Posts and Telecommunication Institute of Technology
Faculty of Information Technology 1

Introduction to Artificial Intelligence

Informed search

Dao Thi Thuy Quynh

Blind search & Informed search

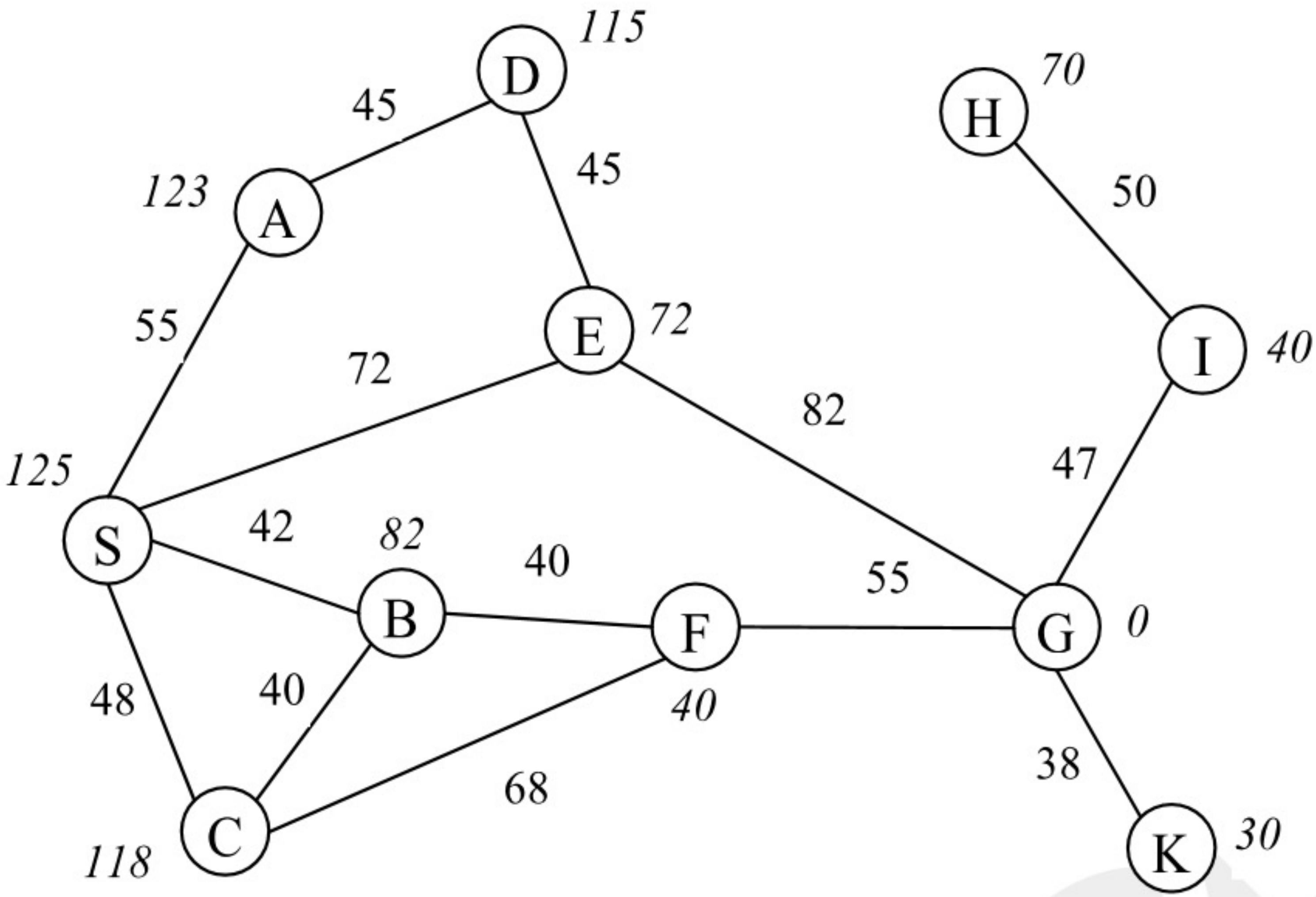
- Blind search
  - Expands nodes according to a pre-defined rule
  - Moves in the state space without orientation; many states must be considered
  - Not suitable for problems with large state spaces

- Informed search
  - Uses additional information from the problem to guide the search
  - Uses a function $f(n)$ to evaluate the potential of node n, then choose the best node to expand first
    - Best-first search
    - How to construct function $f(n)$?

Outline

- Greedy search
- A* search
- Heuristic functions
- Iterative deepening A* (IDA*)

Example



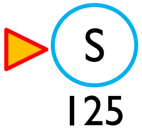
D
115
H
70
45
45
50
123
A
55
E
72
72
I
40
82
125
47
S
42
82
40
55
B
F
G
0
48
40
40
38
68
C
118
K
30

Greedy search

- **Principle**: expand the node with the cheapest path to a goal first
  - $f(n) = h(n)$: *heuristic* function, estimated cost of the path from n to a goal node
  - Example: $h(n)$ = straight-line distance from n to a goal node

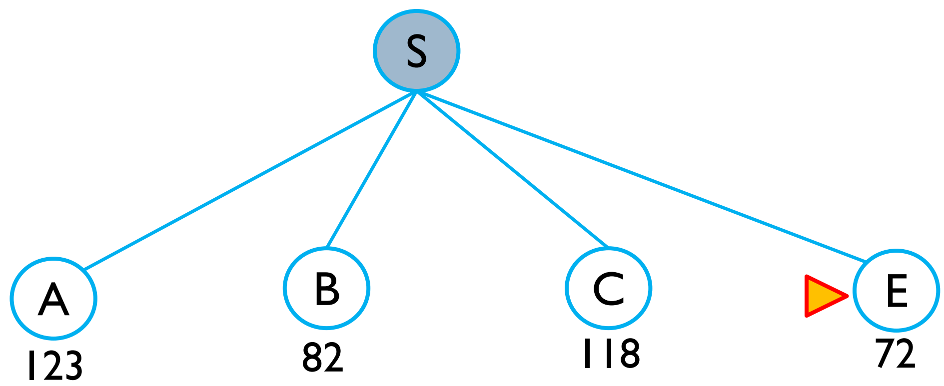
- "Greedy": expands the node that appears to be closest to goal, regardless of the future

Example of greedy search (1/4)

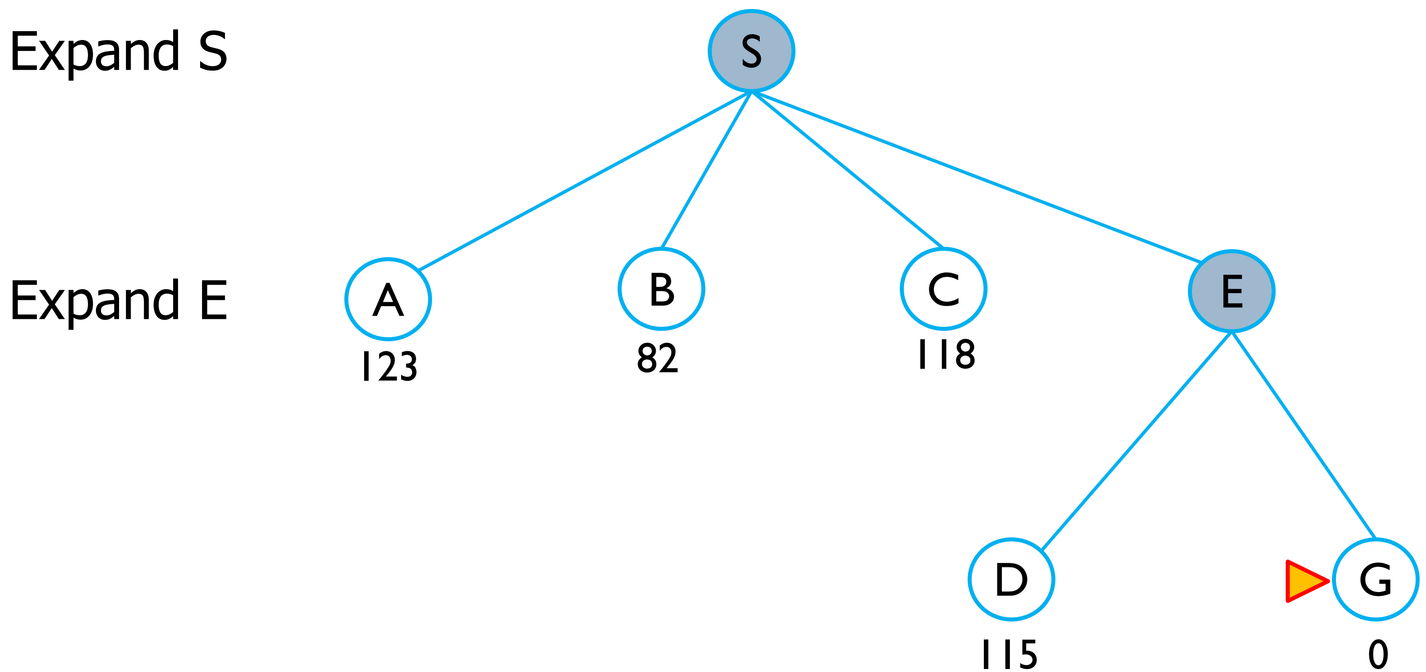


Example of greedy search (2/4)

Expand S



Example of greedy search (3/4)

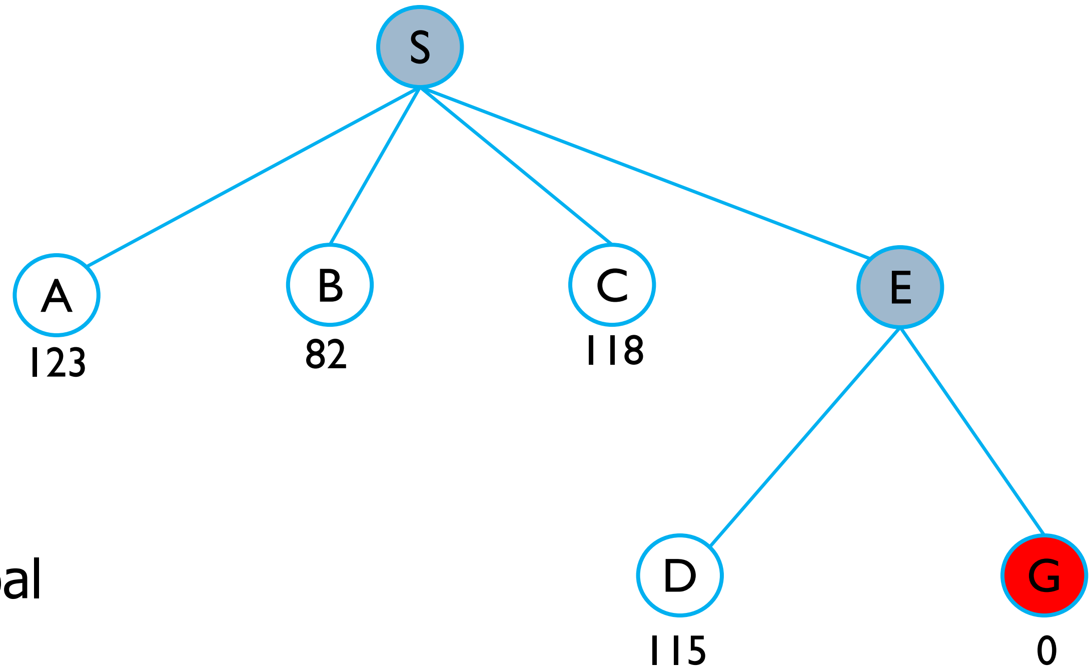


Example of greedy search (4/4)

Expand S

Expand E

Expand G: Goal

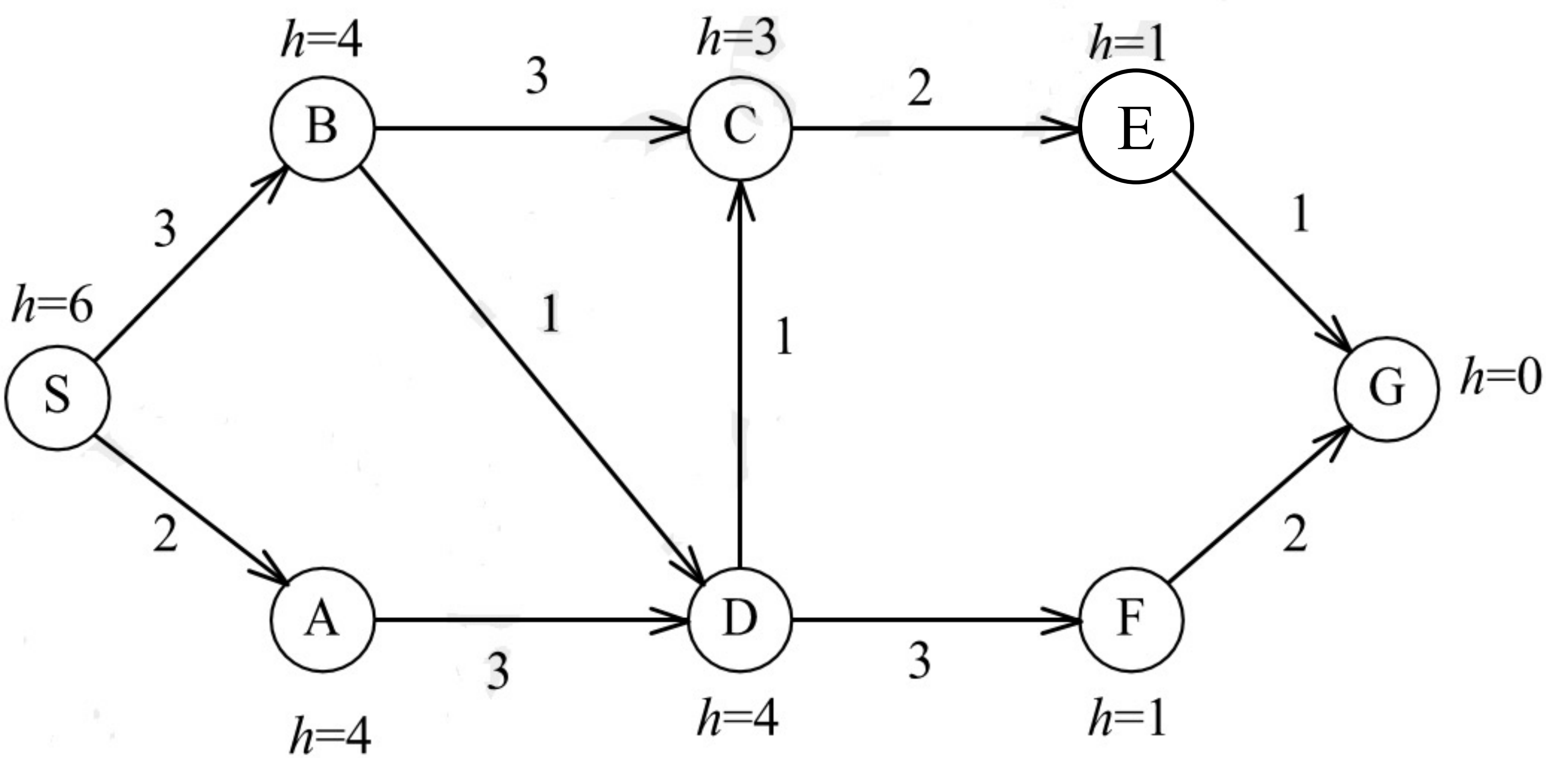


Properties of greedy search

- Completeness?
  - No (can have a loop, or have a branch of infinite nodes with small value of function h but do not lead to a goal)
- Optimality?
  - No
- Time?
  - $O(b^m)$
  - If the *heuristic* function is good, the algorithm can be much faster
- Space?
  - $O(b^m)$: store all nodes in the memory
  - If the *heuristic* function is good, the number of nodes to store can be reduced significantly

Exercise 1

- Use greedy search algorithm to find the path from S to G?



(Phuong TM, 2016)

Outline

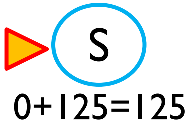
- Greedy search
- A* search
- Heuristic functions
- Iterative deepening A* (IDA*)

A* search: idea

- Overcome the disadvantage of greedy search
  - Greedy: only care about the path to the goal
  - A*: also care about the path from the initial node to the current node
    - avoid expanding paths that are already expensive

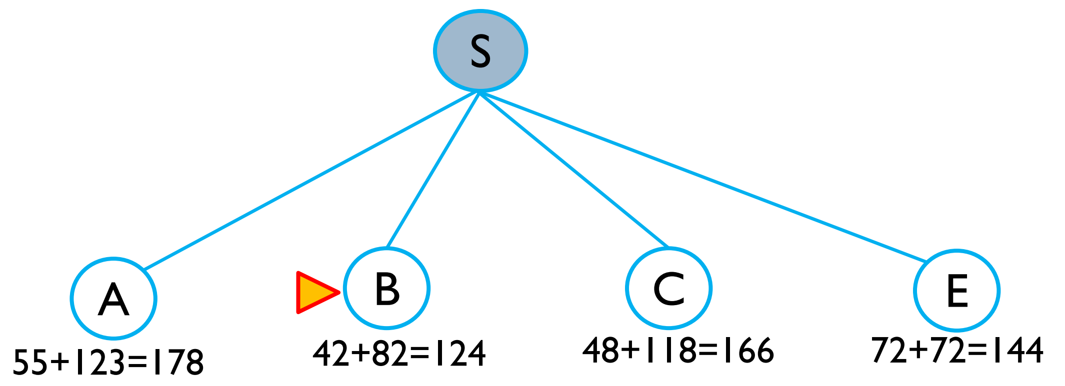
- **Method**: $f(n) = g(n) + h(n)$
  - $g(n)$: cost so far to reach n
  - $h(n)$: heuristic function, estimated cost from n to a goal node
  - $f(n)$: estimated total cost of the path from the initial node, through n, to a goal node

Example of A* search (1/5)



Example of A* search (2/5)

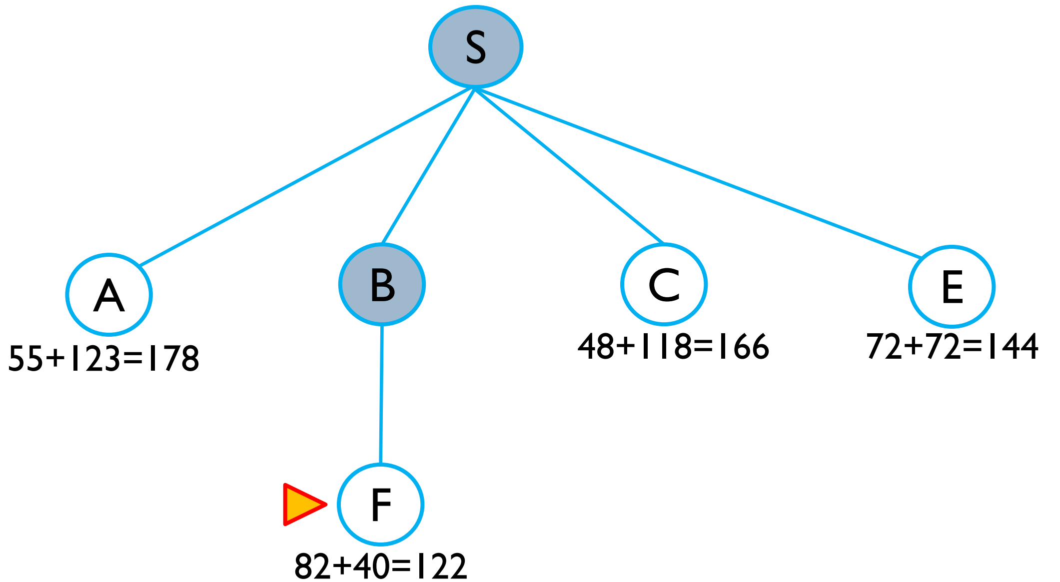
Expand S



Example of A* search (3/5)

Expand S

Expand B

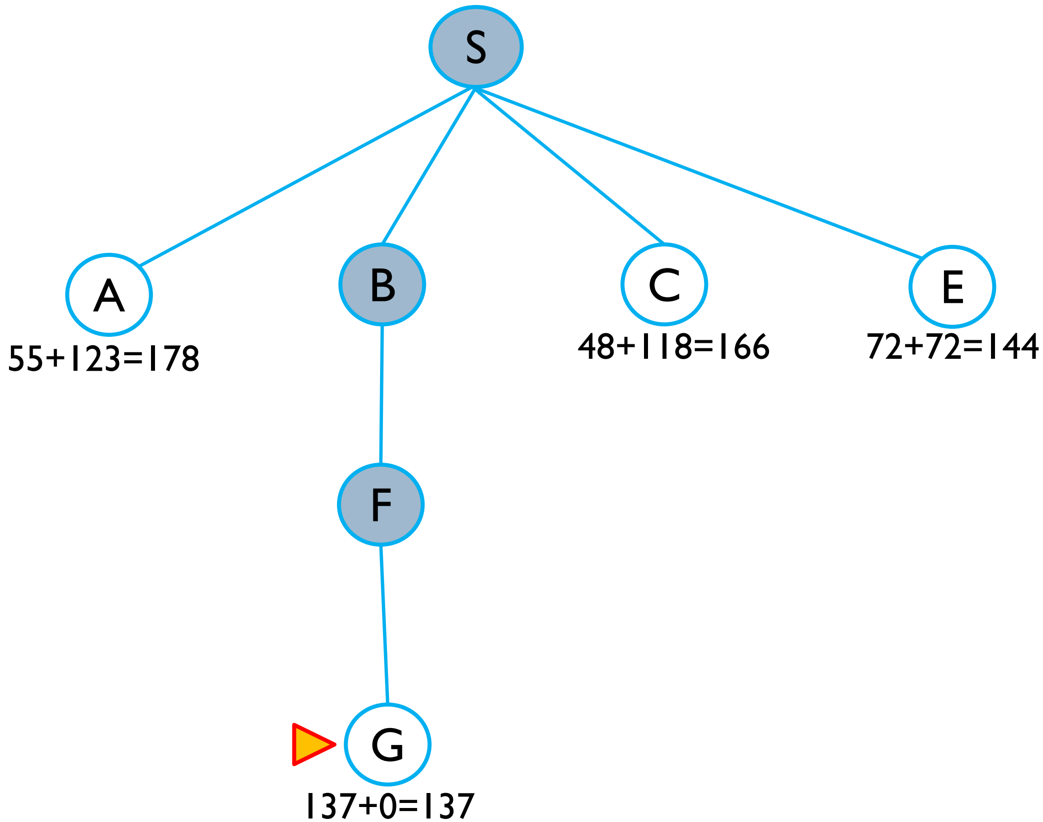


Example of A* search (4/5)

Expand S

Expand B

Expand F



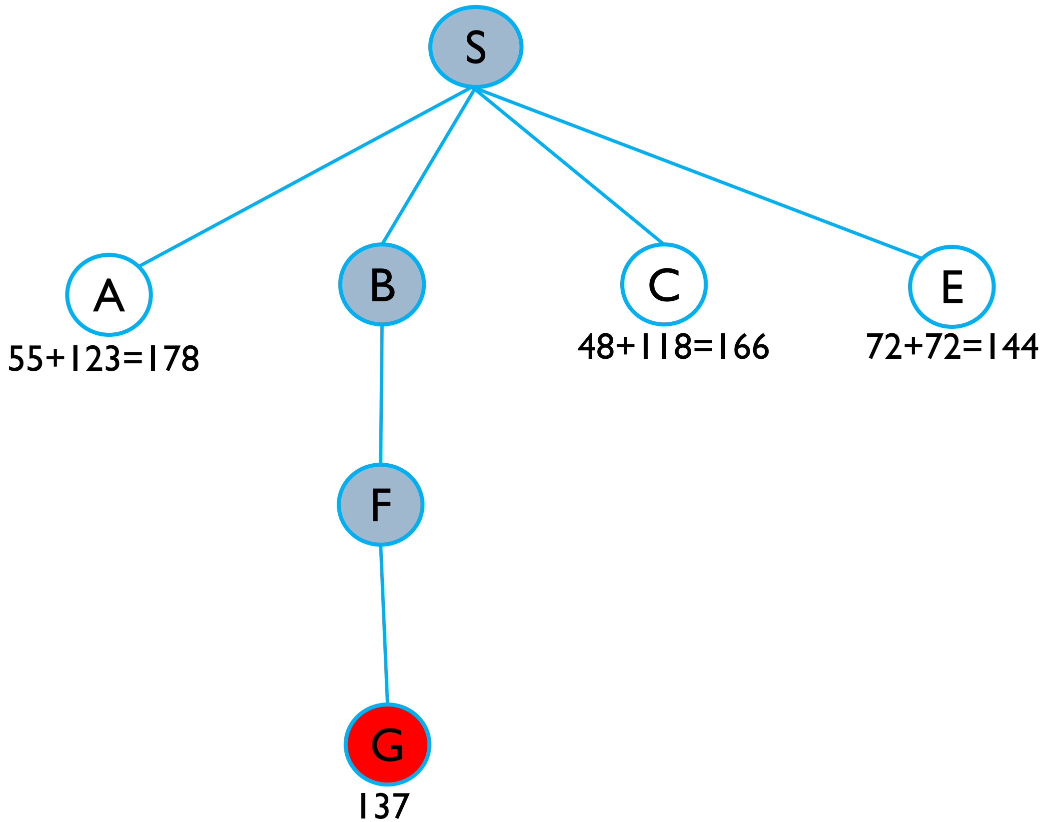
Example of A* search (5/5)

Expand S

Expand B

Expand F

Expand G: Goal



A* search algorithm

$A*(Q, S, G, P, c, h)$

(Q: state space, S: initial state, G: goals, P: successor function, c: cost,
h: heuristic)

Input: search problem, heuristic function h

Output: goal state (path to the goal state)

Initialize: $O \leftarrow S$ (O: the open node list)

while ($O \neq \emptyset$) **do**
1. take node n whose $f(n)$ is the smallest from O
2. **if** $n \in G$, **return** (path to n)
3. for each $m \in P(n)$
   a) $g(m) = g(n) + c(n, m)$
   b) $f(m) = g(m) + h(m)$
   c) add m along with $f(m)$ to O

return no solution

Properties of A* search

- Complete?
  - Yes (unless there are infinite nodes n with $f(n) \leq f(G)$)
- Optimality?
  - Yes (if heuristic function h is admissible)
- Time?
  - $O(b^m)$
  - If the *heuristic* function is good, the algorithm can be much faster
- Space?
  - $O(b^m)$: store all nodes in the memory
  - If the *heuristic* function is good, the number of nodes to store can be reduced significantly

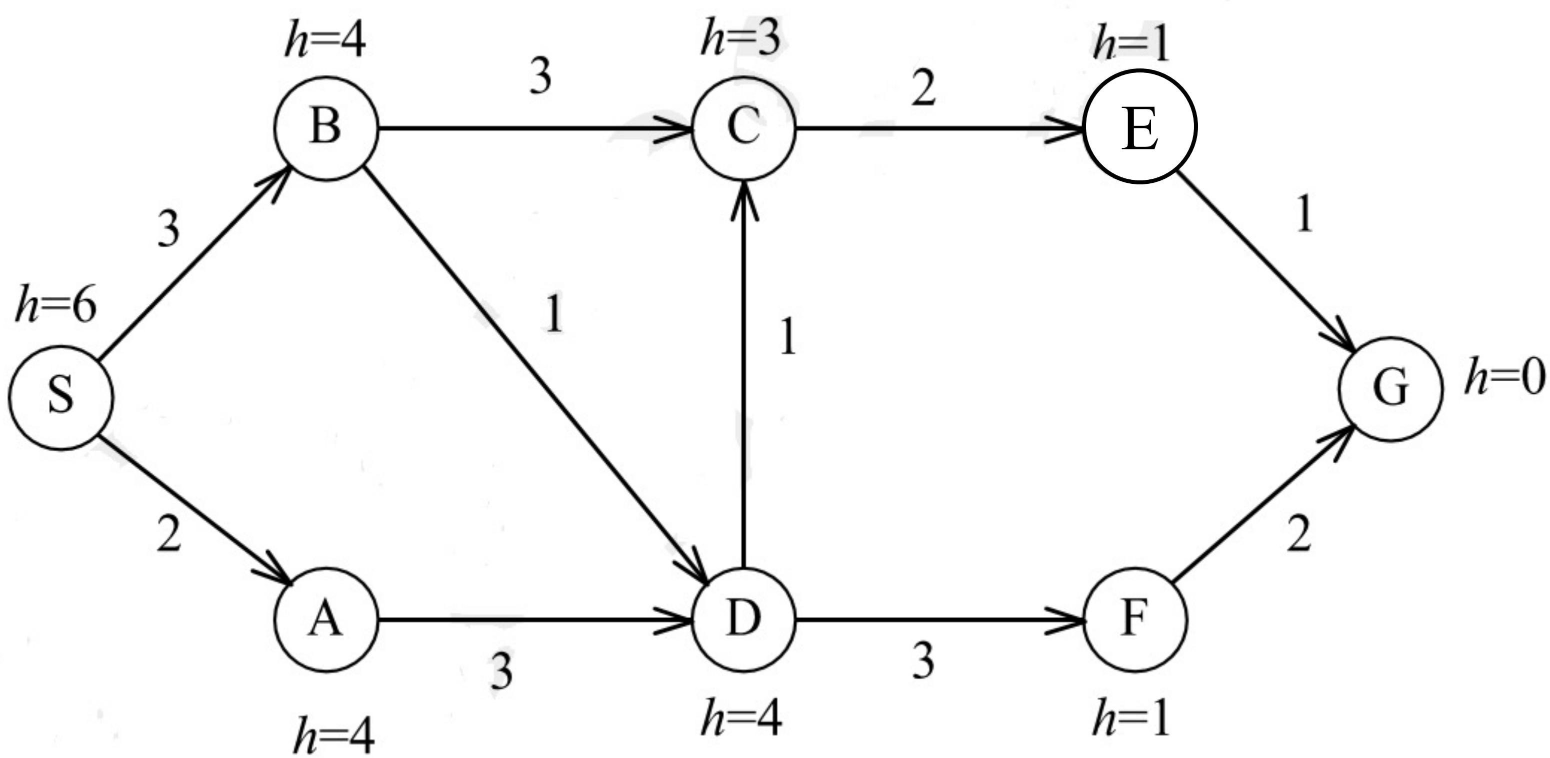
The optimality of A* search

- Admissible heuristic function
  - For every node n, $h(n) \leq h^*(n)$, where $h^*(n)$ is the true cheapest cost from n to a goal node
  - Example: straight-line distance is an admissible heuristic function

- **Theorem**: A* algorithm is guaranteed to find an optimal solution if $h(n)$ is admissible

Exercise 2

- Use A* search to find the path from S to G?



(Phuong TM, 2016)

Outline

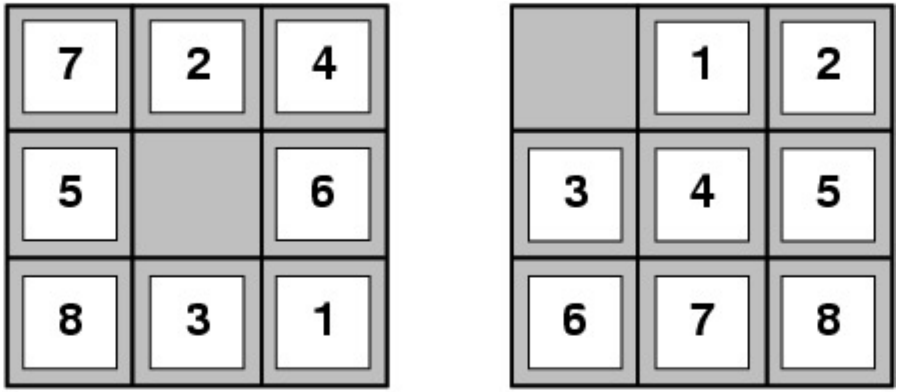
- Greedy search
- A* search
- Heuristic functions
- Iterative deepening A* (IDA*)

Heuristic functions

- Heuristic functions are constructed depending on each specific problem
  - A problem may have some heuristic functions
  - The quality of the heuristic function greatly affects the search process

- Dominance
  - If $h_1(n)$ and $h_2(n)$ are 2 admissible heuristic functions satisfying $h_1(n) \leq h_2(n)$ for all nodes n, then h_2 dominates (is better than) h_1

Example of heuristic functions



Start State Goal State

- $h_1(n)$: number of wrong cells
  - $h_1(S) = 8$
- $h_2(n)$: total of Manhattan distances
  - $h_2(S) = 3 + 1 + 2 + 2 + 2 + 3 + 3 + 2 = 18$

Outline

- Greedy search
- A* search
- Heuristic functions
- Iterative deepening A* (IDA*)

Iterative deepening A* – IDA*

- **Goal:** deals with the memory problem in A* search
  - a variant of IDS (uses the heuristic function from A* search to limit nodes)

- **Method**: Iterate DFS over sub-trees where $f(n)$ is not greater than a threshold
  - The threshold is incremented after each loop so that we have new nodes

IDA* algorithm

$IDA^*(Q, S, G, P, c, h)$

Input: search problem, heuristic function h

Output: goal state (path to the goal state)

Initialize: $O \leftarrow S$ (O: the open node list)

Threshold $i \leftarrow 0$

while (1) **do**

1. **while** $(O \neq \emptyset)$ **do**

   a) Take the first node n from O

   b) **if** $n \in G$, **return** (path to n)

   c) For each $m \in P(n)$

      i) $g(m) = g(n) + c(m, n)$

      ii) $f(m) = g(m) + h(m)$

      iii) **if** $f(m) \leq i$ **then** add m to the head of O

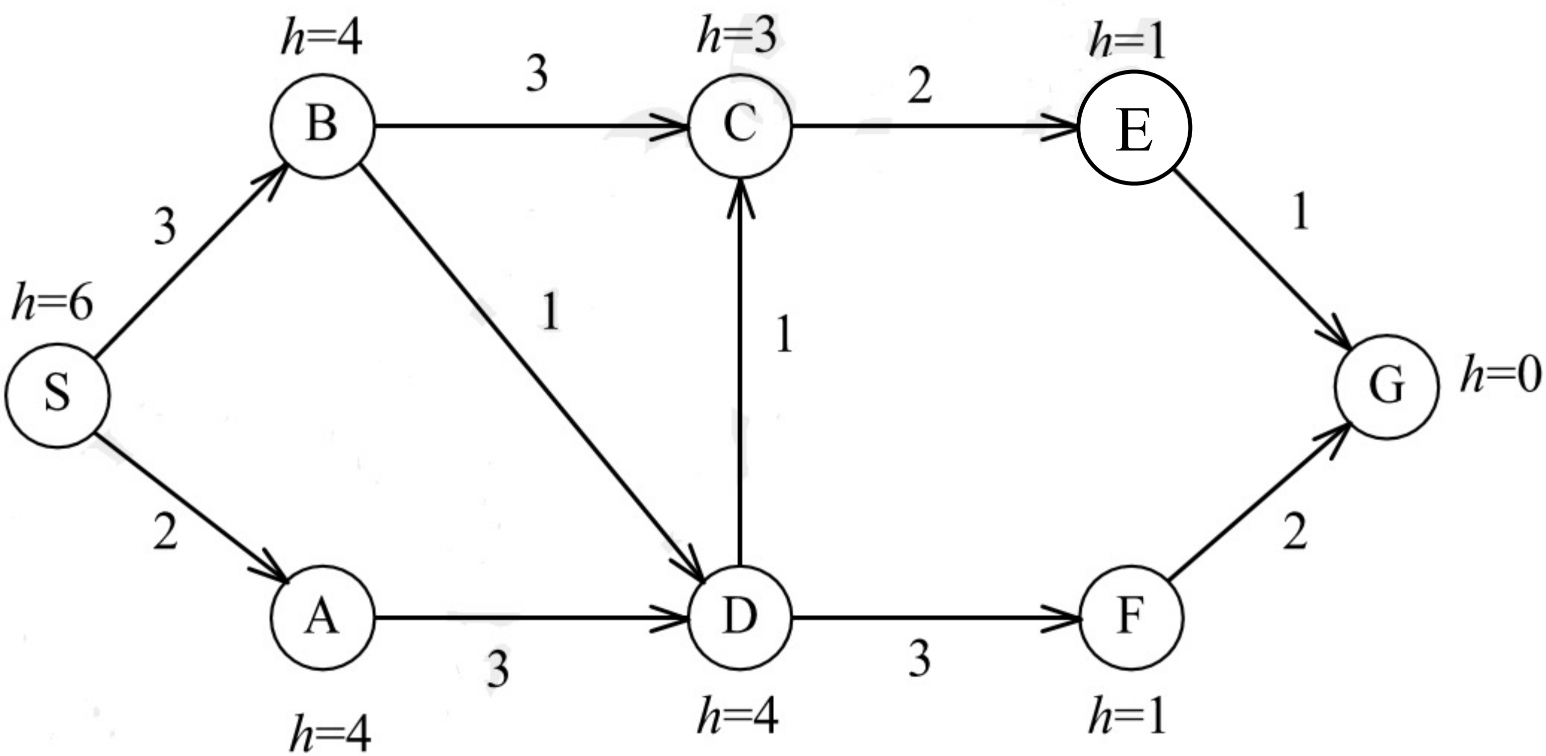
2. $i \leftarrow i + \beta, O \leftarrow S$

Properties of IDA*

- Completeness?
  - Yes
- Optimality?
  - β-optimal (cost of the found solution does not exceed β compared to cost of the optimal solution)
- Time?
  - Computational complexity is greater than that of A* search
- Space?
  - Requires linear memory

Exercise 3

- Use IDA* search to find the path from S to G with $\beta = 2$?



(Phuong TM, 2016)

When to add repeated nodes to the open node list?

- Greedy
  - **No**: adding repeated nodes does not change the algorithm (may lead to loops)
- A*
  - In cases the repeated node has better cost, it will be added to the list (if it is already expanded) or updated to replace the old node (if it is on the list)
- IDA*:
  - **Yes**