Lecture 7

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

Identifying Optimal Paths

- So far we have looked at uninformed and informed searches.
- Both have their advantages and disadvantages.
- But one thing that lacks in both is that whenever they find a solution they immediately stop.
- They never consider that their might be more than one solution to the problem and the solution that they have ignored might be the optimal one.
- A simplest approach to find the optimal solution is this; find all the possible solutions using either an uninformed search or informed search and once you have searched the whole search space and no other solution exists, then choose the most optimal amongst the solutions found.
- This approach is analogous to the brute force method and is also called the British museum procedure.

Identifying Optimal Paths (Cont.)

- But in reality, exploring the entire search space is never feasible and at times is not even possible, for instance, if we just consider the tree corresponding to a game of chess (we will learn about game trees later), the effective branching factor is 16 and the effective depth is 100.
- The number of branches in an exhaustive survey would be on the order of 10120.
- Hence a huge amount of computation power and time is required in solving the optimal search problems in a brute force manner.

Identifying Optimal Paths (Cont.)

- The following more sophisticated techniques for identifying optimal paths are outlined in this section:
- Uniform cost search (Branch and Bound)
- A*
- Greedy search

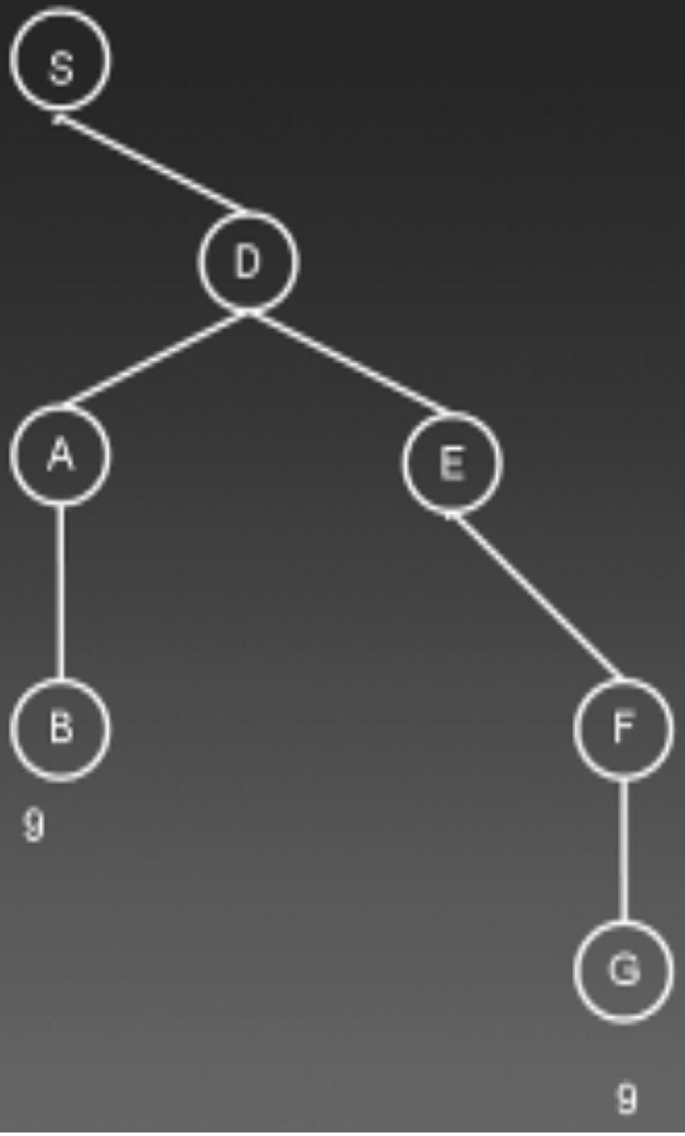
Uniform Cost Search (Branch and Bound)

- **Uniform cost search** (or **Branch and Bound**) is a variation on best-first search that uses the evaluation function g(node), which for a given node evaluates to the cost of the path leading to that node.
- In other words, this is an A* algorithm but where h(node) is set to zero.
- At each stage, the path that has the lowest cost so far is extended.
- In this way, the path that is generated is likely to be the path with the lowest overall cost, but this is not guaranteed.
- To find the best path, the algorithm needs to continue running after a solution is found, and if a preferable solution is found, it should be accepted in place of the earlier solution.

- Uniform cost search is complete and is optimal, provided the cost of a path increases monotonically.
- In other words, if for every node m that has a successor n, it is true that $g(m) < g(n)$, then uniform cost is optimal.
- If it is possible for the cost of a node to be less than the cost of its parent, then uniform cost search may not find the best path.
- Uniform cost search was invented by Dijkstra in 1959 and is also known as Dijkstra's algorithm.

Branch and Bound

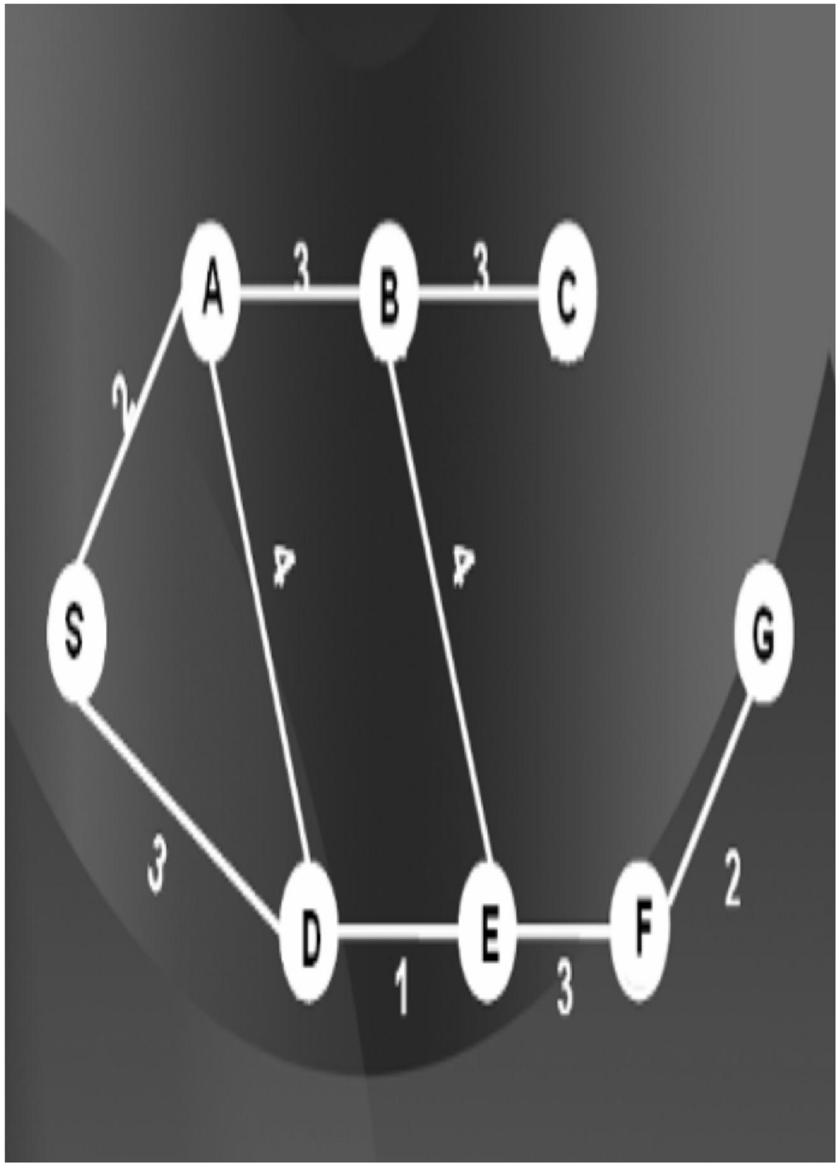
The length of complete path from S to G, S-D-E-F-G is 9. Similarly the length of the partial path S-D-A-B also is 9 and any additional movement along a branch will make it longer than 9

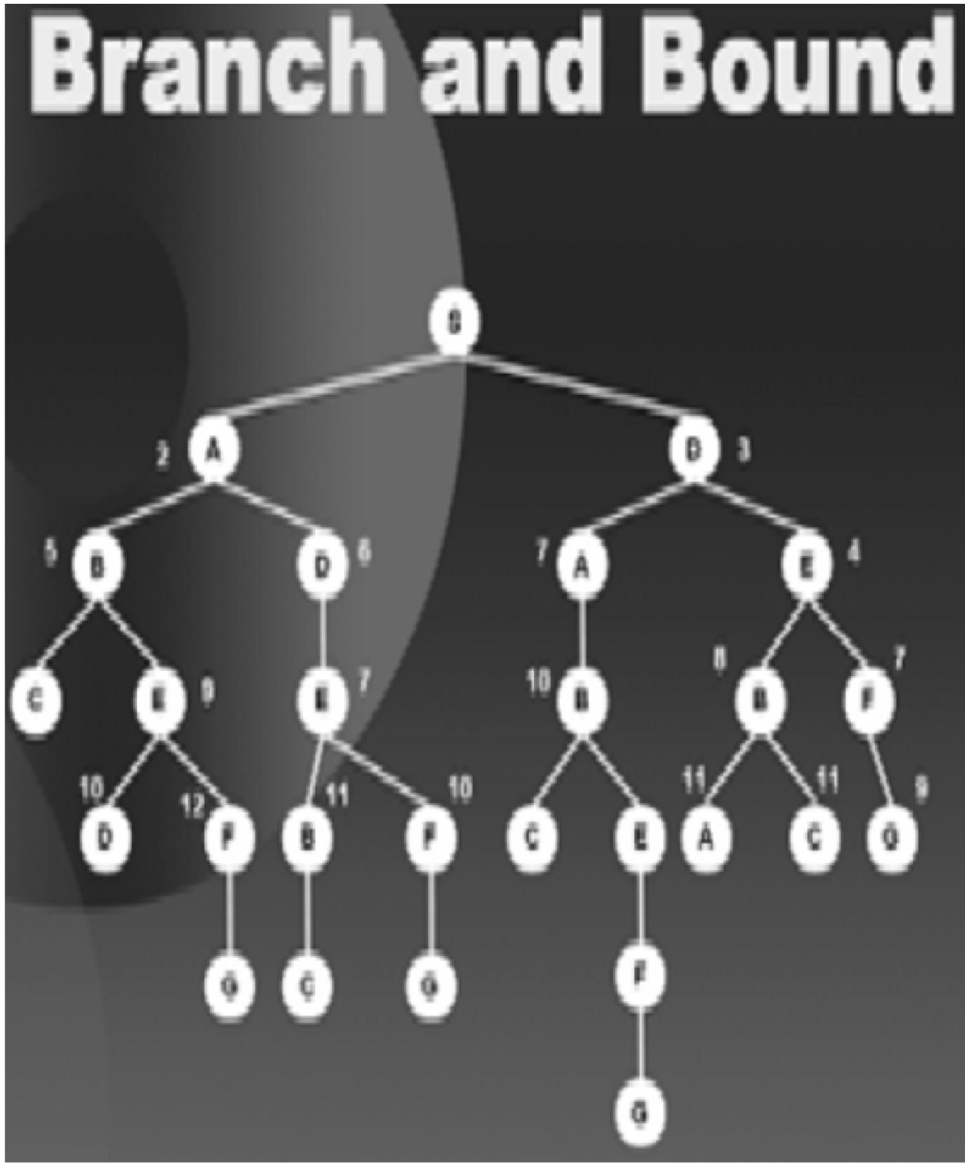


Branch and Bound

- The length of the complete path from S to G is 9.
- Also note that while traveling from S to B we have already covered a distance of 9 units.
- So traveling further from S D A B to some other node will make the path longer.
- So we ignore any further paths ahead of the path S D A B.

Branch and Bound





Branch and Bound

- We proceed in a Best First Search manner.
- Starting at S we see that A is the best option so we explore A.
- From S the options to travel are B and D, the children of A and D the child of S.
- Among these, D the child of S is the best option. So we explore D.
- From here the best option is E so we go there, then B, then D.
- Here we have E, F and A as equally good options so we select arbitrarily and move to say A, then E.
- When we explore E we find out that if we follow this path further, our path length will increase beyond 9 which is the distance of S to G.
- Hence we block all the further sub-trees along this path.

Branch and Bound

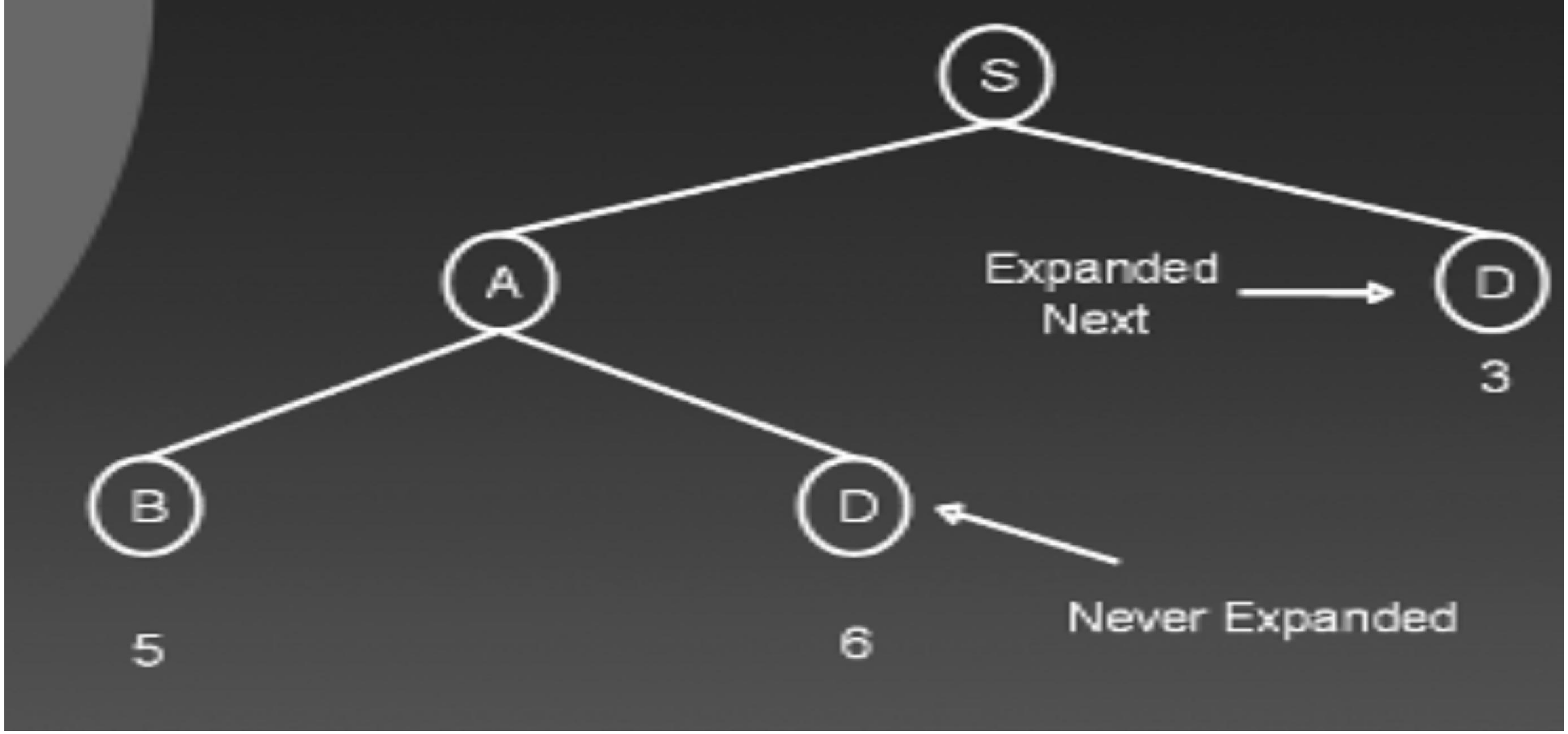
- We then move to F as that is the best option at this point with a value 7, then C, We see that C is a leaf node so we bind C too.
- Then we move to B on the right hand side of the tree and bind the sub trees ahead of B as they also exceed the path length 9.
- We go on proceeding in this fashion, binding the paths that exceed 9 and hence we are saved from traversing a considerable portion of the tree.
- The subsequent diagrams complete the search until it has found all the optimal solution, that is along the right hand branch of the tree.
- Notice that we have saved ourselves from traversing a considerable portion of the tree and still have found the optimal solution.
- The basic idea was to reduce the search space by binding the paths that exceed the path length from S to G.

Improvements in Branch and Bound

- The above procedure can be improved in many different ways.
- We will discuss the two most famous ways to improve it.
  - Estimates
  - Dynamic programming
- The idea of estimates is that we can travel in the solution space using a heuristic estimate.
- By using “guesses” about remaining distance as well as facts about distance already accumulated we will be able to travel in the solution space more efficiently.
- Hence we use the estimates of the remaining distance.

Branch and Bound

- A problem here is that if we go with an overestimate of the remaining distance then we might loose a solution that is somewhere nearby.
- Hence we always travel with underestimates of the remaining distance.
- We will demonstrate this improvement with an example.
- The second improvement is dynamic programming.
- The simple idea behind dynamic programming is that if we can reach a specific node through more than one different path then we shall take the path with the minimum cost.



- In the diagram you can see that we can reach node D directly from S with a cost of 3 and via S A D with a cost of 6 hence we will never expand the path with the larger cost of reaching the same node.
- When we include these two improvements in branch and bound then we name it as a different technique known as A* Procedure.