Algorithms and Complexity

Branch and Bound with Heuristic Techniques

Lecturer: Dr. Alaa Ahmed Abbood
Lecture 10.
Class 2nd .
Time: 8:30-10:30
Department: Businesses Information Technology (BIT)

Backtracking

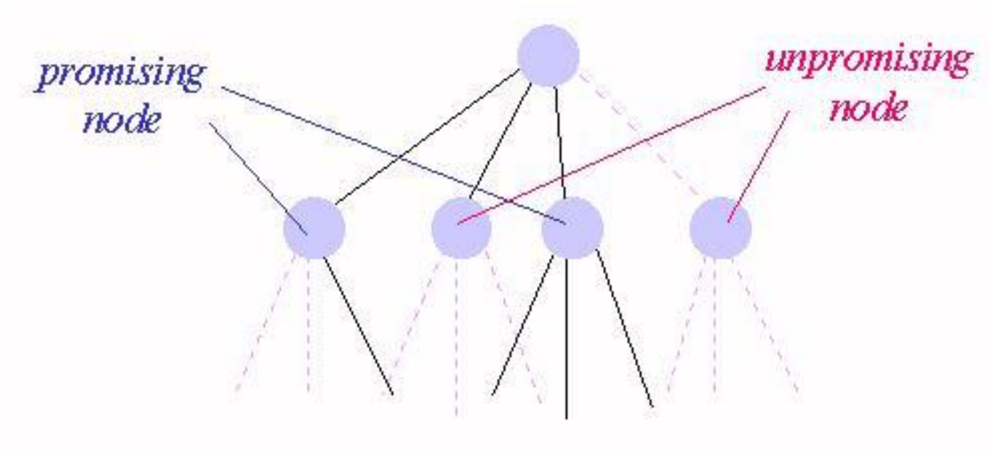
- Each decision leads to a new set of choices
- Some sequence of choices (possibly more than one) may be a solution to your problem

- Backtracking ***is a methodical way of trying out various sequences of decisions, until you find one that "works"***

Branch-and-bound Problem

- The branch-and-bound problem solving method is very similar to backtracking in that a state space tree is used to solve a problem. The differences are that B&B

(1)does not limit us to any particular way of traversing the tree and
(2) is used only for optimization problems.



Branch-and-bound Problem Cont..

- A B&B algorithm computes a number (bound) at a node to determine whether the node is promising.
- The number is a bound on the value of the solution that could be obtained by expanding the state space tree beyond the current node.

Types of Traversal

- When implementing the branch-and-bound approach there is no restriction on the type of state-space tree traversal used. Backtracking, for example, is a simple type of B&B that uses depth-first search.

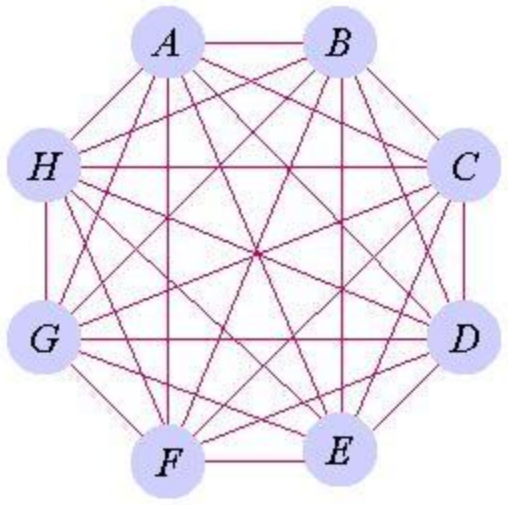
- A better approach is to check all the nodes reachable from the currently active node (breadth-first) and then to choose the most promising node (best-first) to expand next.

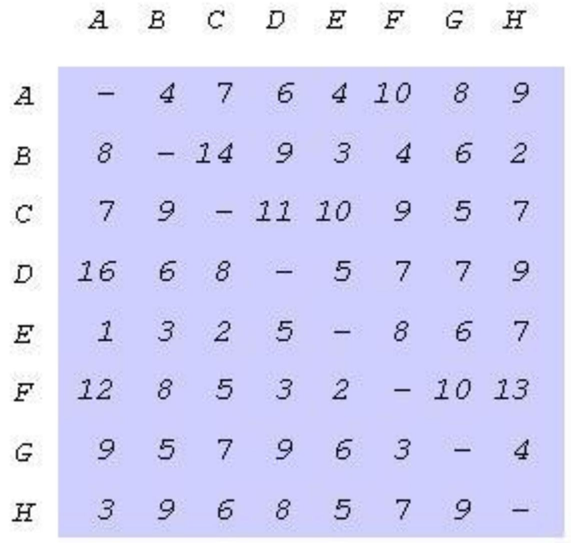
Types of Traversal Cont..

- An essential element of B&B is a greedy way to estimate the value of choosing one node over another. This is called the bound or bounding heuristic. It is an underestimate for a minimal search and an overestimate for a maximal search.
- When performing a breadth-first traversal, the bound is used only to prune the unpromising nodes from the state-space tree. In a best-first traversal the bound cab also used to order the promising nodes on the live-node list.

Traveling Salesperson B&B

In the most general case the distances between each pair of cities is a positive value with dist(A,B) ≠ dist(B,A). In the matrix representation, the main diagonal values are omitted (i.e. dist(A,A) = 0).



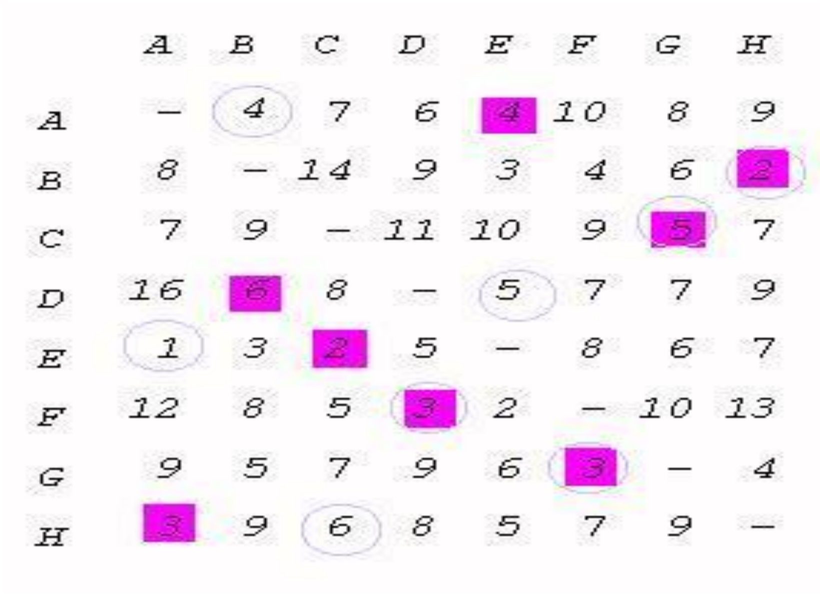


	A	B	C	D	E	F	G	H
A	–	4	7	6	4	10	8	9
B	8	–	14	9	3	4	6	2
C	7	9	–	11	10	9	5	7
D	16	6	8	–	5	7	7	9
E	1	3	2	5	–	8	6	7
F	12	8	5	3	2	–	10	13
G	9	5	7	9	6	3	–	4
H	3	9	6	8	5	7	9	–

Traveling Salesperson B&B

- ***Obtaining an Initial Tour*** Use the greedy method to find an initial candidate tour. We start with city A (arbitrary) and choose the closest city (in this case E). Moving to the newly chosen city, we always choose the closest city that has not yet been chosen until we return to A.

If start with A then (A-E-C-G-F-D-B-H-A)
If start from e then (E-A-B-H-C-G-F-D-E)



	A	B	C	D	E	F	G	H
A	–	4	7	6	4	10	8	9
B	8	–	14	9	3	4	6	2
C	7	9	–	11	10	9	5	7
D	16	6	8	–	5	7	7	9
E	1	3	2	5	–	8	6	7
F	12	8	5	3	2	–	10	13
G	9	5	7	9	6	3	–	4
H	3	9	6	8	5	7	9	–

Traveling Salesperson B&B Cont..

Our candidate tour is

A-E-C-G-F-D-B-H-A

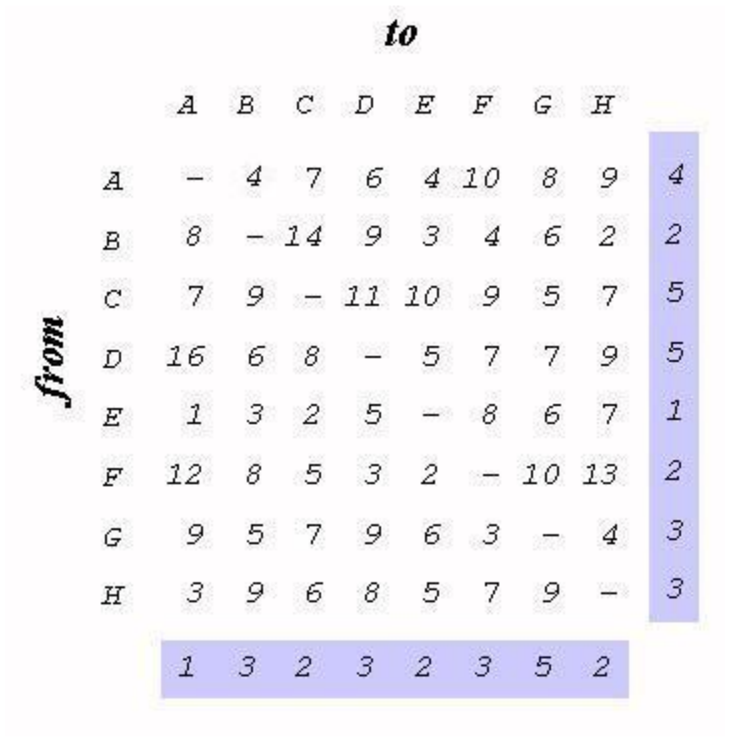
- with a tour length of 28. We know that a minimal tour will be no greater than this.

- It is important to understand that the initial candidate tour is not necessarily a minimal tour nor is it unique.
- If we start with city E for example, we have,

E-A-B-H-C-G-F-D-E

with a length of 30

Defining a Bounding Heuristic



to / from

	A	B	C	D	E	F	G	H	
A	–	4	7	6	4	10	8	9	4
B	8	–	14	9	3	4	6	2	2
C	7	9	–	11	10	9	5	7	5
D	16	6	8	–	5	7	7	9	5
E	1	3	2	5	–	8	6	7	1
F	12	8	5	3	2	–	10	13	2
G	9	5	7	9	6	3	–	4	3
H	3	9	6	8	5	7	9	–	3
	1	3	2	3	2	3	5	2	

Now we must define a bounding heuristic that provides an underestimate of the cost to complete a tour from any node using local information. In this example we choose to use the actual cost to reach a node plus the minimum cost from every remaining node as our bounding heuristic.

$$h(x) = a(x) + g(x)$$

a(x) - actual cost to node x

g(x) - minimum cost to complete

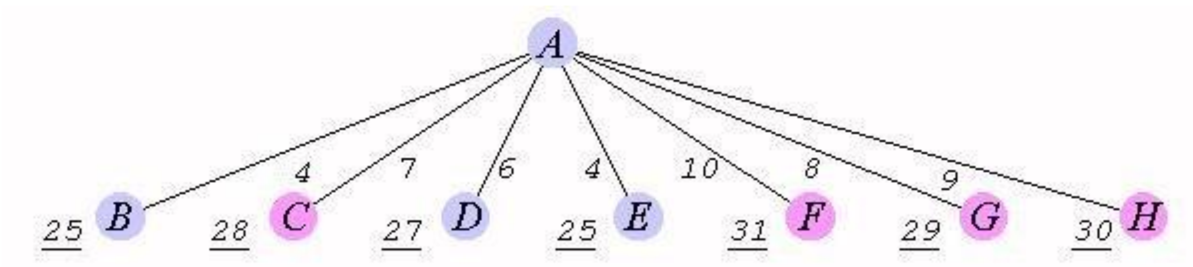
Since we know the minimum cost from each node to anywhere we know that the minimal tour cannot be less than 25.

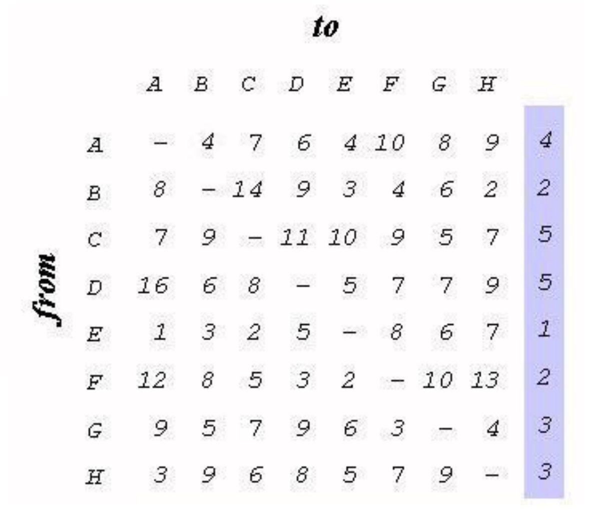
Finding the Minimal Tour

1. Starting with node A determine the actual cost to each of the other nodes.

2. Now compute the minimum cost from every other node 25-4=21.

3. Add the actual cost to a node a(x) to the minimum cost from every other node to determine a lower bound for tours from A through B,C,. . ., H.




to

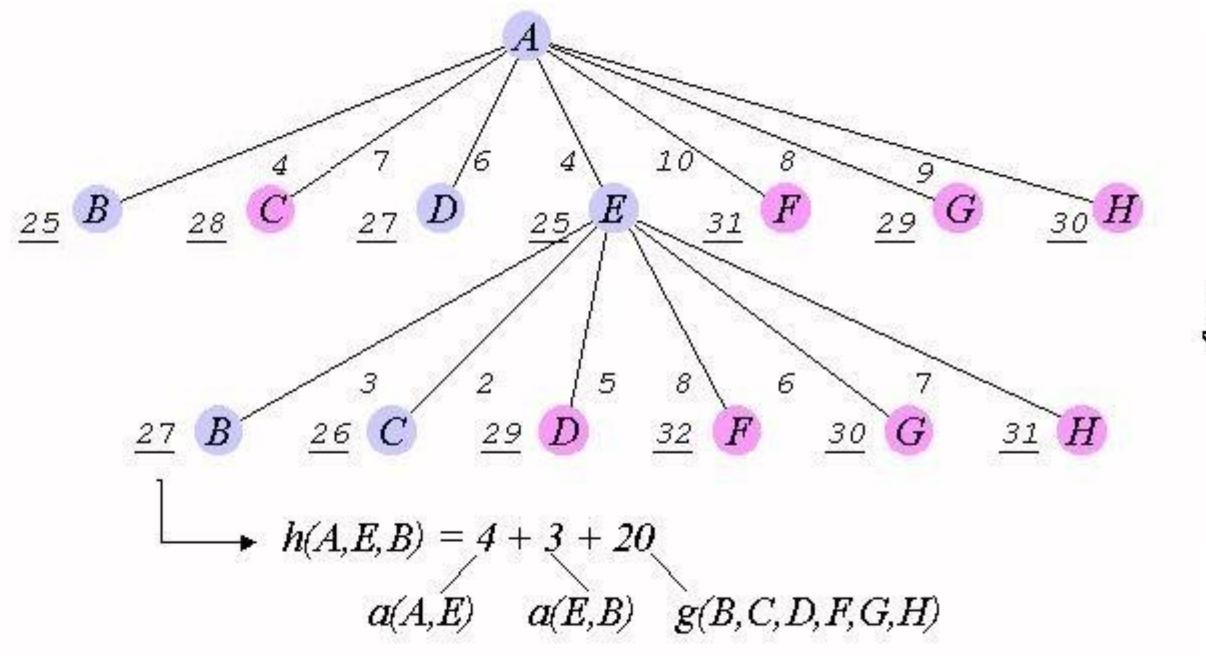
from	A	B	C	D	E	F	G	H	
A	–	4	7	6	4	10	8	9	4
B	8	–	14	9	3	4	6	2	2
C	7	9	–	11	10	9	5	7	5
D	16	6	8	–	5	7	7	9	5
E	1	3	2	5	–	8	6	7	1
F	12	8	5	3	2	–	10	13	2
G	9	5	7	9	6	3	–	4	3
H	3	9	6	8	5	7	9	–	3

Finding the Minimal Tour Cont..

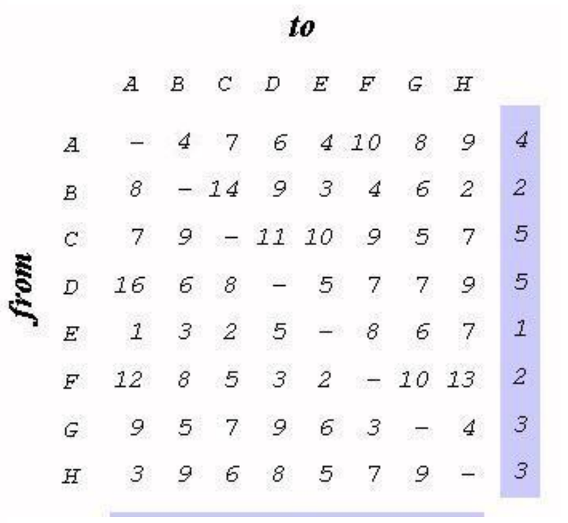
4. Since we already have a candidate tour of 28 we can prune all branches with a lower-bound that is > 28. This leaves B, D and E as the only promising nodes.
5. We continue to expand the promising nodes in a best-first order (E1,B1,,D1).

Finding the Minimal Tour Cont..

5. We continue to expand the promising nodes in a best-first order (E1,B1,,D1).



$$h(A,E,B) = \underbrace{4}_{a(A,E)} + \underbrace{3}_{a(E,B)} + \underbrace{20}_{g(B,C,D,F,G,H)}$$

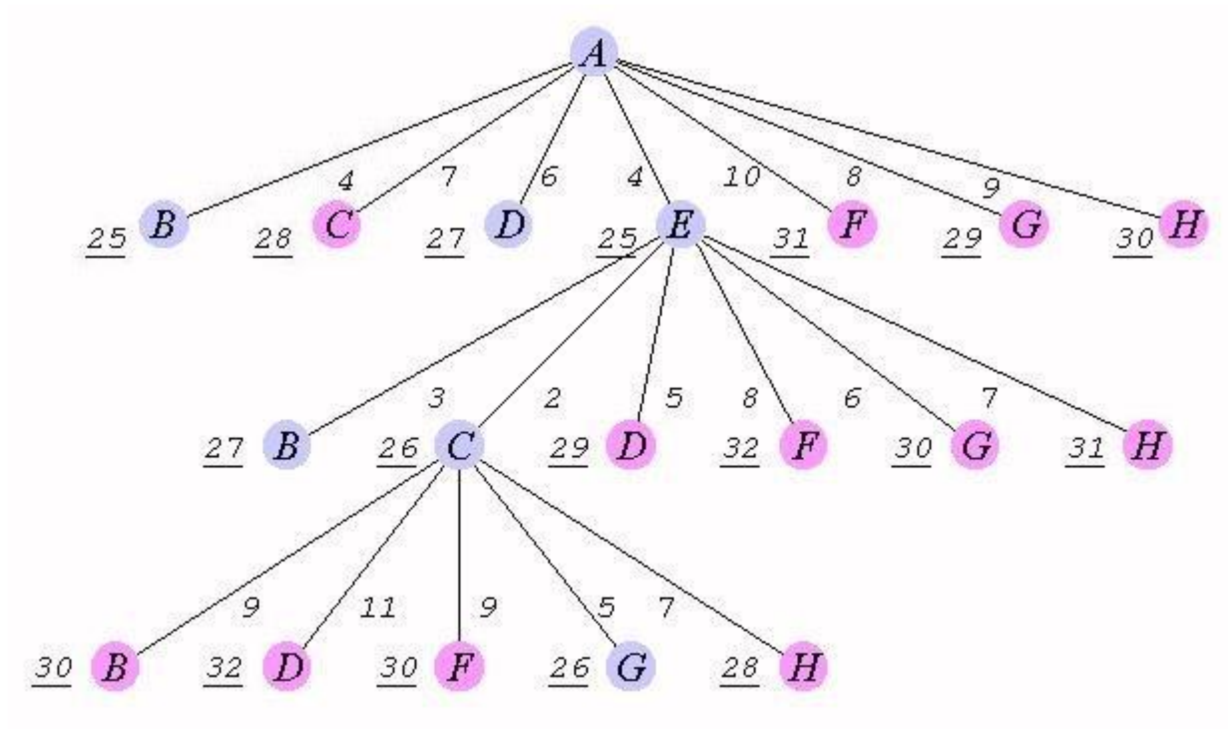


to

from	A	B	C	D	E	F	G	H	
A	–	4	7	6	4	10	8	9	4
B	8	–	14	9	3	4	6	2	2
C	7	9	–	11	10	9	5	7	5
D	16	6	8	–	5	7	7	9	5
E	1	3	2	5	–	8	6	7	1
F	12	8	5	3	2	–	10	13	2
G	9	5	7	9	6	3	–	4	3
H	3	9	6	8	5	7	9	–	3

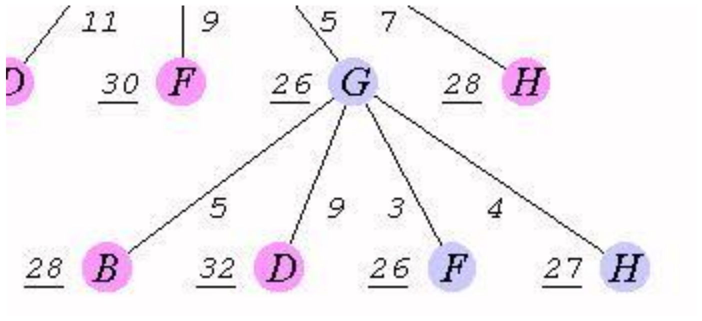
Finding the Minimal Tour Cont..

6. We have fully expanded node E so it is removed from our live-node list and we have two new nodes to add to the list. When two nodes have the same bounding values we will choose the one that is closer to the solution. So the order of nodes in our live-node list is (C2,B1,B2,D1).

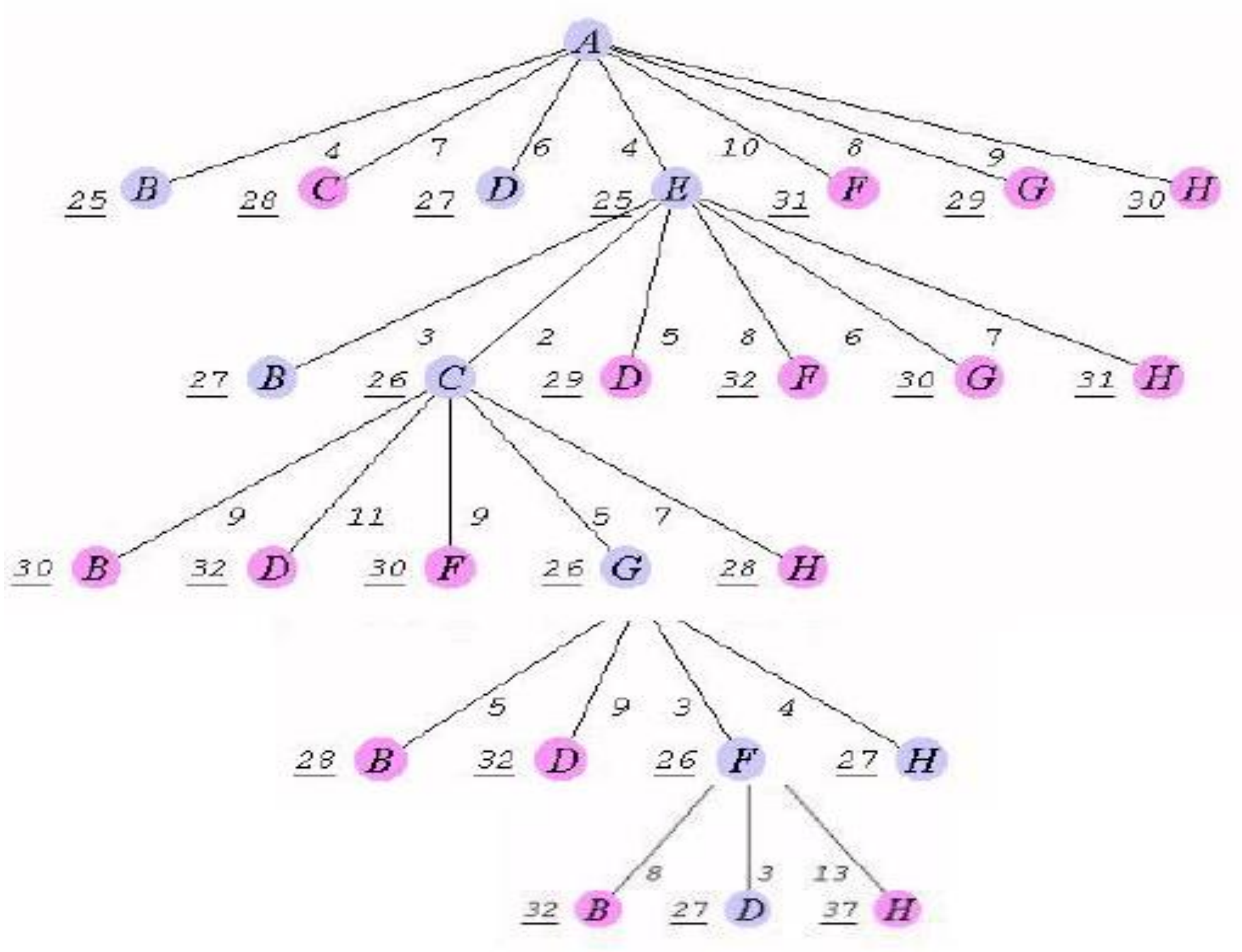


Finding the Minimal Tour Cont..

7. We remove node C2 and add node G3 to the live-node list. (G3,B1,B2,D1).

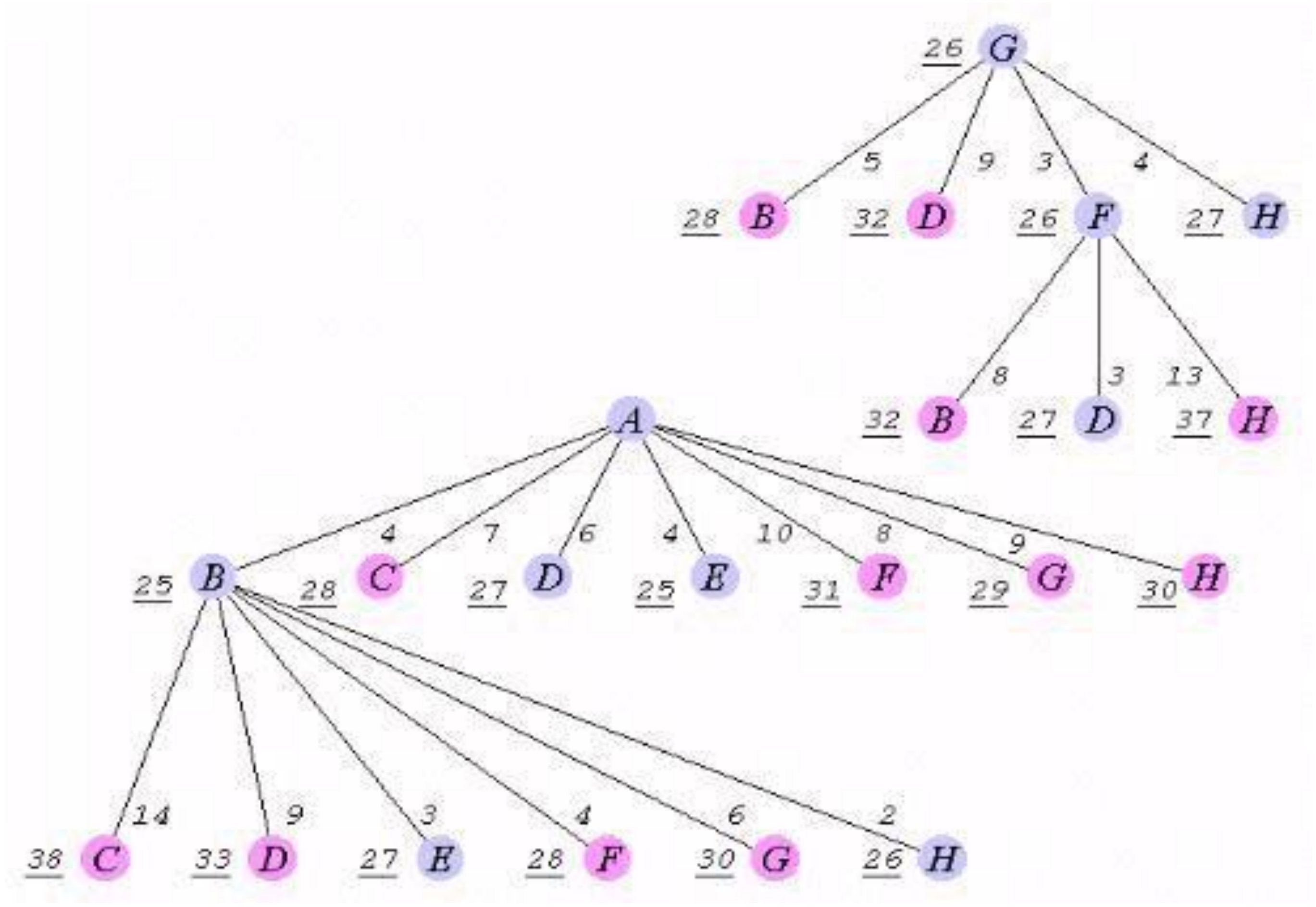


Finding the Minimal Tour Cont..



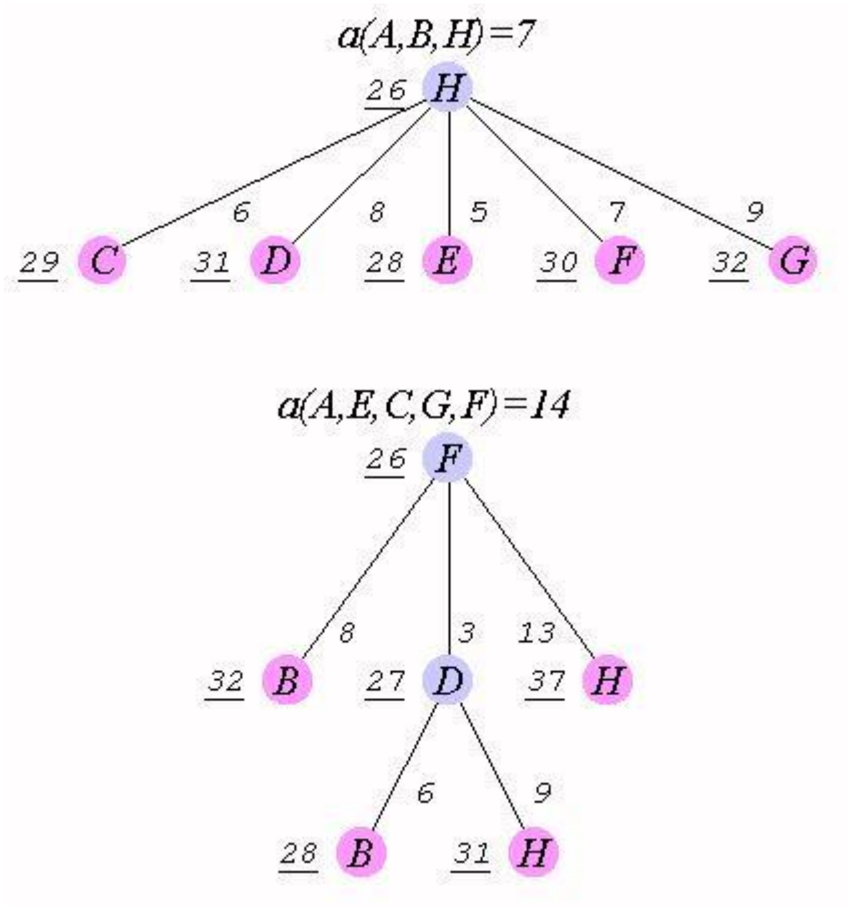
Finding the Minimal Tour Cont..

8. Expanding node G gives us two more promising nodes F4 and H4. Our live-node list is now (F4,B1,H4,B2,D1).



9. A 1st level node has moved to the front of the live-node list. (B1,D5,H4,B2,D1)

10. (H2,D5,H4,B2,D1)



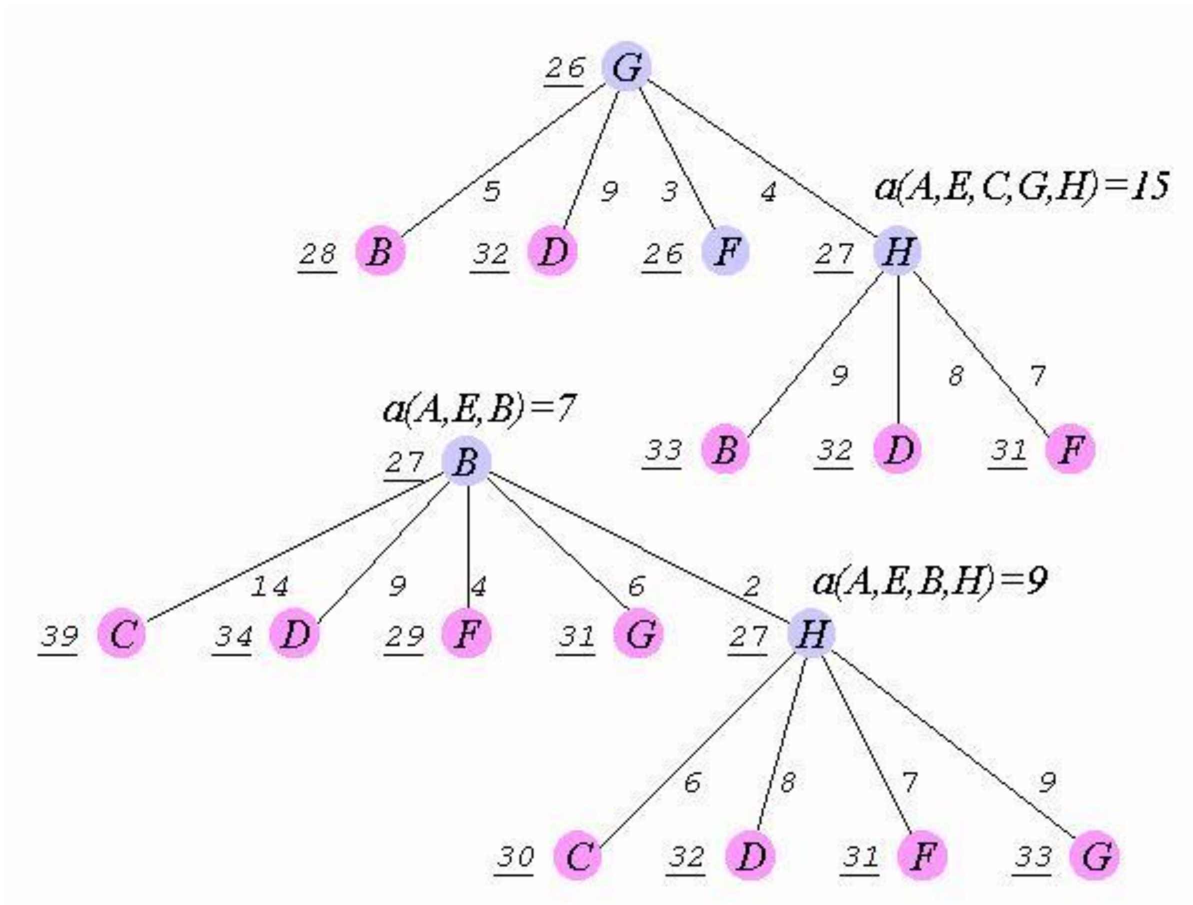
11. (D5,H4,B2,D1)

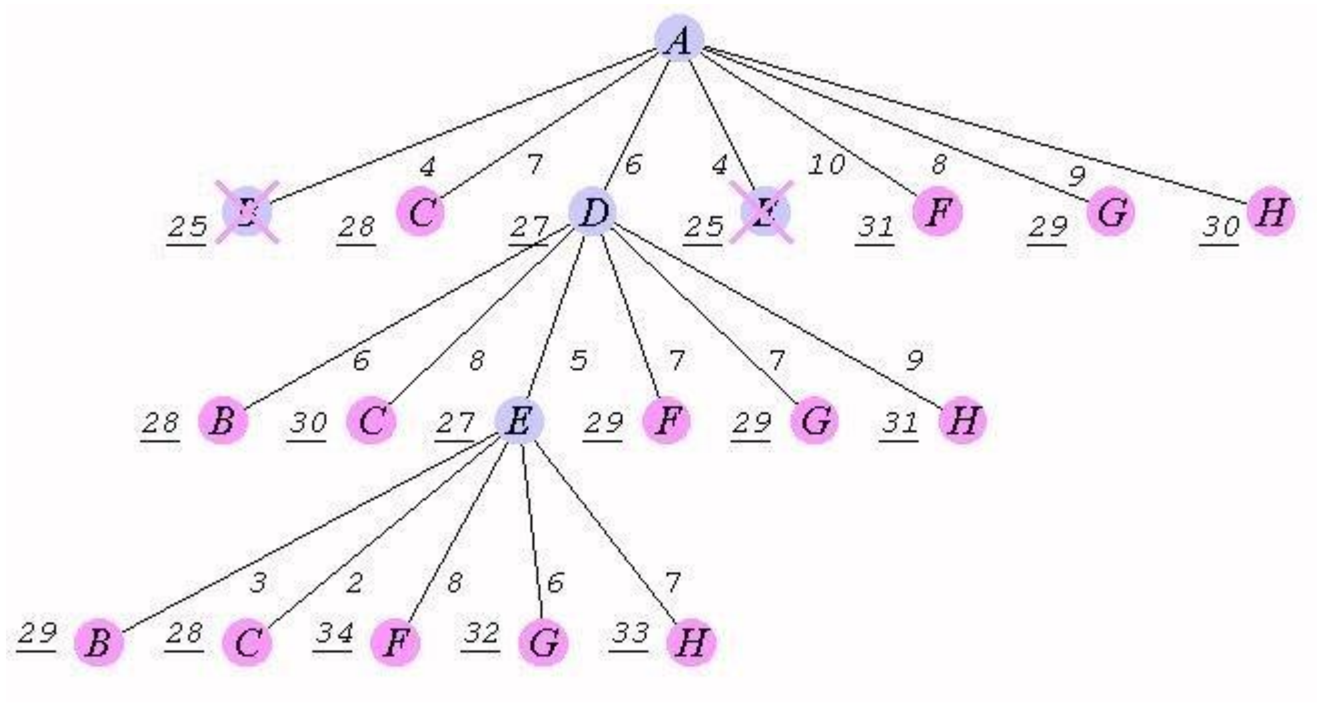
12. (H4,B2,D1)

13. (B2,D1)

14. (H3,D1)

15. (D1)


26 G
5
9
3
4
28 B
32 D
26 F
27 H
a(A,E,C,G,H)=15
9
8
7
33 B
32 D
31 F
a(A,E,B)=7
27 B
14
9
4
6
2
39 C
34 D
29 F
31 G
27 H
a(A,E,B,H)=9
6
8
7
9
30 C
32 D
31 F
33 G



16.We have verified that a minimal tour has length >= 28. Since our candidate tour has a length of 28 we now know that it is a minimal tour, we can use it without further searching of the state-space tree.

THANK YOU