5	B	D,C	All but the two best paths are discarded from the queue.
6	D	C	
7	D	C,E,F	The two children of the current node are added to the back of the queue.
8	D	E,F	At this step, C is removed from the queue because we only require the two best paths.
9	E	F	
10	E	F,C,F	The two children of E are added to the back of the queue.
11	E	F,F	The path that leads to C is discarded, in favor of the two better paths, both of which lead to F.
12	F	F	SUCCESS: Path is reported as A,B,D,E,F.

The Knapsack (Fractional) Problem

- A man is packing items into his knapsack.
- He wants to take the most valuable items he can, but there is a limit on how much weight he can fit in his knapsack.
- Each item has a weight *wi and is worth vi.*
- *He can only fit a total* weight of *Win his knapsack.*
- *The items that he wants to take are things that* can be broken up and still retain their value (like flour or milk), and he is able to take fractions of items.
- Hence, the problem is called the *fractional* knapsack problem.
- In solving this problem, a greedy-search algorithm provides the best solution.

The Knapsack(0-1) Problem

- The 0-1 knapsack problem is the same as the fractional knapsack problem, except that he cannot take parts of items.
- Each item is thus something like a television set or a laptop computer, which must be taken whole.
- In solving this problem, a greedy-search approach does not work.

Example

- Our man has a knapsack that lets him carry a total of 100 pounds. His items are:
  - 1 gold brick worth $1800 and weighing 50 pounds
  - 1 platinum brick worth $1500 and weighing 30 pounds
  - 1 laptop computer worth $2000 and weighing 50 pounds
- Hence, we have three items, whose values of *v and w are as follows:*
  - *v1 = 1800 w1 = 50 v1/w1 = 36*
  - *v2 = 1500 w2 = 30 v2/w2 = 50*
  - *v3 = 2000 w3 = 50 v3/w3 = 40*
- In this case, a greedy-search strategy would pick item 2 first, and then would take item 3, giving a total weight of 80 pounds, and a total value of $3500.
- In fact, the best solution is to take items 1 and 3 and to leave item 2 behind giving a total weight of 100 pounds and a total value of $3800.