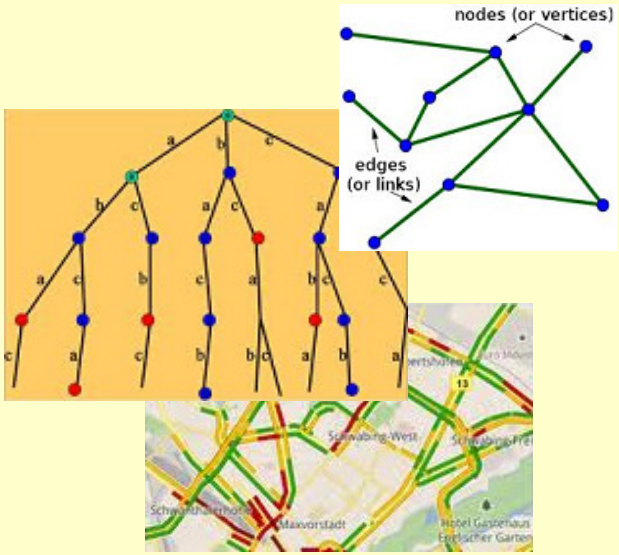


State Search & Algorithms

Ali Ridho Barakbah

Intelligent Knowledge Creation (iKnow) Research Group
Department of Information and Computer Engineering
Politeknik Elektronika Negeri Surabaya

Description

- It is an algorithm for finding potential solutions.
- It is frequently encountered by researchers in the field of AI.

Defining Problem

- Defining a state (state space)
- Specifying one or more initial states
- Specifying one or more goal states
- Establishing rules (a set of rules)

A case...

A farmer wants to transport himself, a wolf, a fat goose, and a bundle of rice across a river. Unfortunately, his boat is very small; he can only carry one item per trip. Furthermore, he cannot leave the wolf and the goose together, because the wolf would eat the goose. Likewise, he cannot leave the goose and the rice together.

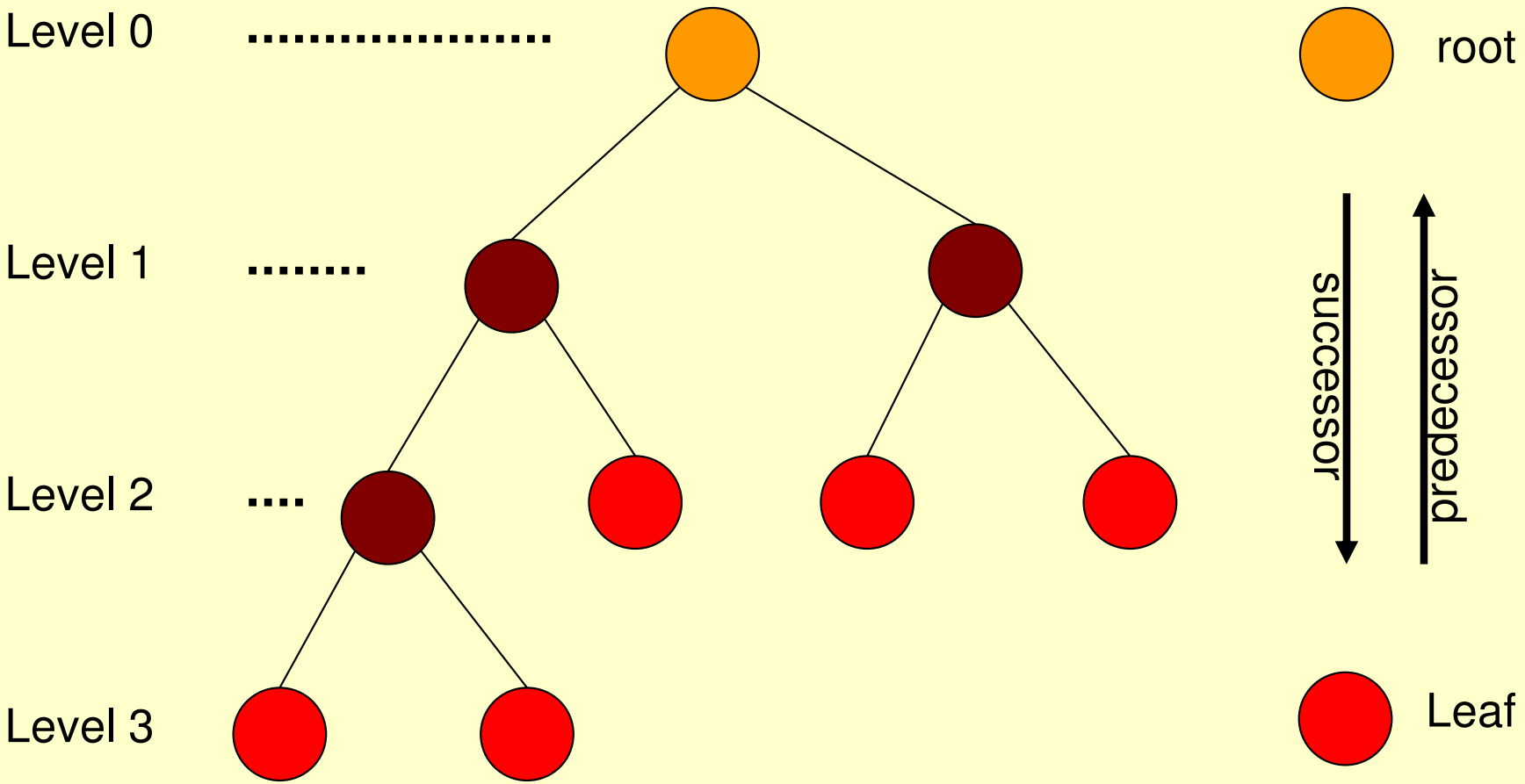
State Space

- State → (Wolf, Goose, Rice, Farmer)
- The origin place, when there are only wolves and rice, can be represented by the state (1, 0, 1, 0), while the destination area is (0, 1, 0, 1)

Initial & Destination State

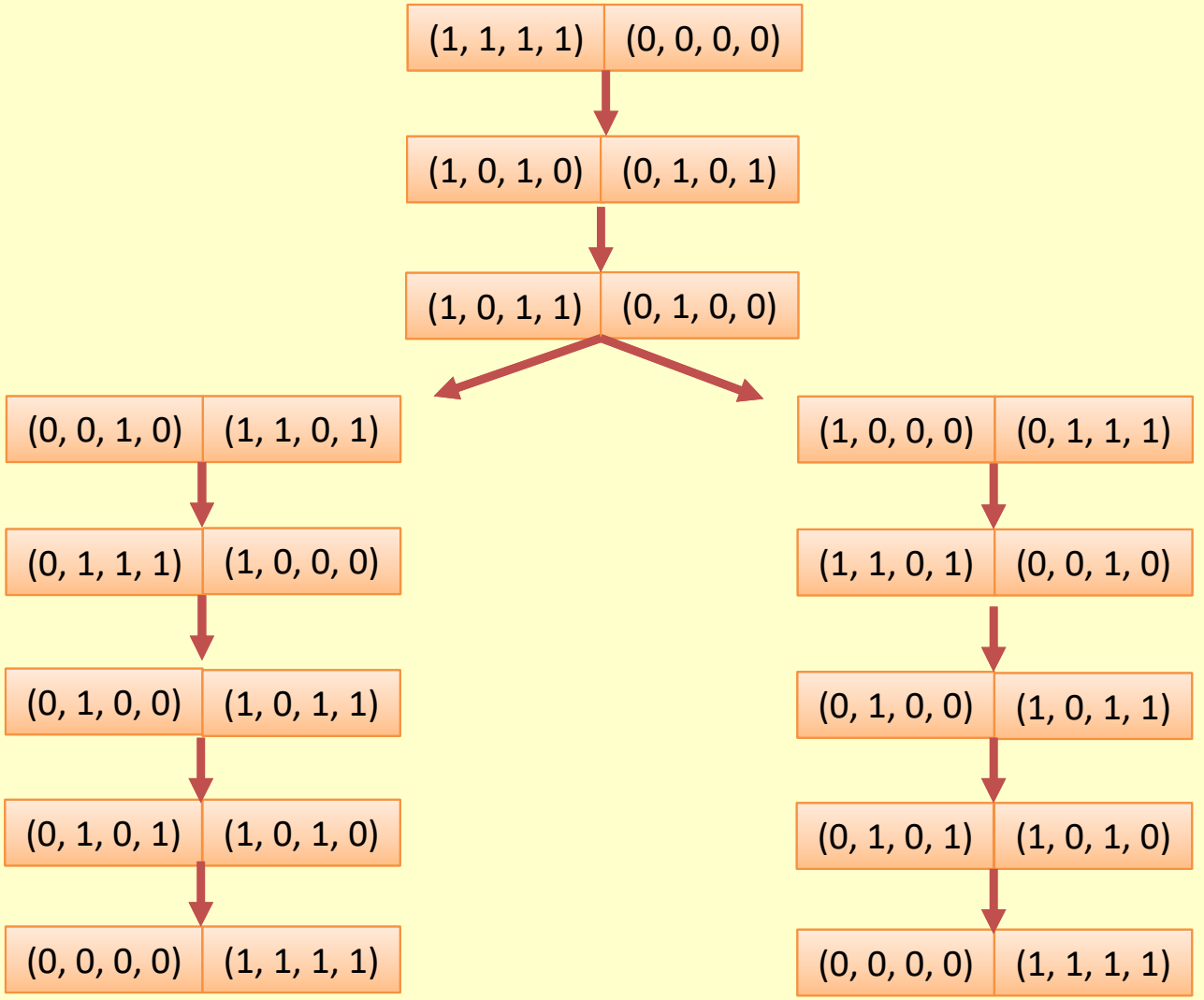
- Initial State
  - Origin → (1, 1, 1, 1)
  - Destination → (0, 0, 0, 0)
- Final State
  - Origin → (0, 0, 0, 0)
  - Destination → (1, 1, 1, 1)

Hierarchical Representation

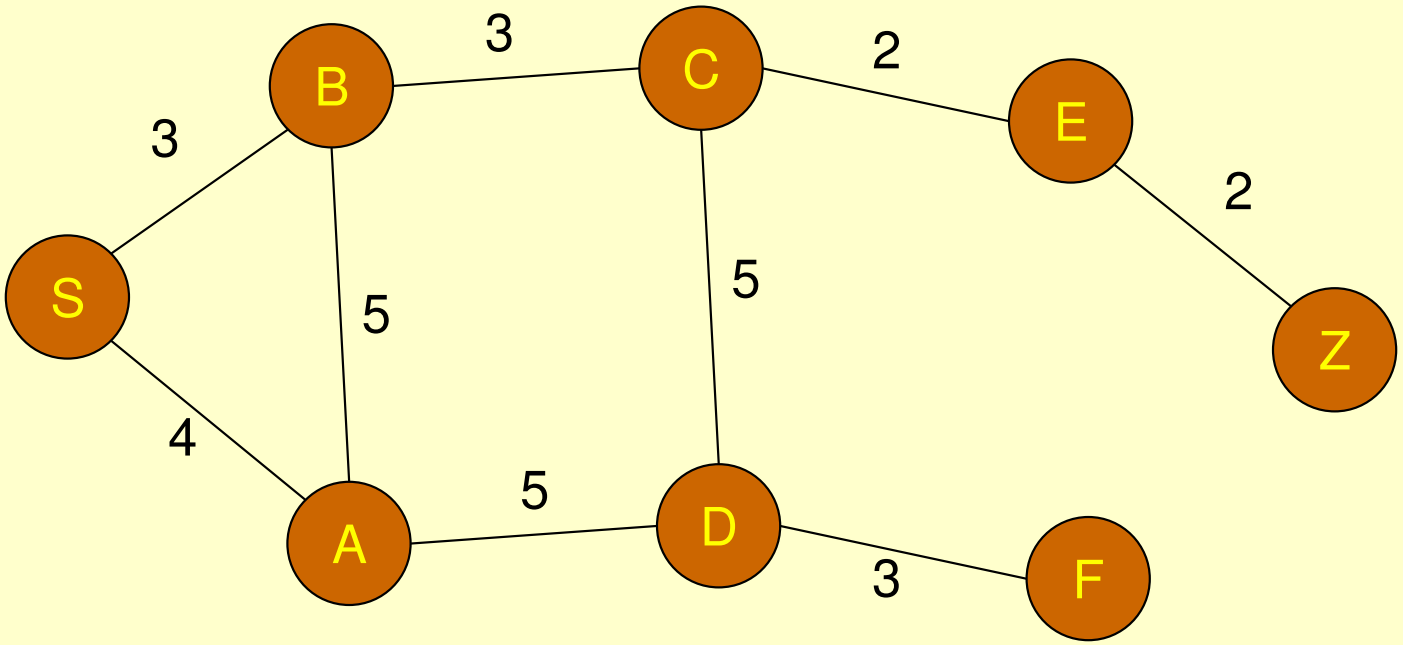


State Representation

Origin	Destination
(Wolf, Goose, Rice, Farmer)	(Wolf, Goose, Rice, Farmer)

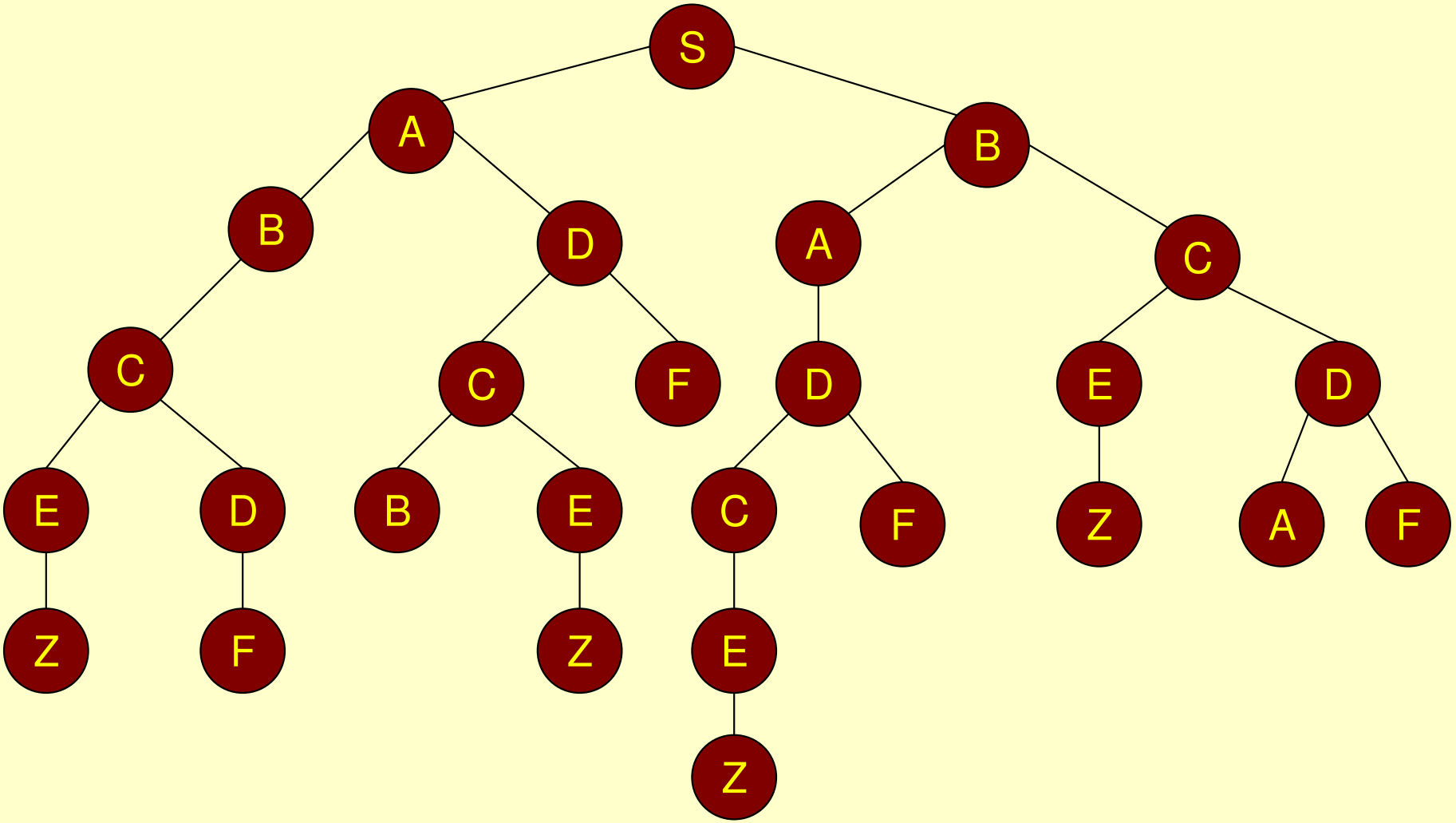


Another case...

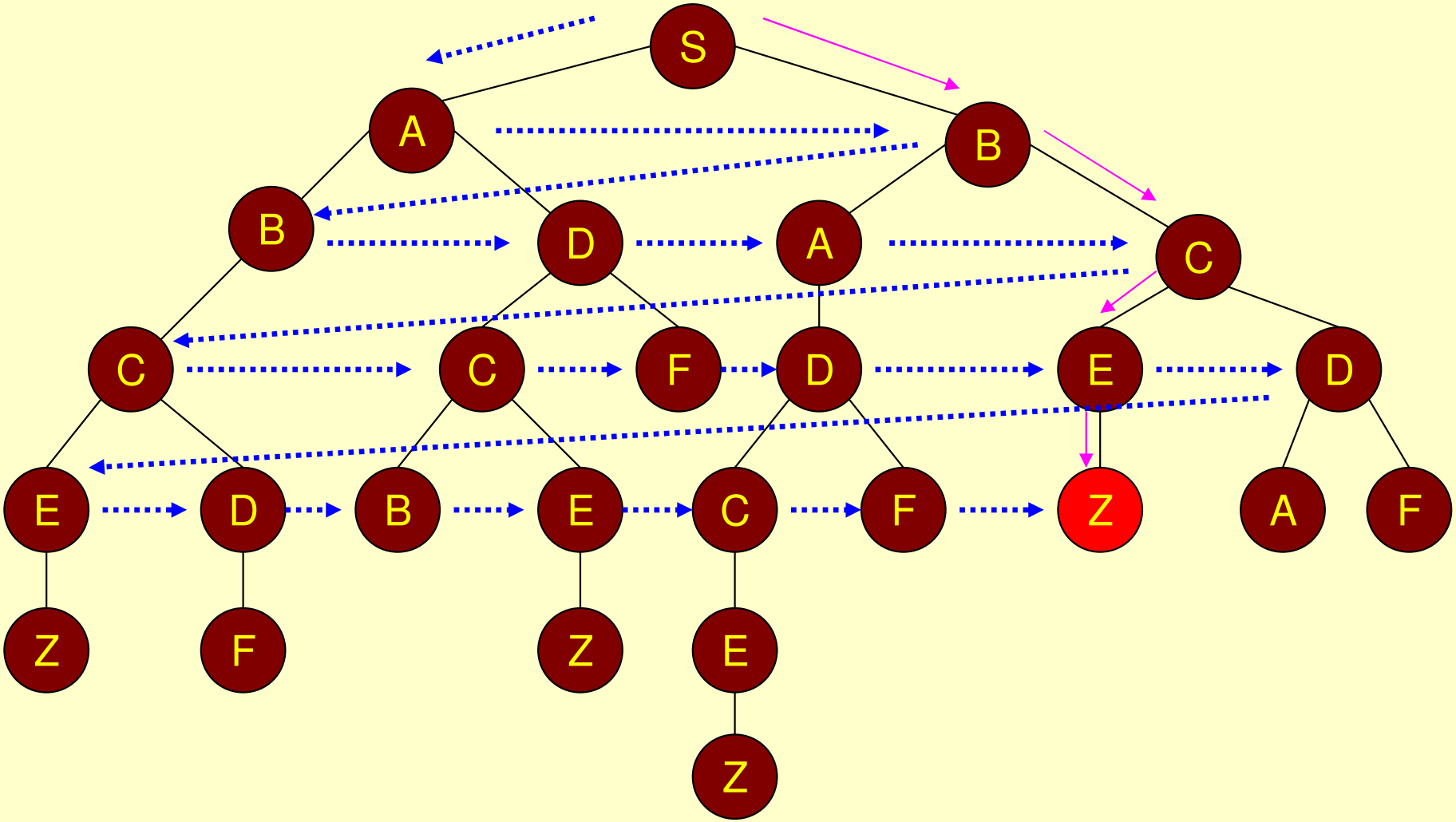


A graph containing a Hamiltonian circuit

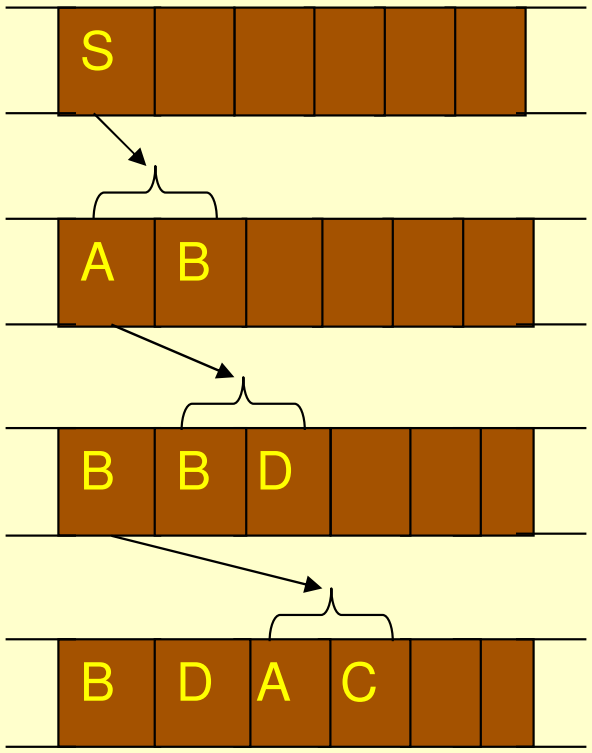
Hierarchical Representation of the Problem



Breadth First Search



Algorithm

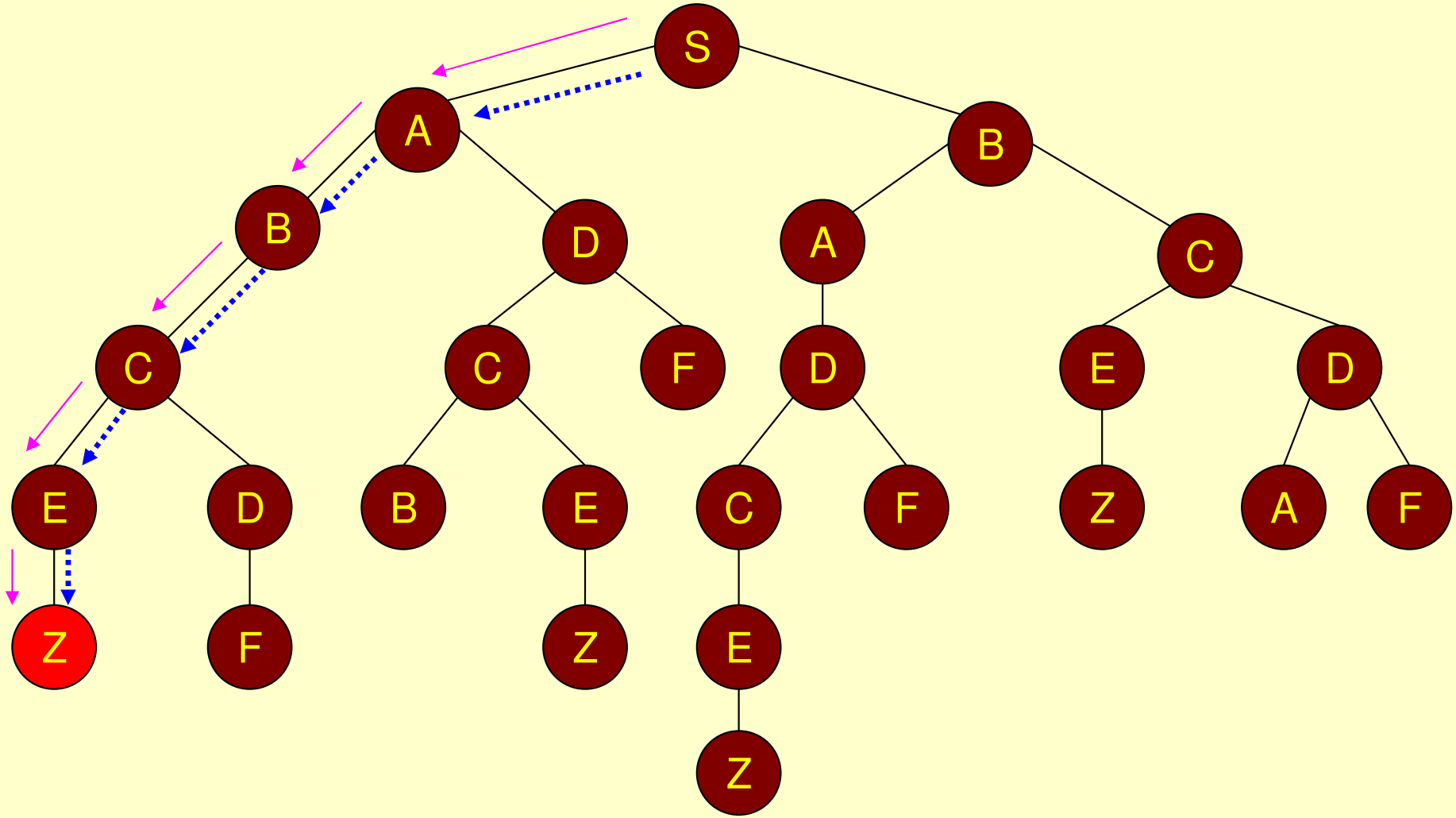


dan seterusnya …

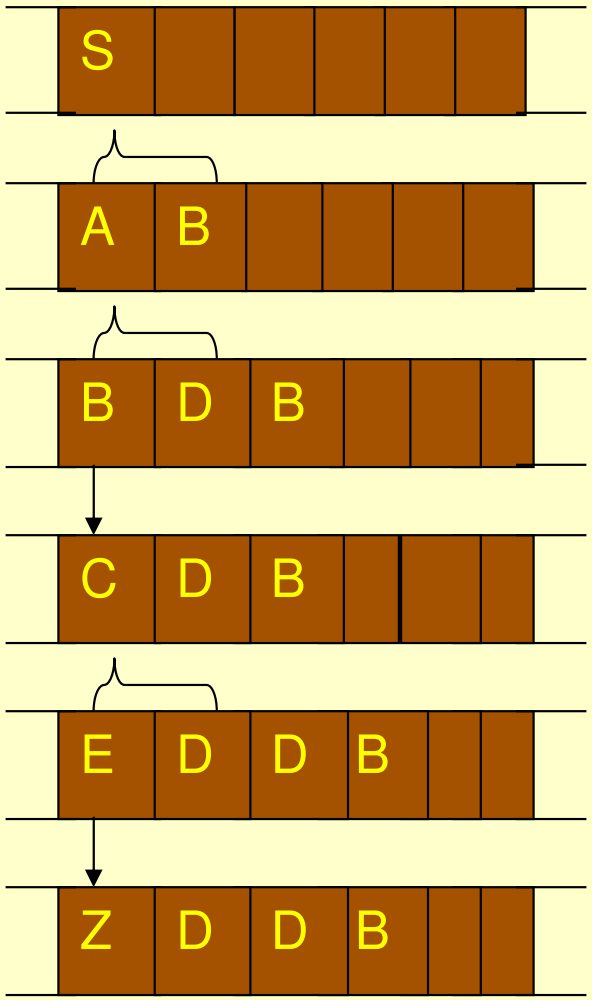
Analysis

- Kelebihan
  - Tidak akan menemui jalan buntu
  - Jika ada satu solusi, pasti diketemukan
- Kelemahan
  - Boros memori
  - Mungkin terjebak pada local optima

Depth First Search



Algorithm



Analysis

- Advantages
  - Requires relatively little memory
  - Finds a solution without needing extensive further testing
- Disadvantages
  - May get stuck in local optima

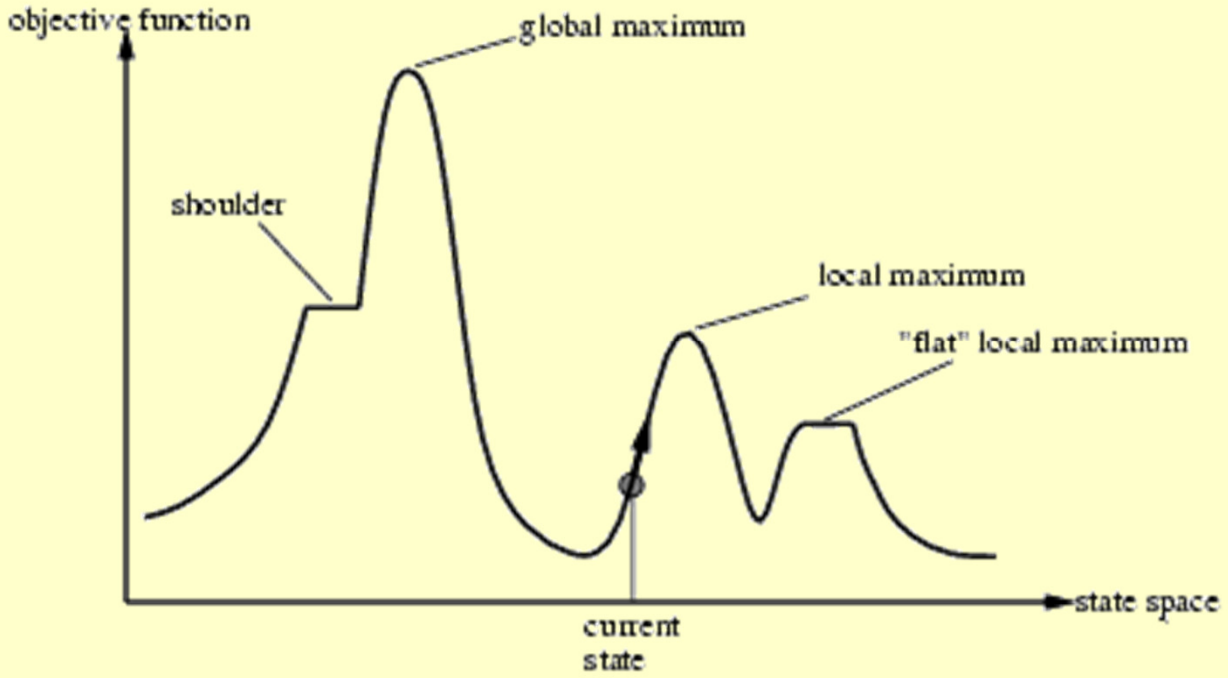
Hill Climbing Search

- "Like climbing Everest in thick fog with amnesia"

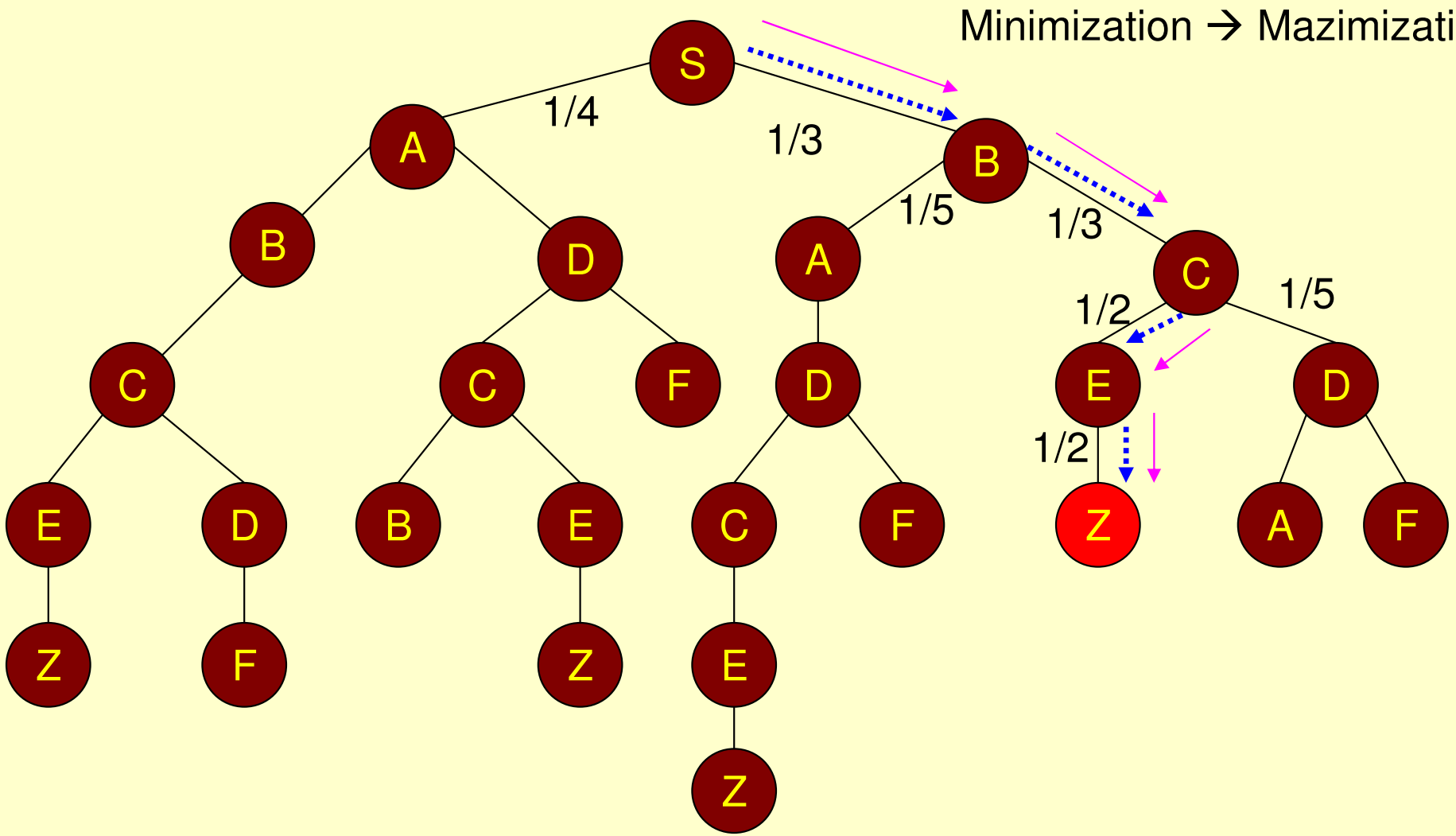
```
function HILL-CLIMBING( problem) returns a state that is a local maximum
    inputs: problem, a problem
    local variables: current, a node
                     neighbor, a node

    current ← MAKE-NODE(INITIAL-STATE[problem])
    loop do
        neighbor ← a highest-valued successor of current
        if VALUE[neighbor] ≤ VALUE[current] then return STATE[current]
        current ← neighbor
```

Hill-climbing search

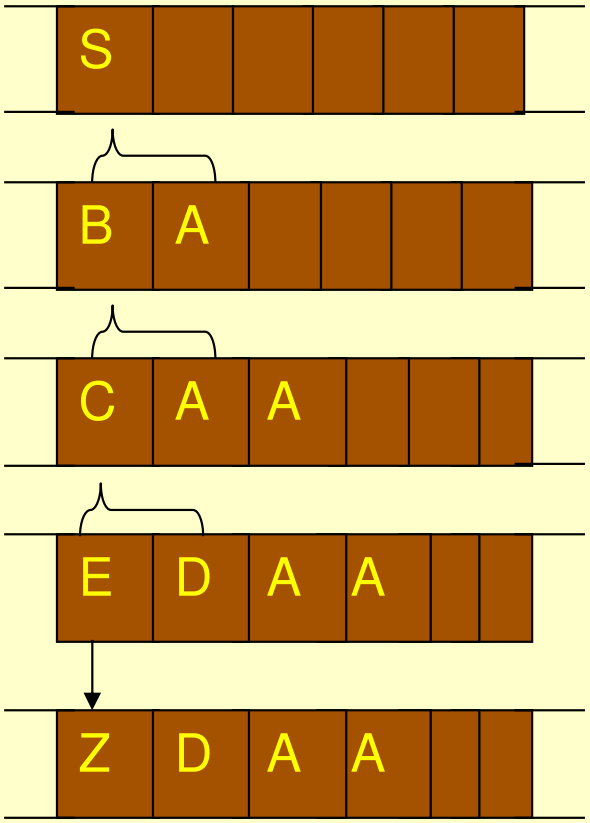
- Problem: depending on initial state, can get stuck in local maxima



Hill climbing



Algorithm



Analysis

- Advantages
  - Requires little memory
  - Finds a solution without needing extensive further testing
- Disadvantages
  - May get stuck in local optima

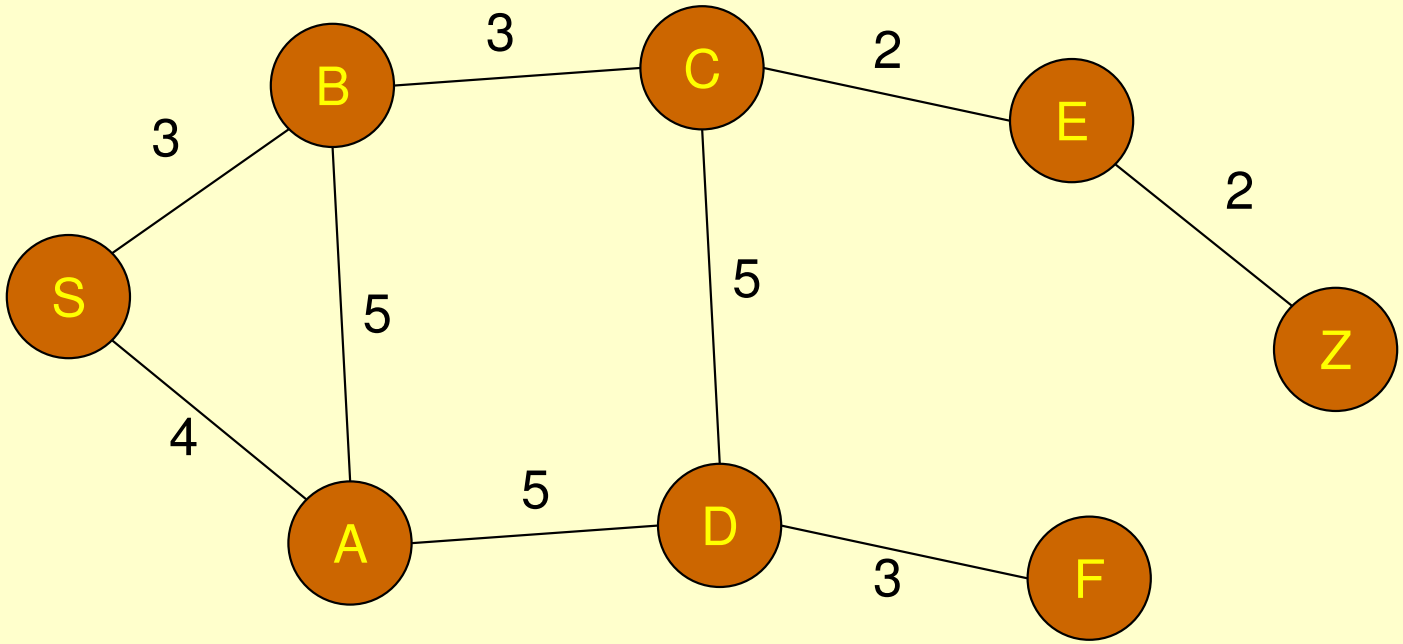
Best First Search

Termasuk Greedy Search

Heuristic Function

f(n) = h(n)

where h(n) = estimated cost to goal from n



Perkiraan Jarak

Node	Perkiraan Jarak ke Z
S	10
A	8
B	6
C	3,7
D	4,5
E	2
F	2

Heuristic

Webster's Revised Unabridged Dictionary (1913) (web1913)

Heuristic \Heu*ris"tic\, a. [Greek. to discover.] Serving to discover or find out.

The Free On-line Dictionary of Computing (15Feb98)

heuristic 1. <programming> A rule of thumb, simplification or educated guess that reduces or limits the search for solutions in domains that are difficult and poorly understood. Unlike algorithms, heuristics do not guarantee feasible solutions and are often used with no theoretical guarantee. 2. <algorithm> approximation algorithm.

From WordNet (r) 1.6

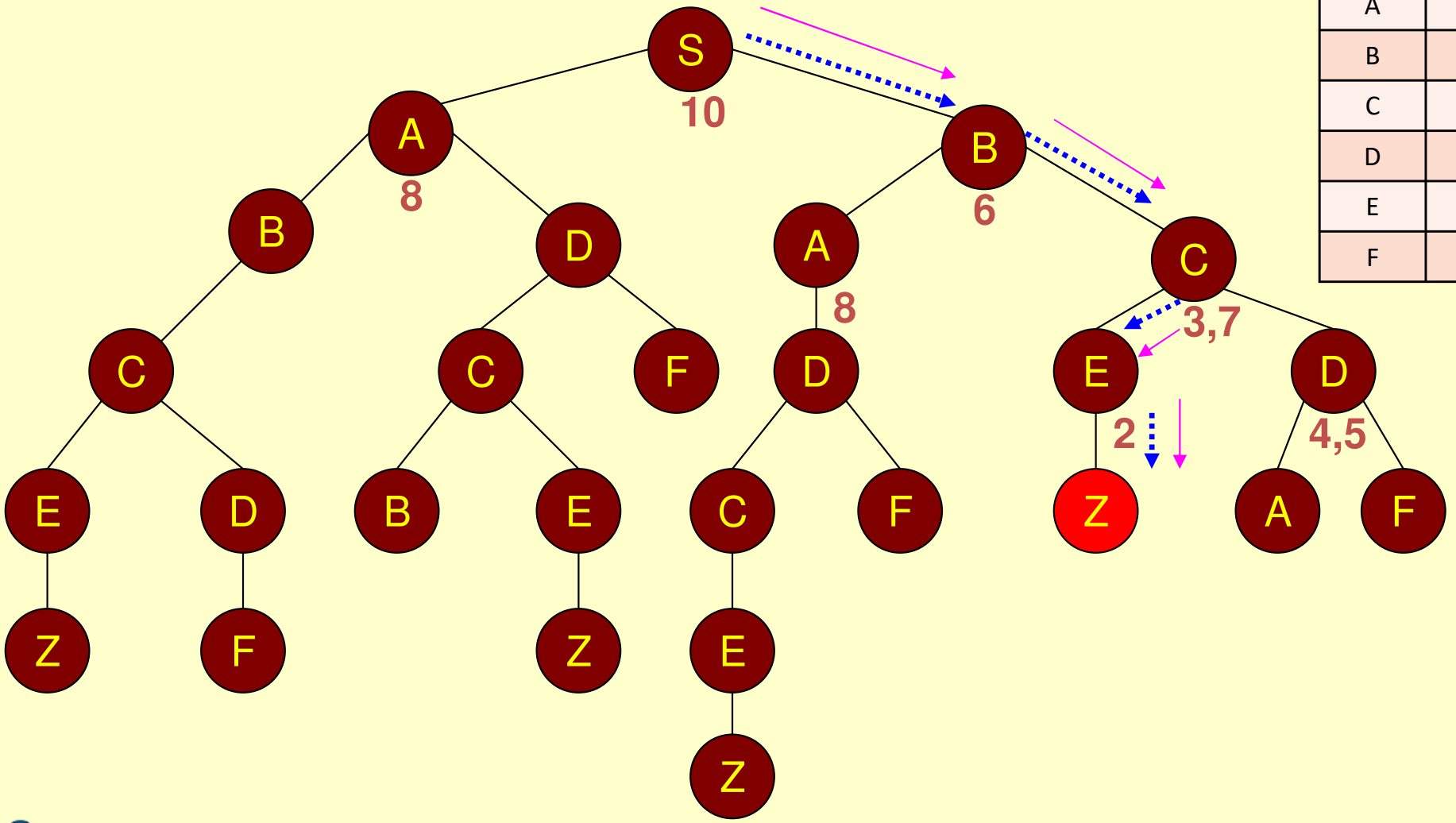
heuristic adj 1: (computer science) relating to or using a heuristic rule 2: of or relating to a general formulation that serves to guide investigation [ant: algorithmic] n : a commonsense rule (or set of rules) intended to increase the probability of solving some problem [syn: heuristic rule, heuristic program]

Heuristic Search

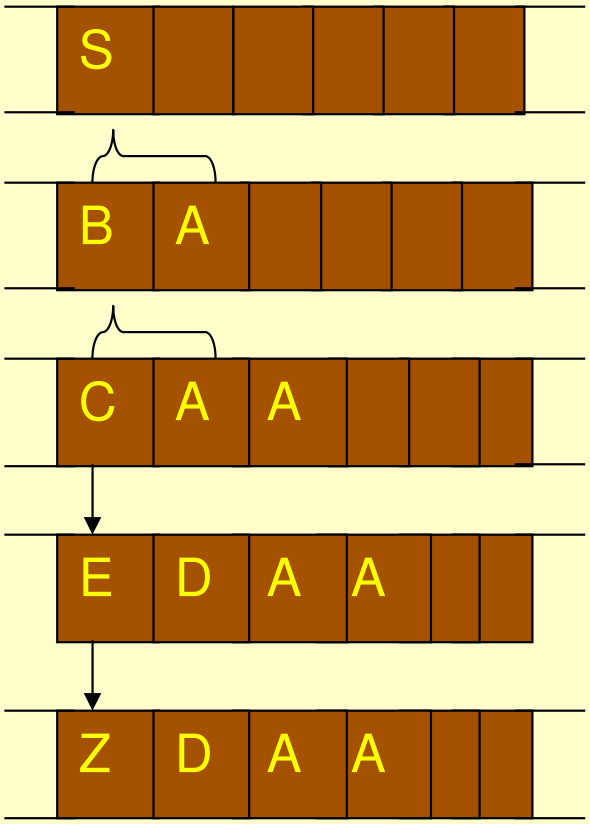
- Define a heuristic function, **h(n)**, that estimates the "goodness" of a node n.
- Specifically, h(n) = **estimated cost** (or distance) of minimal cost path from n **to a goal state**.
- The heuristic function is an estimate, based on domain-specific information that is computable from the current state description, of how close we are to a goal

Best First Search

Node	Distance Estimation to Z
S	10
A	8
B	6
C	3,7
D	4,5
E	2
F	2



Algorithm



Analysis

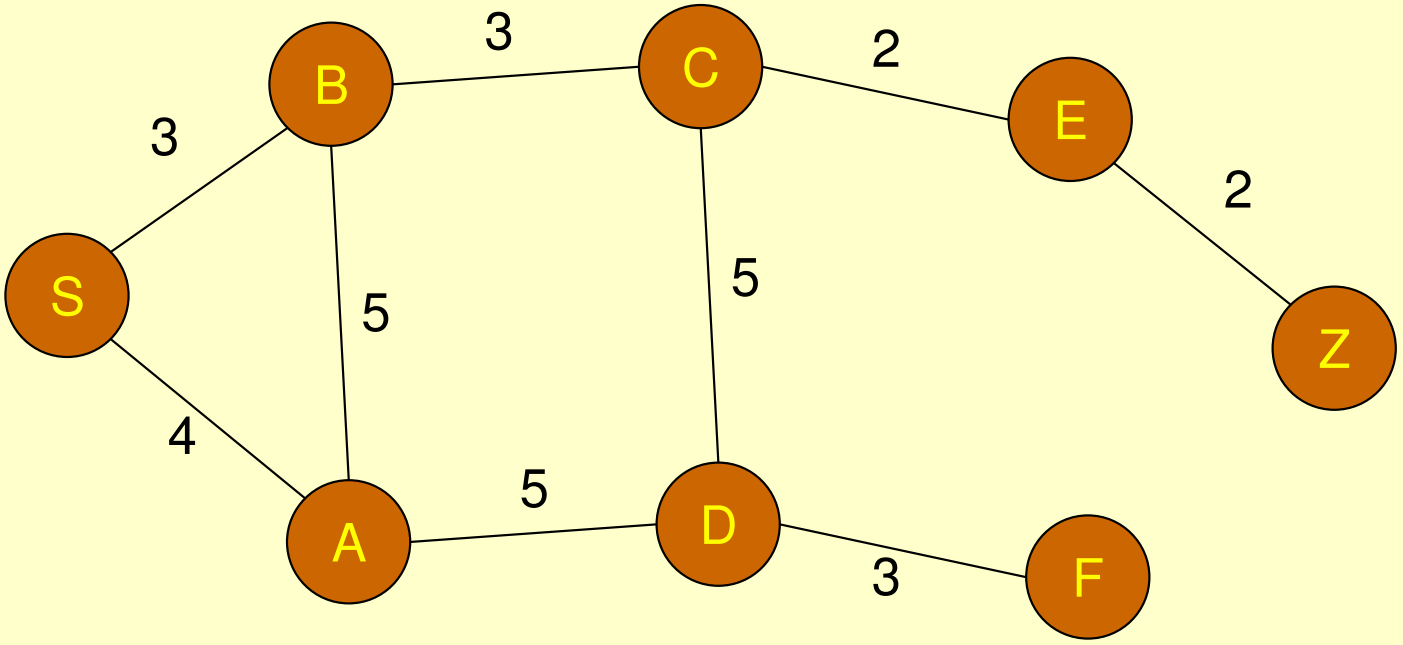
- Advantages
  - Requires little memory
  - Finds a solution without needing extensive further testing
- Disadvantages
  - May get stuck in local optima

A* Search

including Informed Search

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

where
- $g(n)$ = cost so far to reach n
- $h(n)$ = estimated cost to goal from n



Perkiraan Jarak

Node	Distance Estimation to Z
S	10
A	8
B	6
C	3,7
D	4,5
E	2
F	2

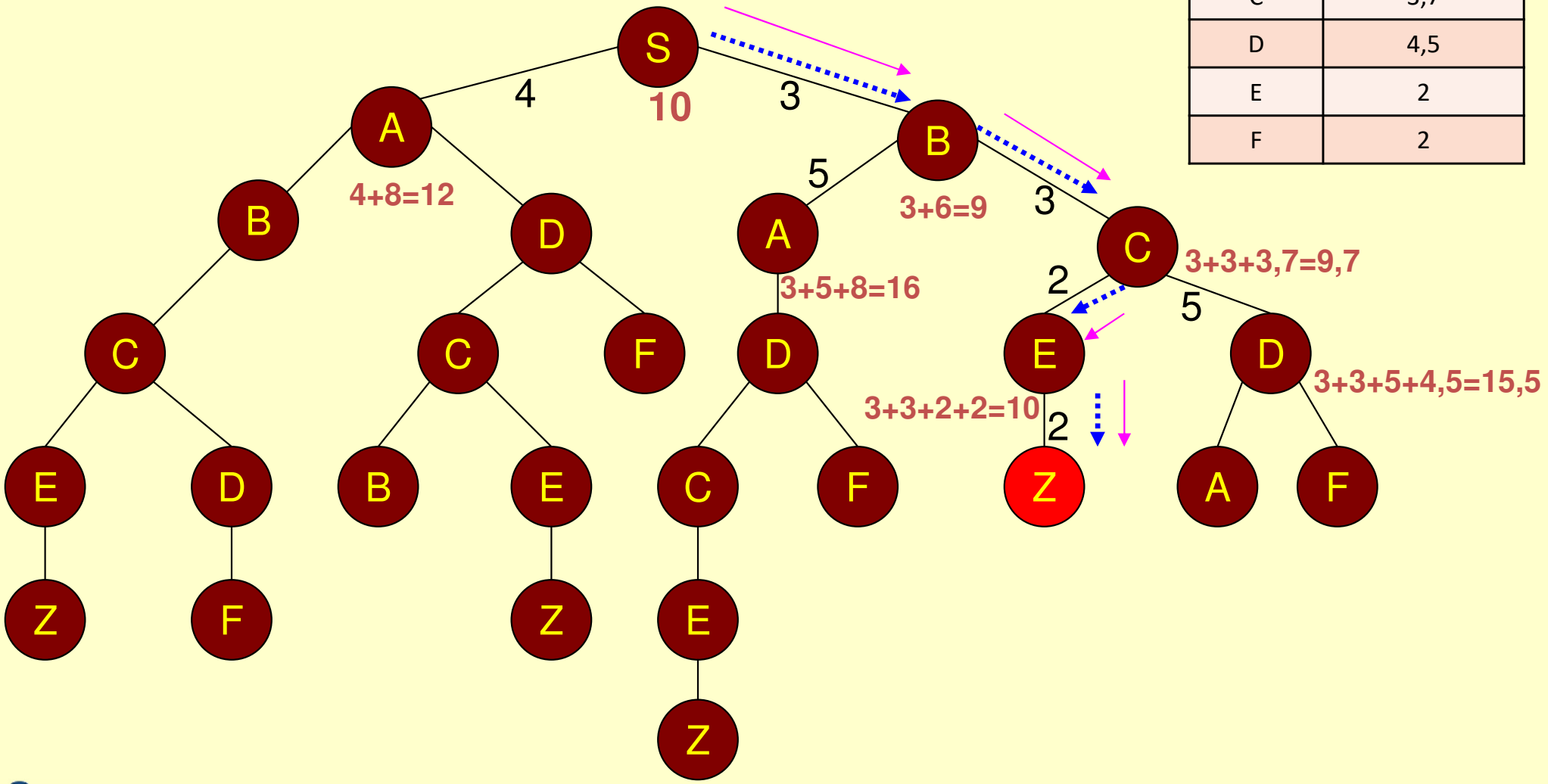
Informed Search

- Add domain-specific information to select the best path along which to continue searching
- We kept looking at nodes closer and closer to the goal, but were accumulating costs as we got further from the initial state
- Our goal is not to minimize the distance from the current head of our path to the goal, we want to minimize the *overall* length of the path to the goal!
- Let g(N) be the cost of the best path found so far between the initial node and N

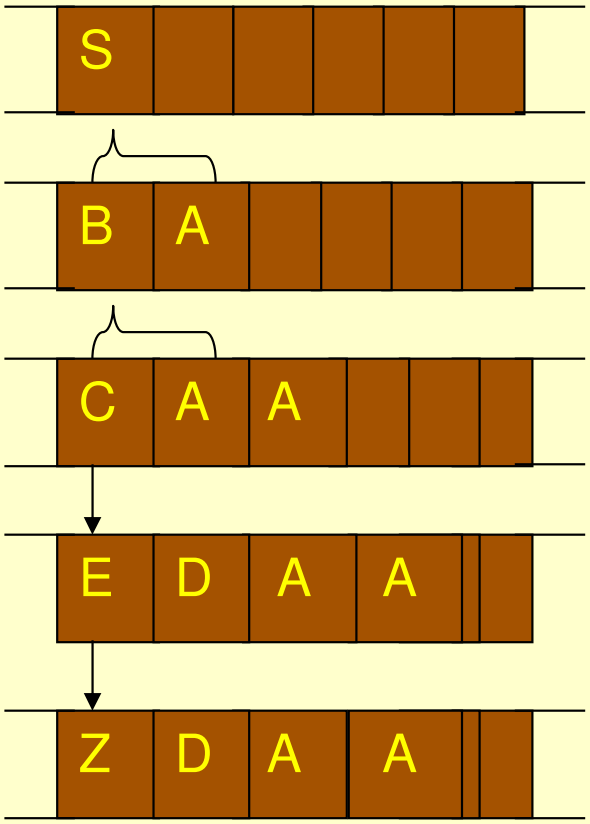
- $f(N) = g(N) + h(N)$

A* Search

Node	Distance Estimation to Z
S	10
A	8
B	6
C	3,7
D	4,5
E	2
F	2



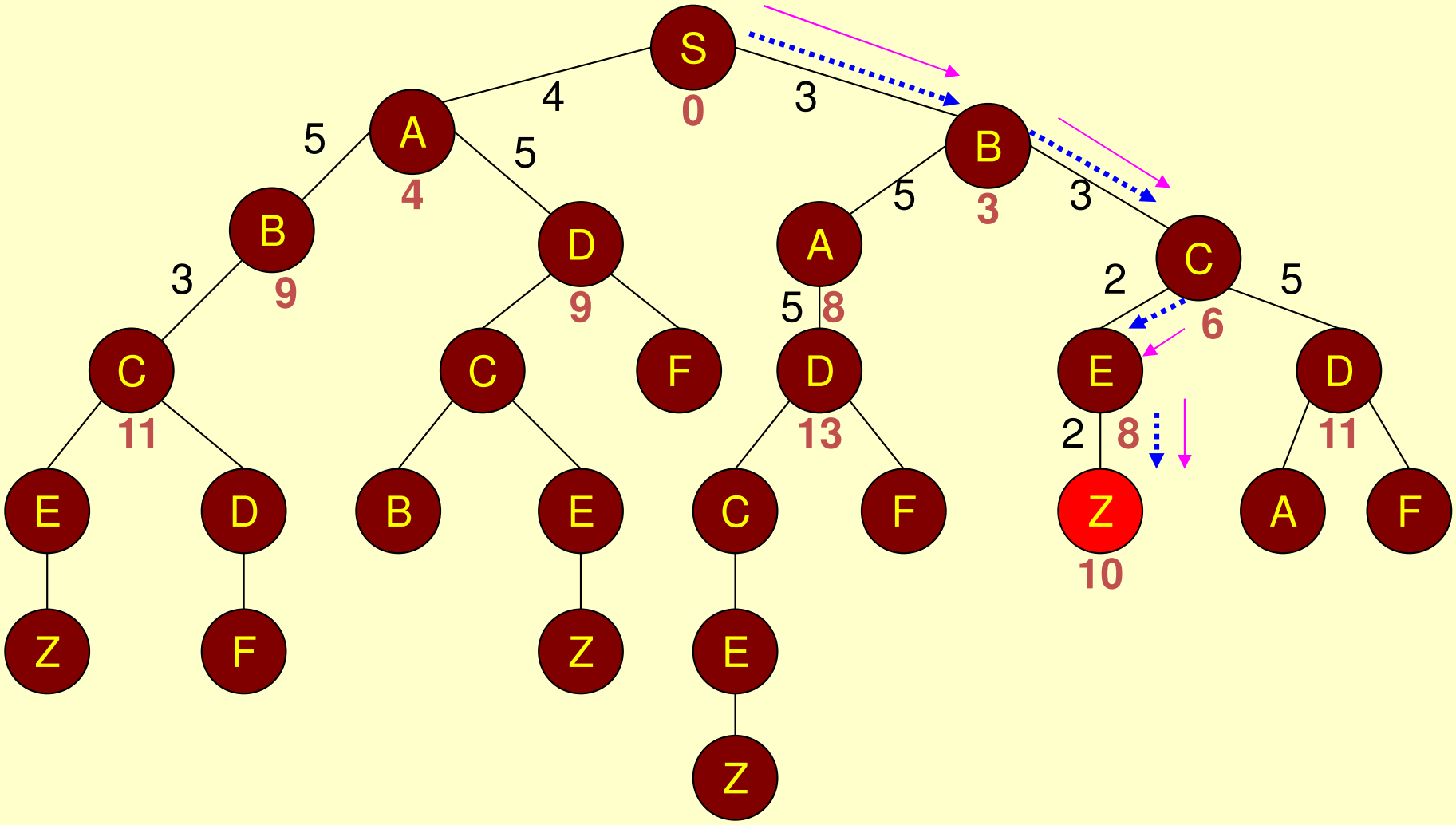
Algorithm



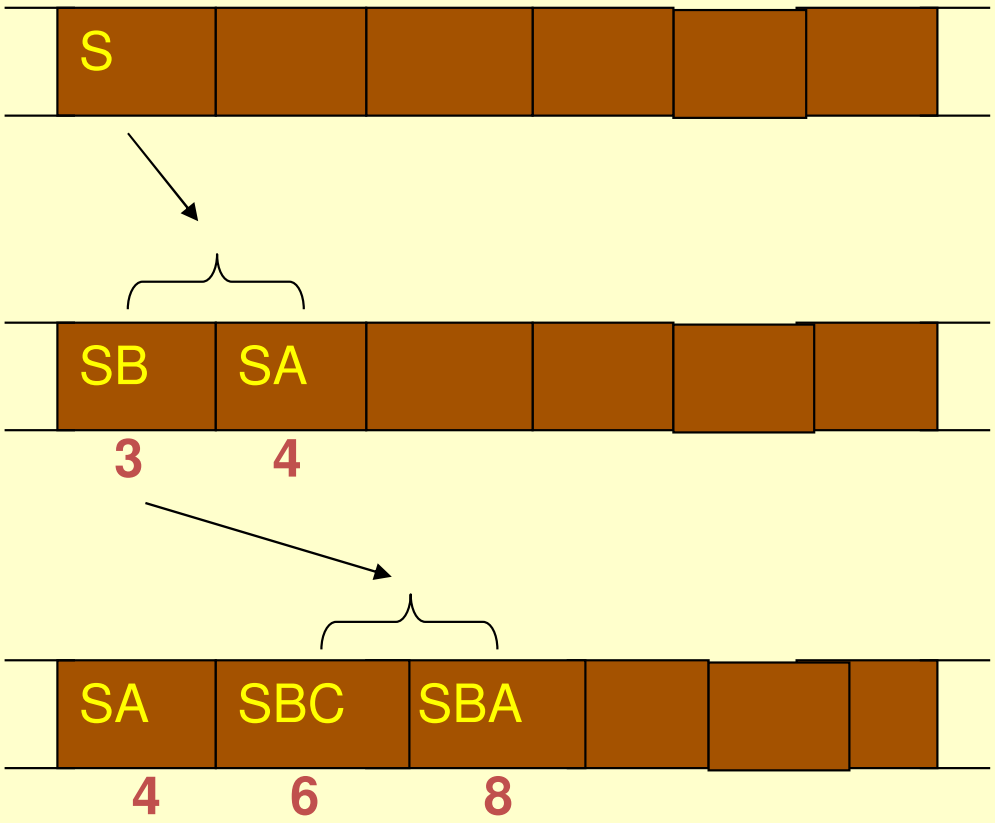
Analysis

- Advantages
  - Better than Best-First Search because it incorporates actual cost
  - Requires little memory
  - Finds a solution without needing to perform further testing
- Disadvantages
  - May get stuck in local optima

Branch and Bound



Algorithm

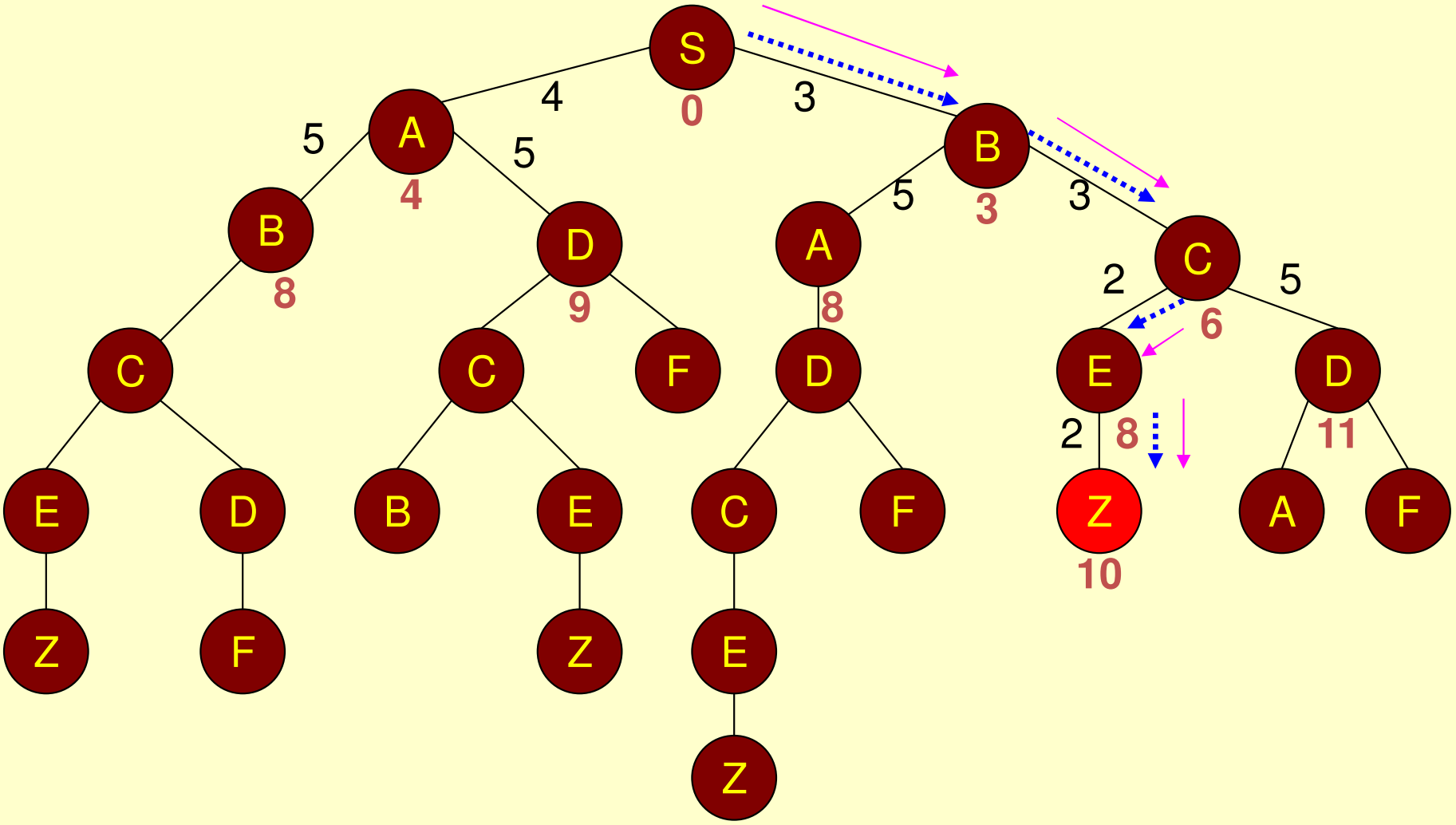


dan seterusnya…

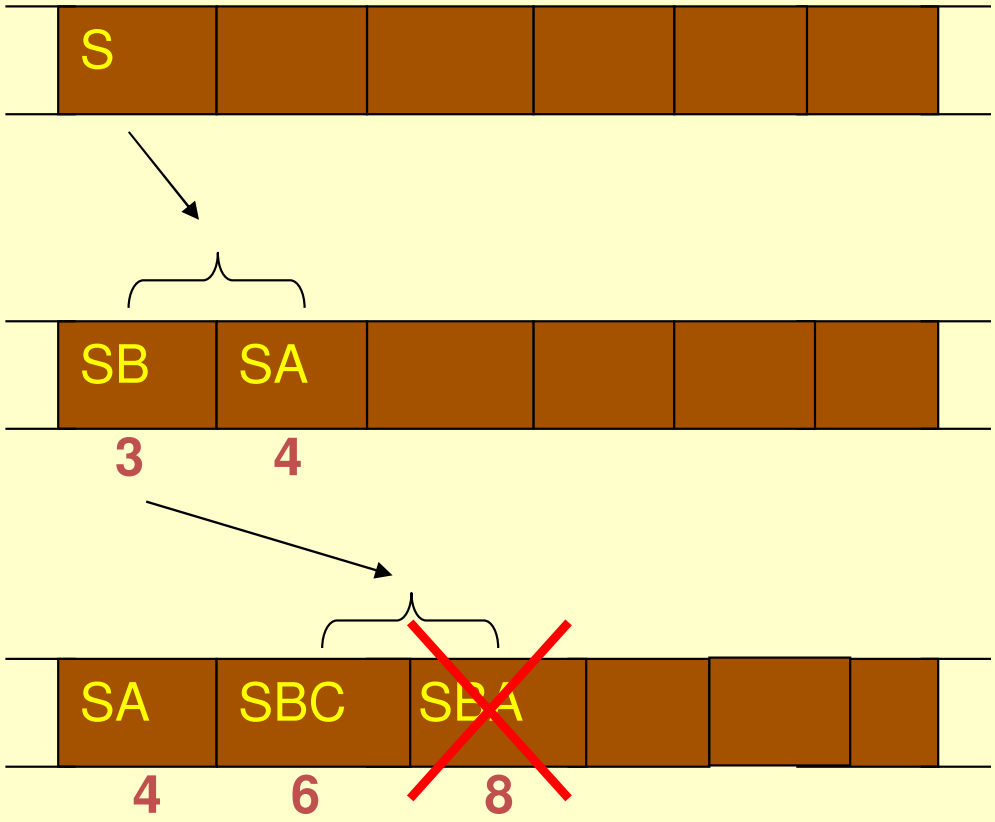
Analysis

- Advantages
  - Always finds the global optimum
- Disadvantages
  - Memory-intensive due to storing partial paths more than once

Dynamic Programming



Algorithm



dan seterusnya…

Analysis

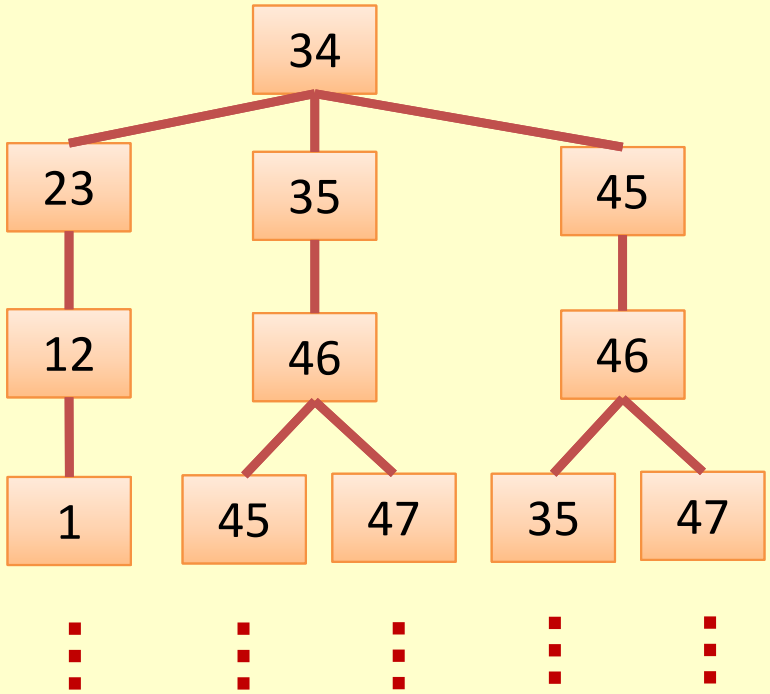
- Advantages
  - Always finds the global optimum
  - Faster and more memory-efficient because it stores the partial path only once
- Disadvantages
  - Must remember the last node of the previously reached partial path

Applications

- Games
- Robot Navigation
- Graph based problems
- Travelling Salesman Problem
- etc

Robot Navigation

1	2	3	4	5	6	7	8	9	10	11
12	13	14	15	16	17	18	19	20	21	22
23	24	25	26	27	28	29	30	31	32	33
34	35	36	37	38	39	40	41	42	43	44
45	46	47	48	49	50	51	52	53	54	55



Robot Navigation

f(N) = h(N), with h(N) = Manhattan distance to the goal

8	7	6	5	4	3	2	3	4	5	6
7		5	4	3						5
6			3	2	1	0	1	2		4
7	6									5
8	7	6	5	4	3	2	3	4	5	6

Robot Navigation

Best First Search - Heuristic Search

8	7	6	5	4	3	2	3	4	5	6
7		5	4	3						5
6			3	2	1	0	1	2		4
7	6									5
8	7	6	5	4	3	2	3	4	5	6

f(N) = h(N), with h(N) = Manhattan distance to the goal

Robot Navigation

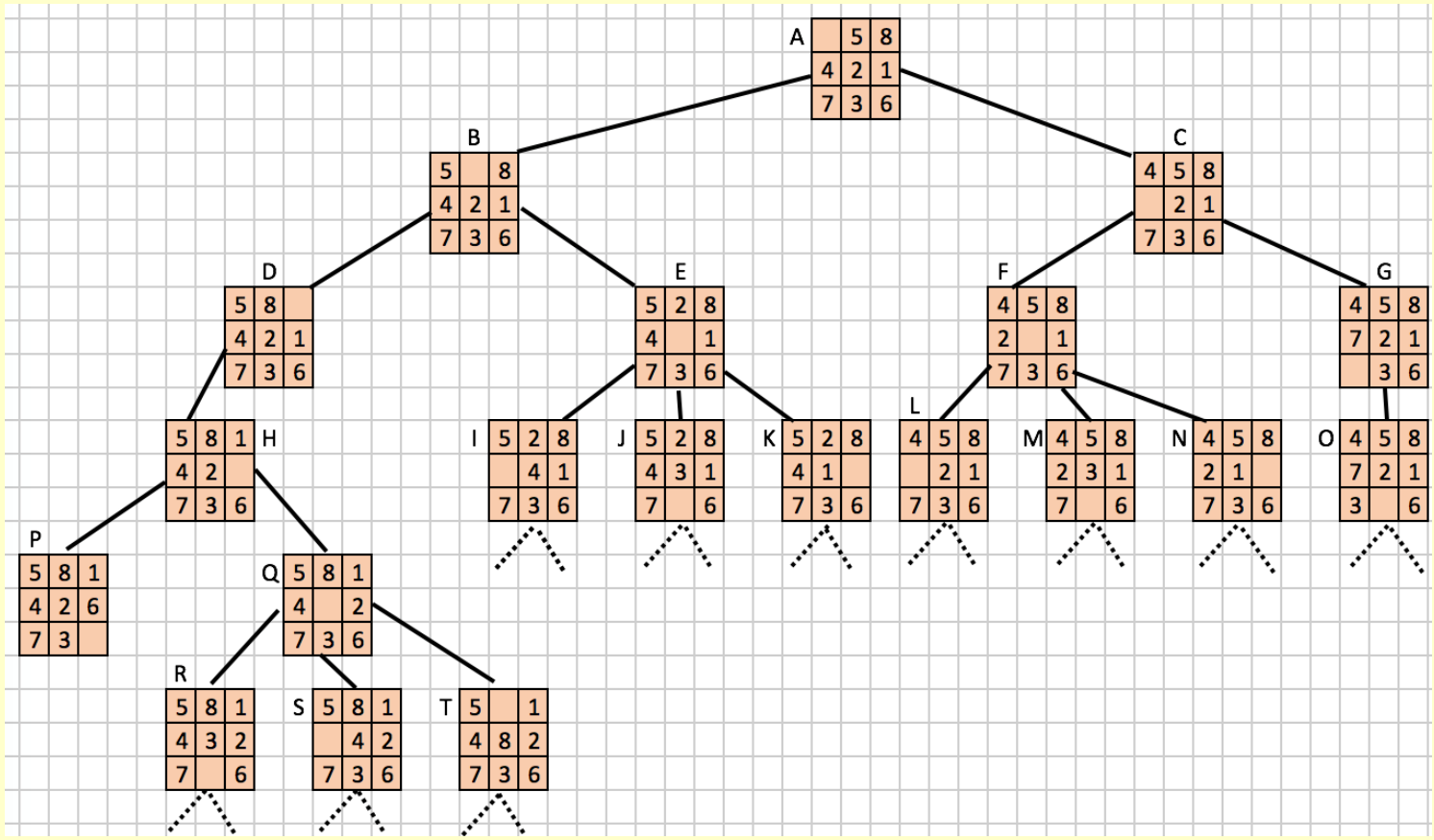
A* - Informed Search

8+3	7+4	6+3	5+6	4+7	3+8	2+9	3+10	4	5	6
7+2		5+6	4+7	3+8						5
6+1			3	2+9	1+10	0+11	1	2		4
7+0	6+1									5
8+1	7+2	6+3	5+4	4+5	3+6	2+7	3+8	4	5	6

f(N) = g(N)+h(N), with h(N) = Manhattan distance to goal

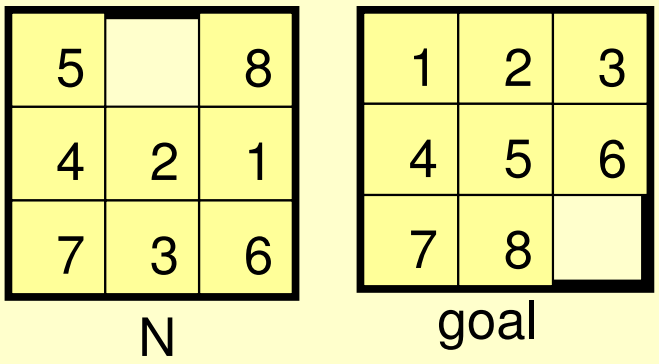
Puzzle Game

	5	8
4	2	1
7	3	6



Puzzle Game

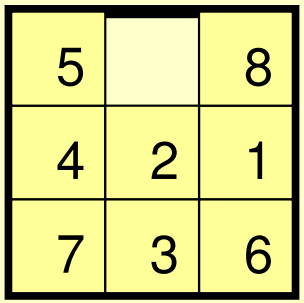
- Function h(N) that estimates the cost of the cheapest path from node N to goal node.
- Example: 8-puzzle



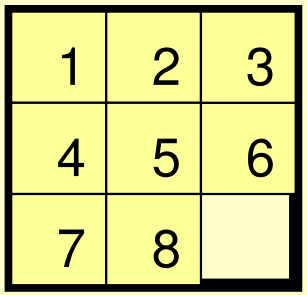
h(N) = number of misplaced tiles
= 6

Heuristic Function

- Function h(N) that estimate the cost of the cheapest path from node N to goal node.
- Example: 8-puzzle



N



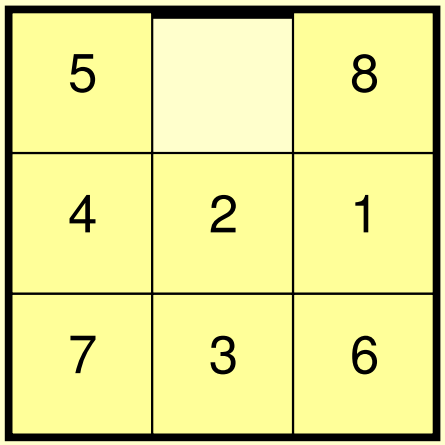
goal

h(N) = sum of the distances of every tile to its goal position

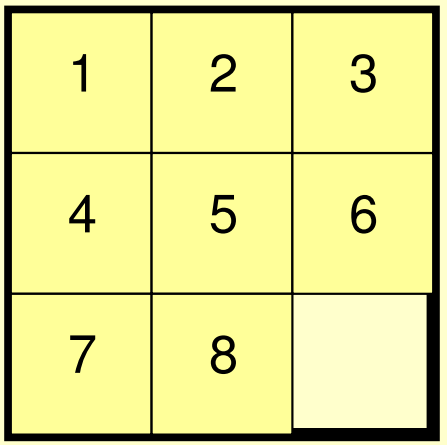
= 2 + 3 + 0 + 1 + 3 + 0 + 3 + 1

= 13

Heuristic Function for 8-Puzzle



N

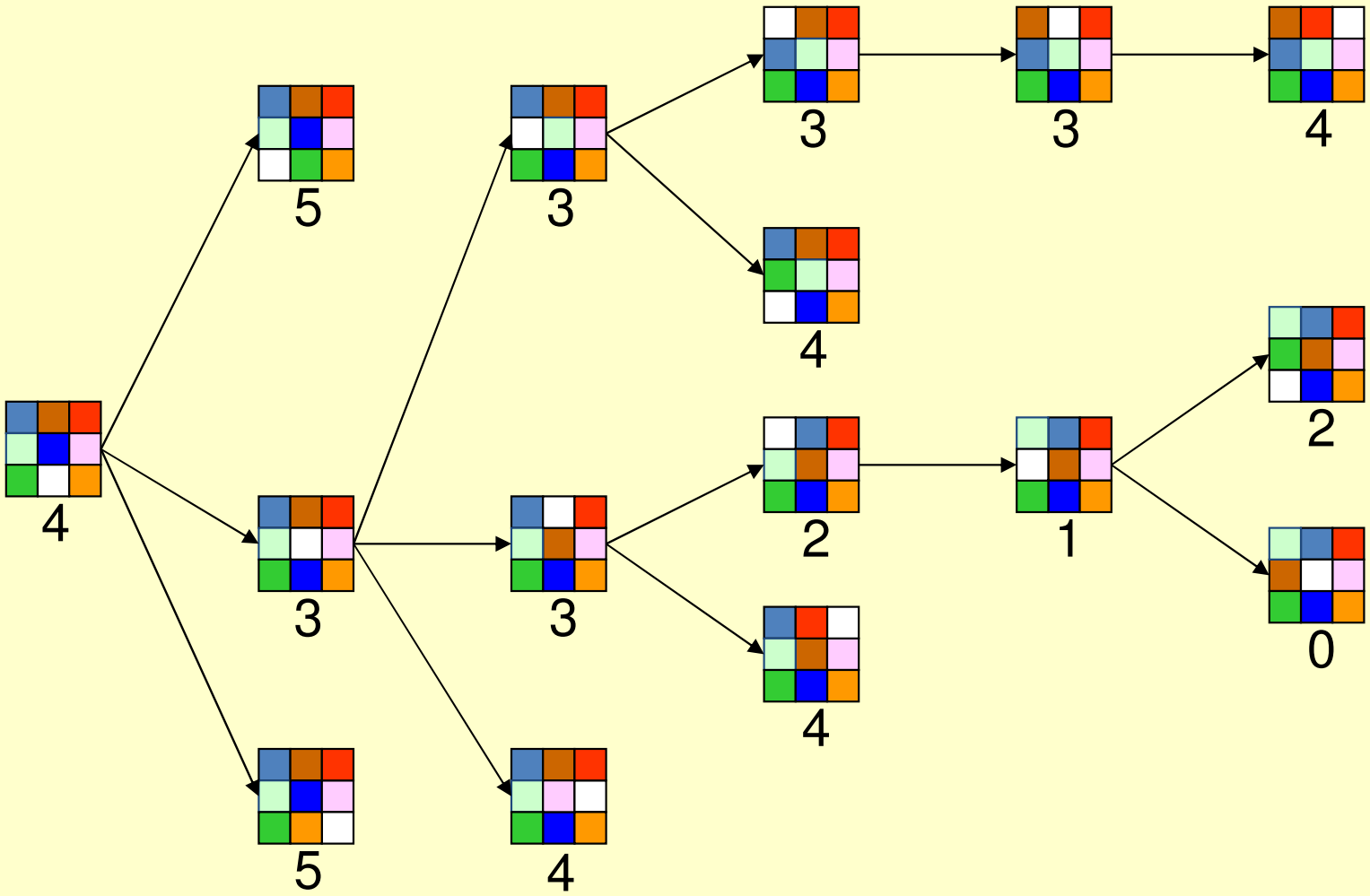


goal

- h1(N) = number of misplaced tiles = 6 → admissible
- h2(N) = sum of distances of each tile to goal = 13 → admissible
- h3(N) = (sum of distances of each tile to goal)
  + (sum of score functions for each tile) = 49 → not admissible

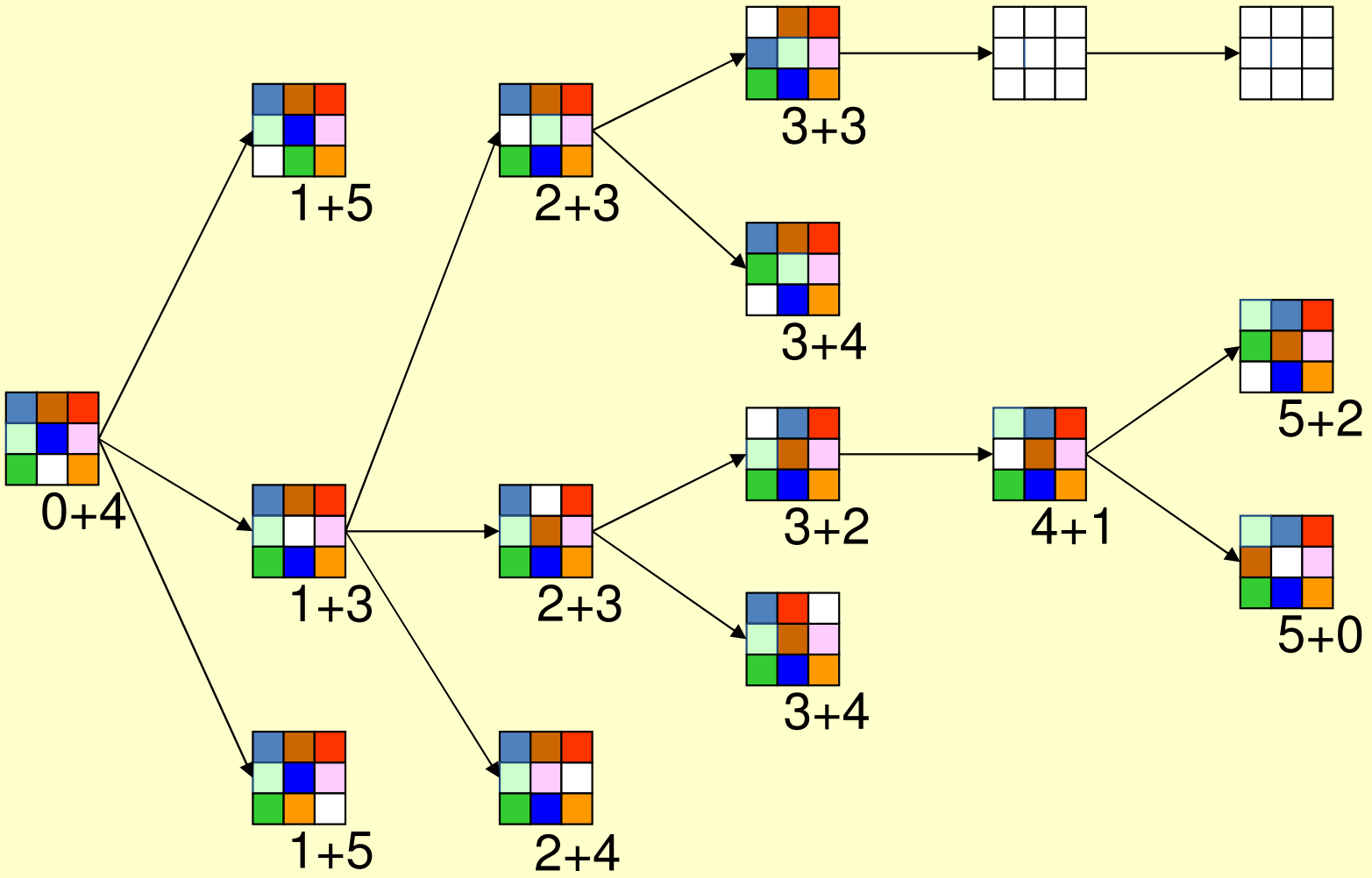
8-Puzzle

f(N) = h(N) = number of misplaced tiles



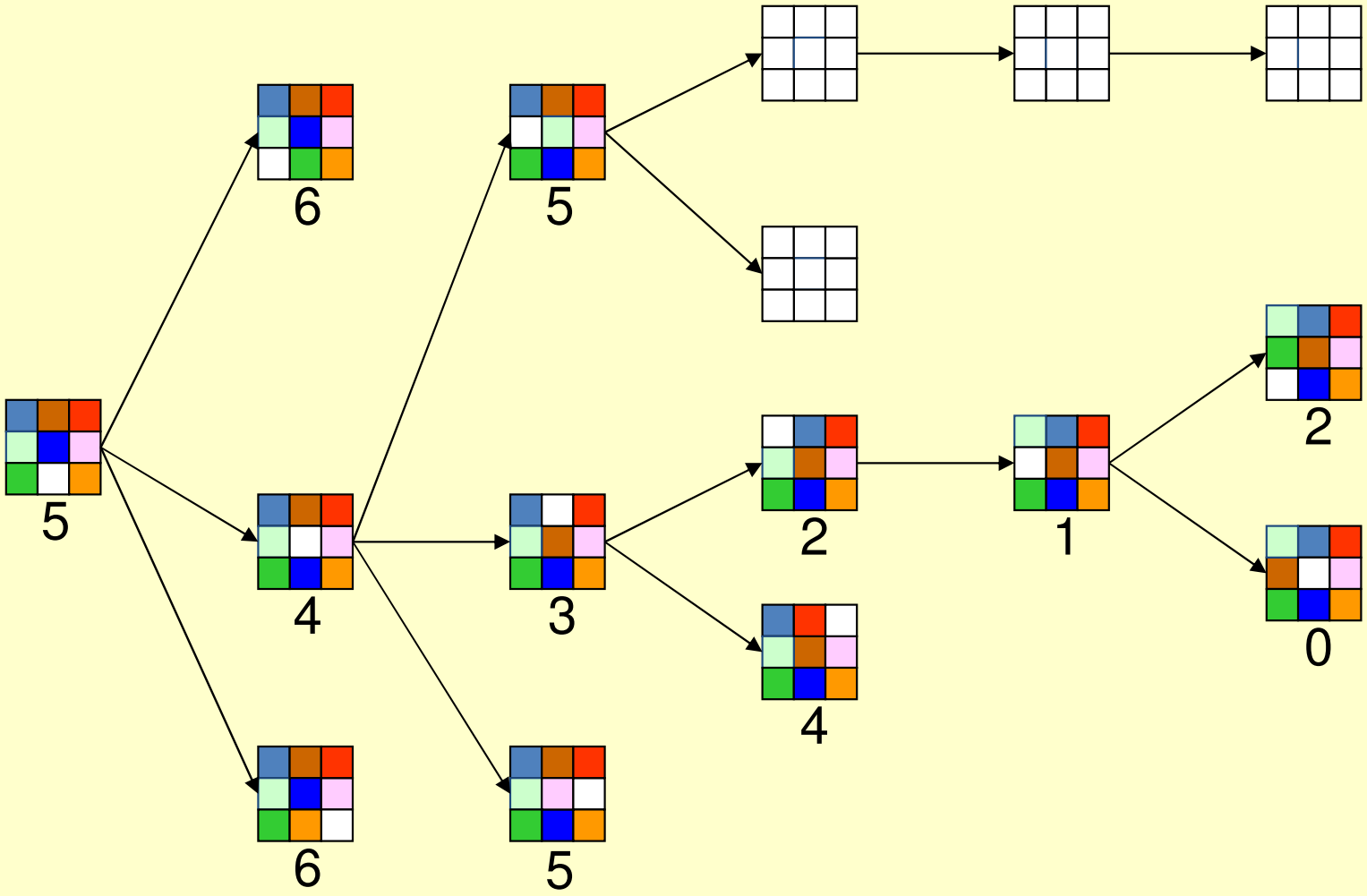
8-Puzzle

f(N) = g(N) + h(N)
with h(N) = number of misplaced tiles



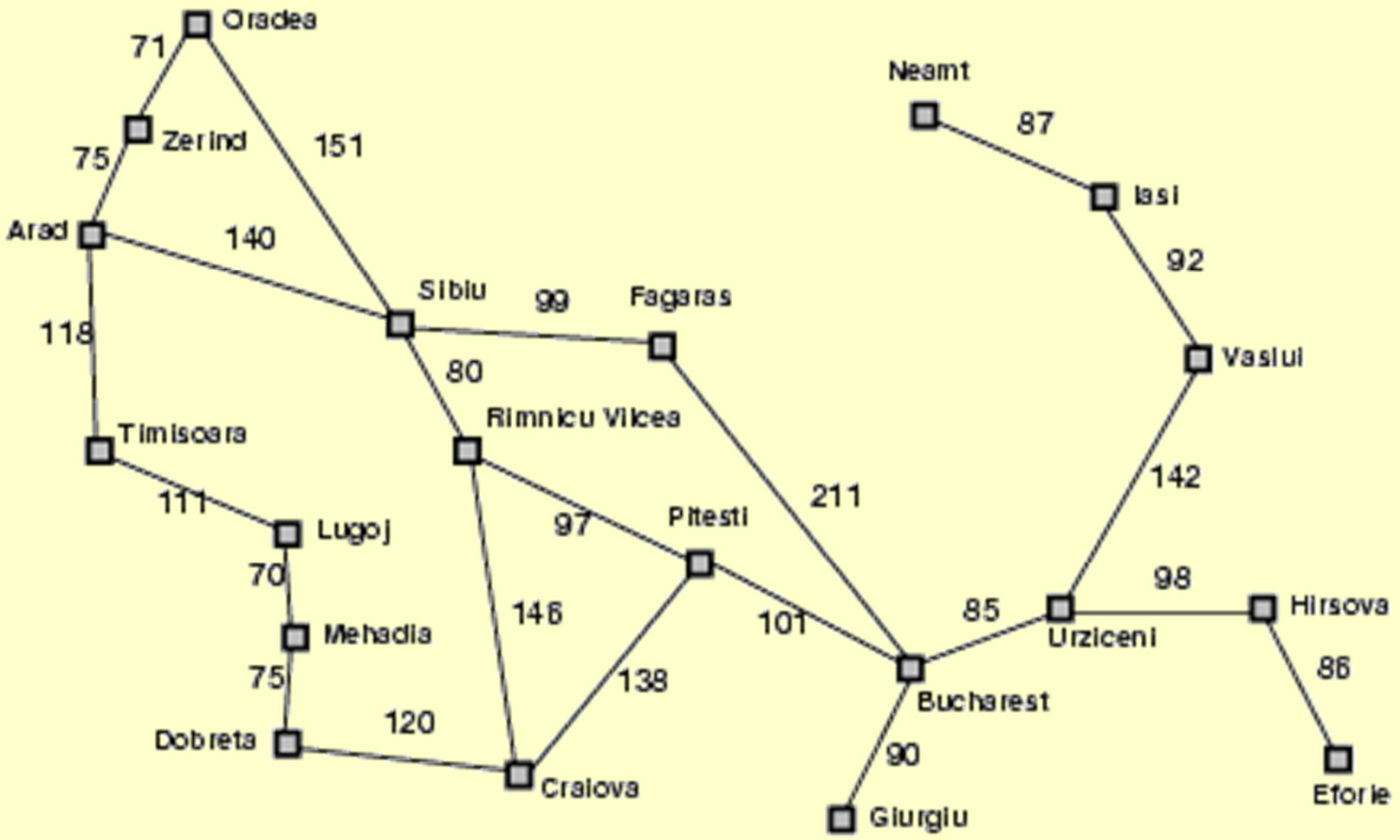
8-Puzzle

f(N) = h(N) = Σ distances of tiles to goal



Map Navigation

Romania with step costs in km



Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374