

Best First Search

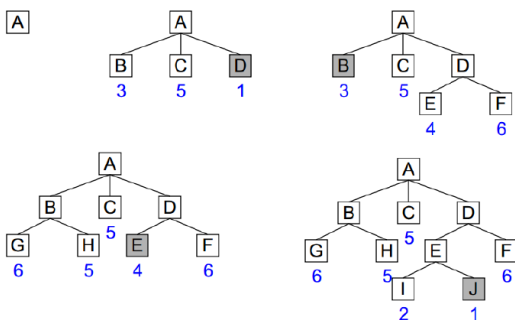
Problem Solving by search

Prof. Deepak C Vegda
email.:deepakvegda.it@ddu.ac.in

Best First Search

- It is a way of combining the advantages of both depth-first search and breadth first search into a single method.
- One way of combining the DFS and BFS is to follow a single path at a time, but switch paths whenever some competing path looks more promising than the current one does.
- At each step of the best-first search process, we select the most promising of the nodes we have generated so far. This is done by applying an appropriate heuristic function to each of them. We then expand the chosen node by using the rules to generate its successors. If one of them is a solution, we can quit. If not, all those new nodes are added to the set of nodes generated so far. Again the most promising node is selected and the process continues.

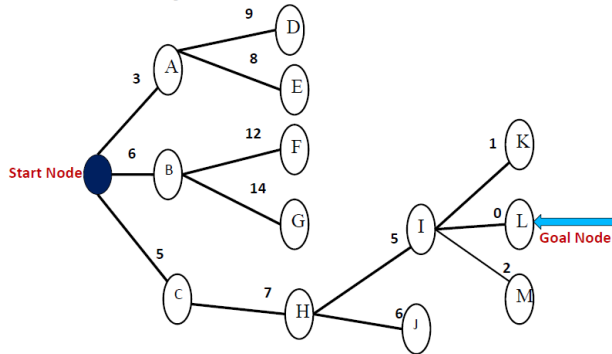
Best First Search Example



Algorithm of Best First Search

1. OPEN = {initial state}.
2. Loop until a goal is found or there are no nodes left in OPEN do:
 - a. Pick the best node in OPEN
 - b. Generate its successors.
 - c. For each successor do:
 - i. If it has not been generated before, evaluate it, add it to OPEN, and record its parent.
 - ii. If it has been generated before, change the parent if this new path is better than the previous one. In that case, update the cost of getting to this node and to any successors that this node may already have.

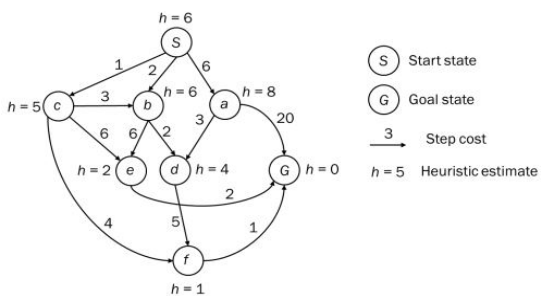
A Sample tree for best first search



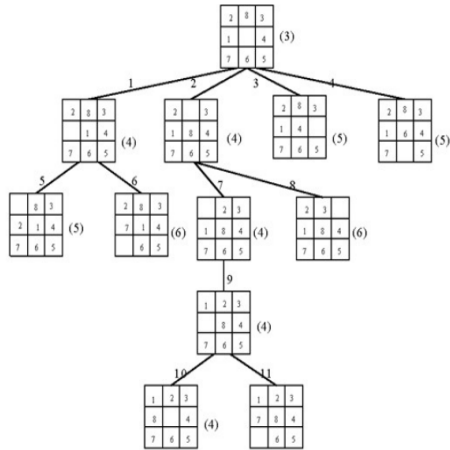
Search process of best first search

Step	Node being expanded	Children	OPEN List	CLOSE List
1	S	(A:3)(B:6)(C:5)	(A:3)(B:6)(C:5)	(A:3)
2	A	(D:9)(E:8)	(B:6)(C:5) (D:9)(E:8)	(C:5)
3	C	(H:7)	(B:6) (D:9) (E:8) (H:7)	(B:6)
4	B	(F:12) (G:14)	(D:9) (E:8) (H:7) (F:12) (G:14)	(H:7)
5	H	(I:5) (J:6)	(D:9) (E:8) (F:12) (G:14) (I:5) (J:6)	(I:5)
6	I	(K:1) (L:0) (M:2)	(D:9) (E:8) (F:12) (G:14) (J:6) (K:1)(L:0)(M:2)	Search stops as goal is reached

How does BFS search for goal state?



Is Best First Search Complete?



Apply BFS and try to solve given puzzle

A*

- A* algorithm was given by Hart, Nilsson and Rafael in 1968.

A* is a best first search algorithm with

$$f(n) = g(n) + h(n)$$

where

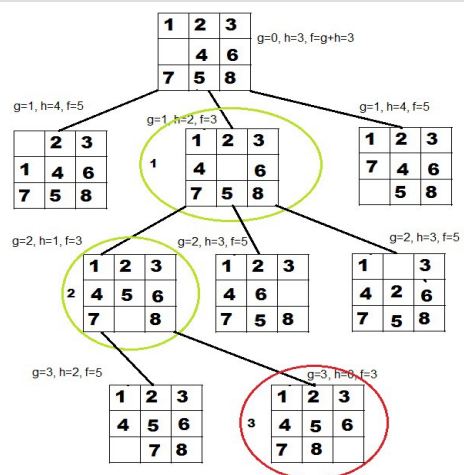
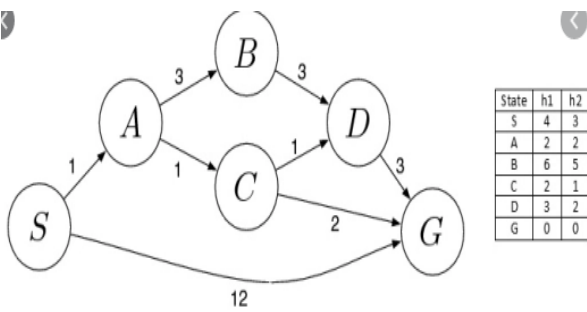
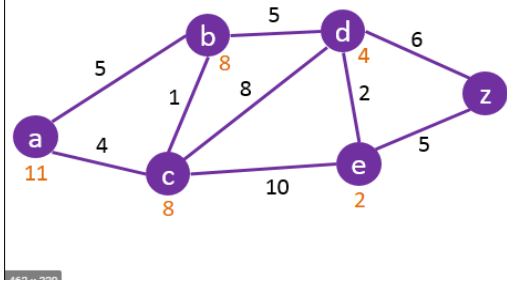
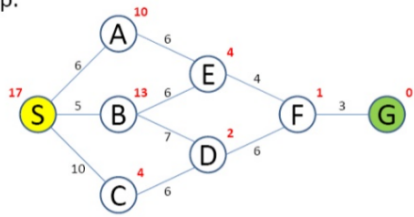
$g(n)$ = sum of edge costs from start to n

$h(n)$ = estimate of lowest cost path from n to goal

$f(n)$ = actual distance so far + estimated distance remaining

1. Initialize: set OPEN= {s}, CLOSED={ }
2. Fail: if OPEN = { }, Terminate & fail.
3. Select: select the minimum cost state, n, from OPEN. Save n in CLOSED.
4. Terminate: if n = G, terminate with success, and return f(n).
5. Expand: for each successor, m, of n
 - If m \notin [open U closed]
 - Set $g(m) = g(n) + C(n, m)$
 - Set $f(m) = g(m) + h(m)$
 - Insert m in OPEN.
 - If m belongs to [open U closed]
 - Set $g(m) = \min\{g(m), g(n) + C(n, m)\}$
 - Set $f(m) = g(m) + h(m)$
 - If f(m) has decreased and m belongs to CLOSED, move m to OPEN
6. Loop: go to step 2

- Perform the A* Algorithm on the following figure. Explicitly write down the queue at each step.



Condition for optimal solution

- Finite path
- For each edge $c(m,n) > \text{some positive value}$
- $h(n) < h(n)^*$

A* admissible

- Terminate for finite graph
- At all time before termination there should be a node N

$$f(N) = F(s)^*$$

If there is a path to goal state, algo will terminate for infinite path also.

A* admissible

- Terminate for finite graph
- At all time before termination there should be a node N

$$f(N) = F(s)^*$$

$$f(N) = g(N) + h(N)$$

$$= g^*(N) + h(N)$$

$$\leq g^*(N) + h^*(N)$$

$$\leq f^*(N) = F^*(s)$$

$$F(a) \quad f(b) \quad f(c) \quad f(d) \quad f(e) \quad f(n) \quad f(g)$$

Merit and demerit of A* Algorithm

Merits

A* is both complete and admissible. Thus A* always finds an optimal path, if one exists.

Demerits

It is costly if the computation cost is high.

Constraint Satisfaction

❖ Many AI problems can be viewed as problems of **constraint satisfaction**.

Examples

- Scheduling
- Timetabling
- Supply Chain Management
- Graph colouring
- Puzzles

Constraint Satisfaction Problem(CSP)

- A CSP consists of
 - A set of variables, X
 - For each variable x_i in X , a domain D_i
 - D_i is a finite set of possible values
- A solution is an assignment of a value in D_i to each variable x_i such that every constraint satisfied.

Crypt-arithmetic puzzle

- We have every letters standing for a digit and every letter stands for a different digit.
- We have to find an assignment of letters to digits such that a given arithmetic formula is correct.
- Variables are D, E, M, N, O, R, S, Y
- Domains are
 - {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} for D, E, N, O, R, Y
 - {1, 2, 3, 4, 5, 6, 7, 8, 9} for S, M

$$\begin{array}{r} \text{SEND} \\ + \text{MORE} \\ \hline \text{MONEY} \end{array}$$

Constraints for this problem

Constraint 1:

We can write one long constraint for the sum.

$$\begin{aligned} &1000 * S + 100 * E + 10 * N + D \\ &+ 1000 * M + 100 * O + 10 * R + E \\ &= 10000 * M + 1000 * O + 100 * N + 10 * E + Y \end{aligned}$$

Constraint 2:

alldifferent(D, E, M, N, O, R, S, Y)

These two constraints express the problem precisely.

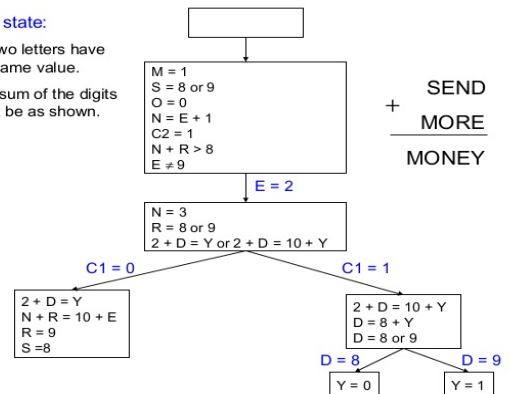
$$\begin{array}{r} \text{S E N D} \\ + \\ \text{M O R E} \\ \hline \end{array}$$

M O N E Y

0 1 2 3 4 5 6 7 9

Initial state:

- No two letters have the same value.
- The sum of the digits must be as shown.



Solution

- Rules for propagating constraints generates the following constraints:

- $M = 1$, since two single-digit numbers plus a carry can not total more than 19.
- $S = 8$ or 9 , since $S + M + C3 > 9$ (to generate the carry) and $M = 1$, $S + 1 + C3 > 9$, so $S + C3 > 8$ and $C3$ is at most 1.
- $O = 0$, since $S + M(1) + C3(<=1)$ must be at least 10 to generate a carry and it can be most 11. But M is already 1, so O must be 0.
- $N = E$ or $N = E + 1$, depending on the value of $C2$. But N cannot have the same value as E . So $N = E + 1$ and $C2$ is 1.
- In order for $C2$ to be 1, the sum of $N + R + C1$ must be greater than 9, so $N + R$ must be greater than 8.
- $N + R$ cannot be greater than 18, even with a carry in so E cannot be 9.


Solution...

- Suppose E is assigned the value 2.
- The constraint propagator now observes that:
- $N = 3$ since $N = E + 1$.
- $R = 8$ or 9 , since $R + N(3) + C1(1 \text{ or } 0) = 2$ or 12 . But since N is already 3, the sum of these nonnegative numbers cannot be less than 3. Thus $R + 3 + (0 \text{ or } 1) = 12$ and $R = 8$ or 9 .
- $2 + D = Y$ or $2 + D = 10 + Y$, from the sum in rithmost column.

Try These

- BASE + BALL = GAMES
- YOUR + YOU = HEART
- EAT + THAT = APPLE
- TWO + TWO = FOUR
- CROSS + ROADS = DANGER

AND or OR

- The complex problem and the sub problem, there exist two kinds of relationships.
 - AND relationship
 - OR relationship.
- In AND relationship, the solution for the problem is obtained by solving all the sub problems.
- In OR relationship, the solution for the problem is obtained by solving any of the sub problem.
- An arc () connecting different branches is called AND.

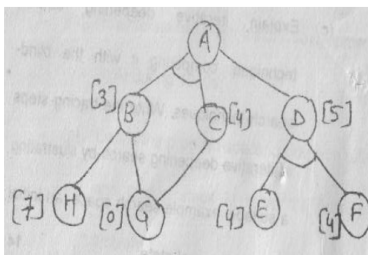
AND/OR graphs

- Real life situations do not exactly decompose into either AND tree or OR tree but are always a combination of both.
- AND/OR graph is useful for representing the solutions of problem that can be solve by decomposing them into a set of smaller problems.
- A* algorithm is not adequate for AND/OR graphs.
- AO* algorithm is used for AND/OR graphs.

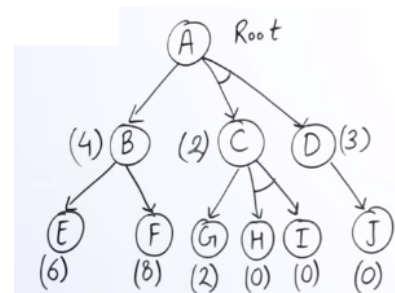
AO* Algorithm

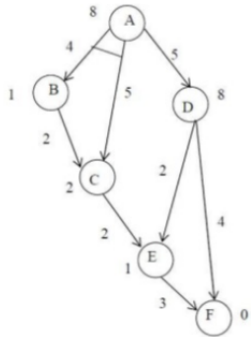
1. Initialize: set $G^* = \{s\}$, $f(s)=h(s)$
if $s \in T$, label s as SOLVED
2. Terminate: if s is SOLVED, then terminate.
3. Select: select a non-terminal leaf node n from the marked sub tree.
4. Expand: make explicit the successors of n
For each new successor, m :
set $f(m)=h(m)$
if m is terminal, label m SOLVED
5. Cost revision: call cost-revise(n)
6. Loop: Go to step 2

Example



Illustrate the operation of AO* search upon the following





CROSS
ROADS

DANGER