COMP3308/COMP3608, Lecture 3a
ARTIFICIAL INTELLIGENCE

A* Algorithm

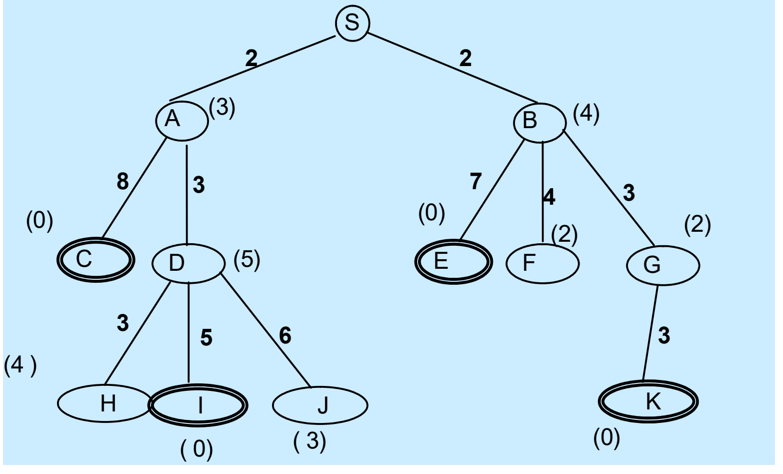
Reference: Russell and Norvig, ch. 3

Outline

- **A* search algorithm**
- **How to invent admissible heuristics**

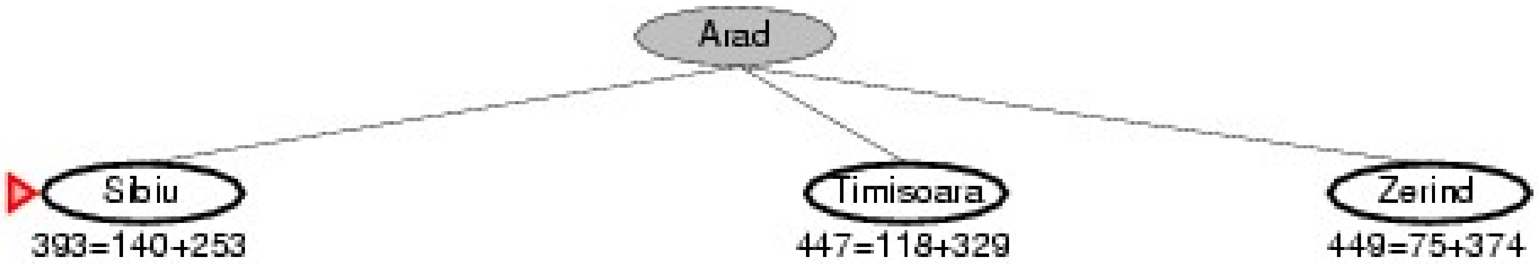
A* Search

- **UCS minimizes the cost so far *g(n)***
- **GS minimizes the estimated cost to the goal *h(n)***
- **A* combines UCS and GS**
- **Evaluation function: *f(n)=g(n)+h(n)***
  - ***g(n)* = cost so far to reach *n***
  - ***h(n)* = estimated cost from *n* to the goal**
  - ***f(n)* = estimated total cost of path through *n* to the goal**

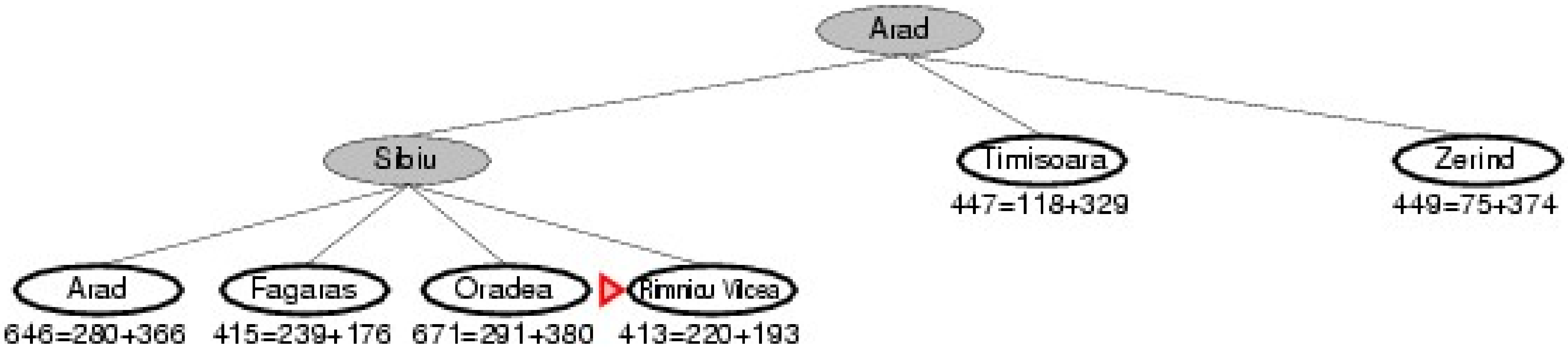


A* Search for Romania Example

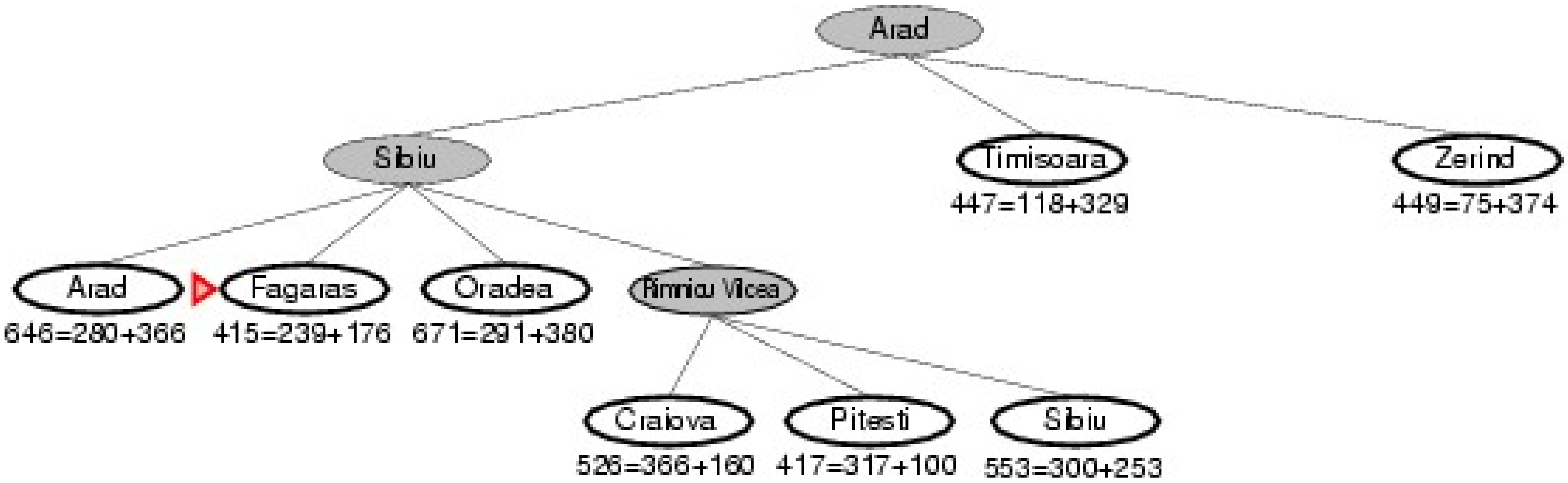
A* Search for Romania Example



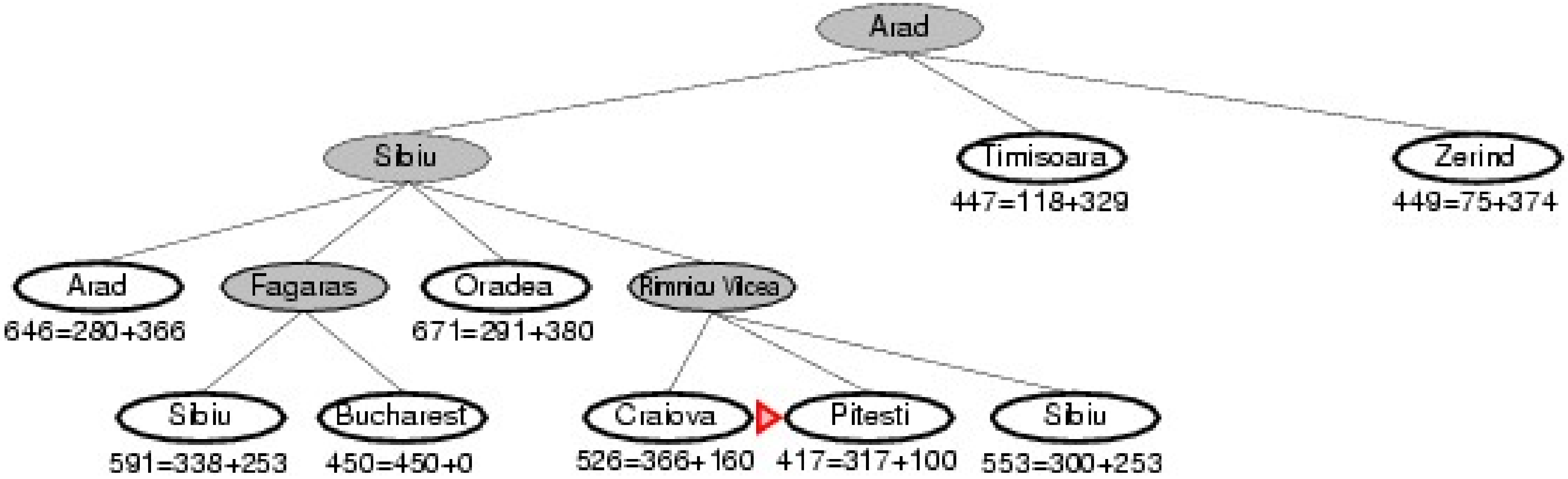
A* Search for Romania Example



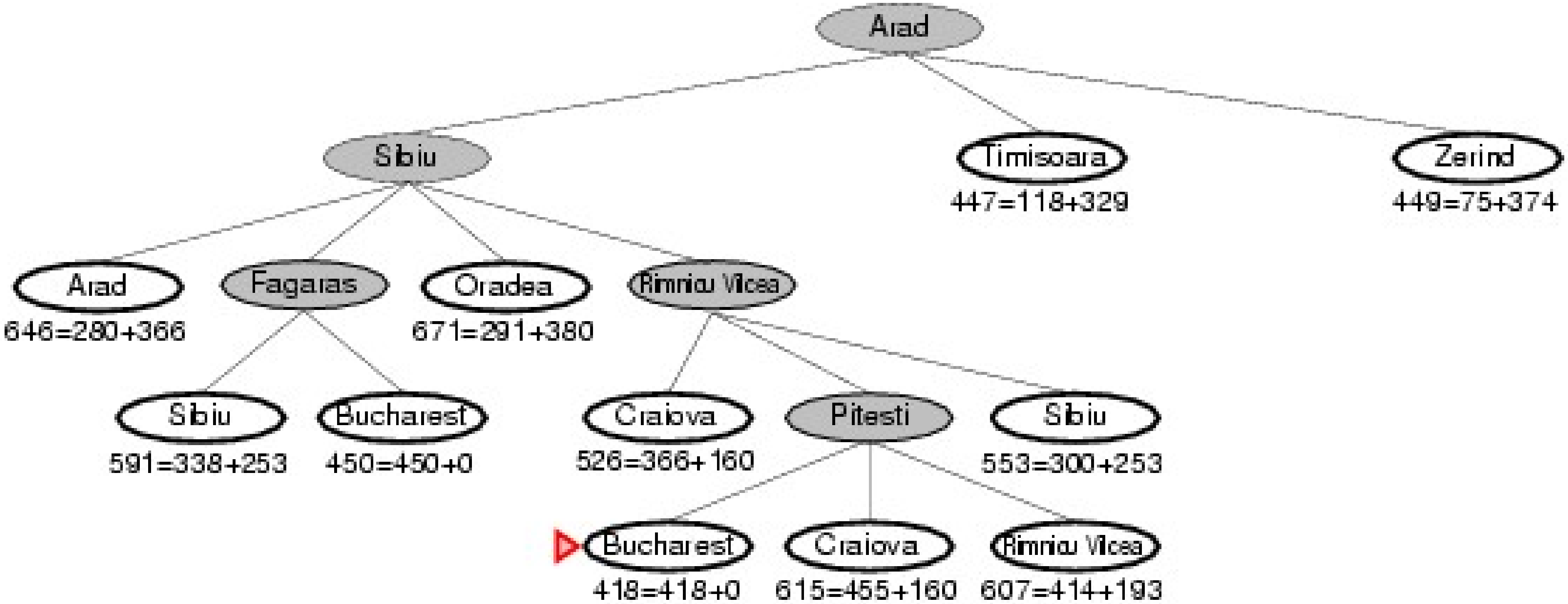
A* Search for Romania Example



A* Search for Romania Example



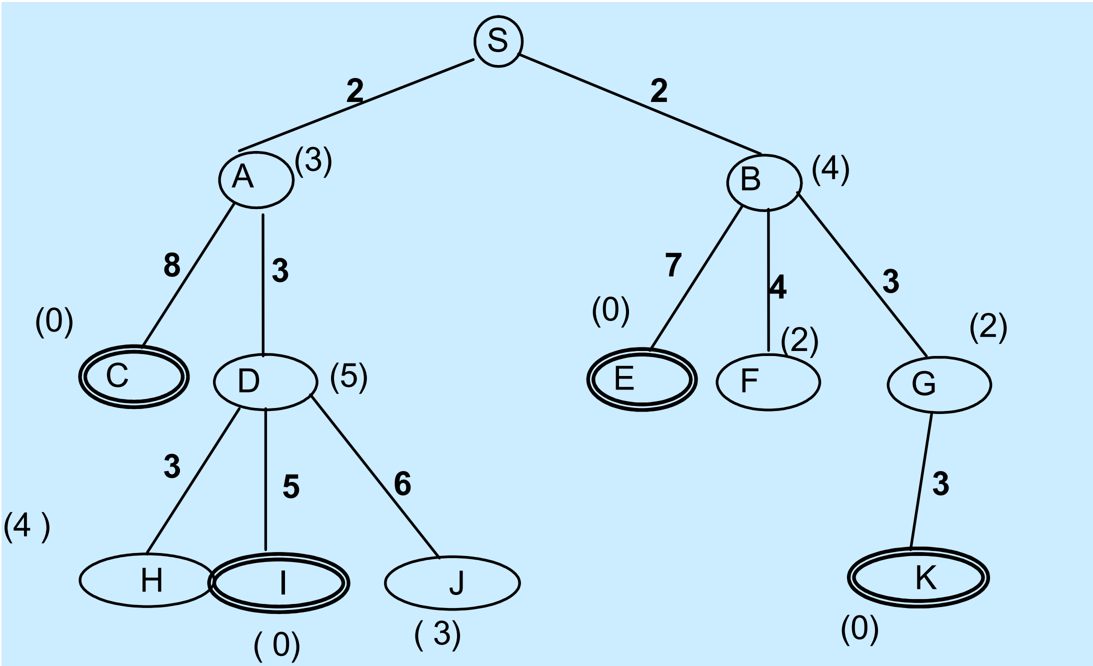
A* Search for Romania Example



Bucharest is selected for expansion and it is a goal node => stop
Solution path: Arad-Sibiu-Riminicu Vilcea-Pitesti-Bucharest, cost=418

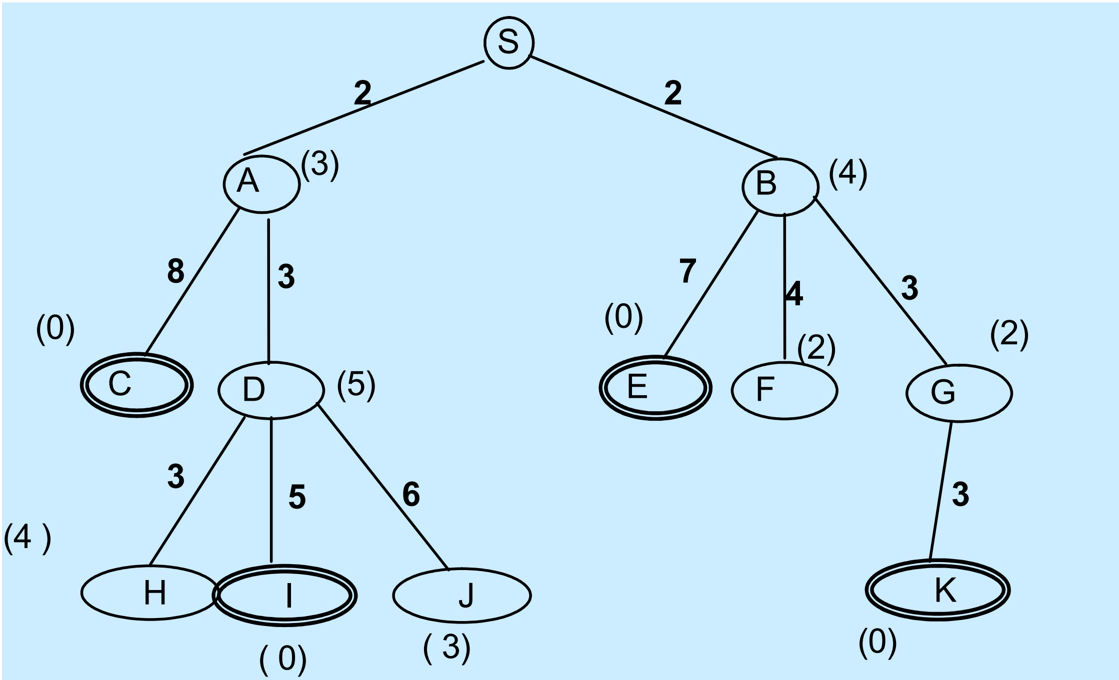
A* Search – Another Example

- **Given:**
  - **Goal nodes: C, I, E and K**
  - **Step path cost: along the links**
  - ***h* value of each node: in brackets ()**
  - **Same priority nodes -> expand the last added first**
- **Run A***
  - **list of expanded nodes =?**
  - **solution path =?**
  - **cost of the solution:=?**



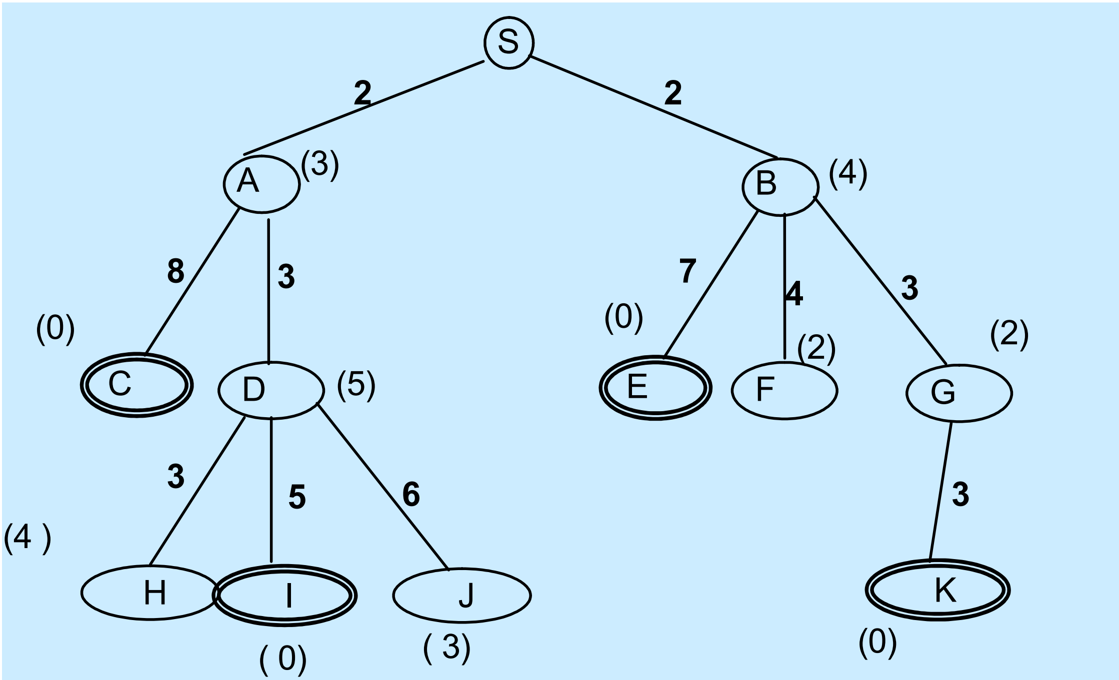
Solution

- **Fringe: S**
- **Expanded: nill**



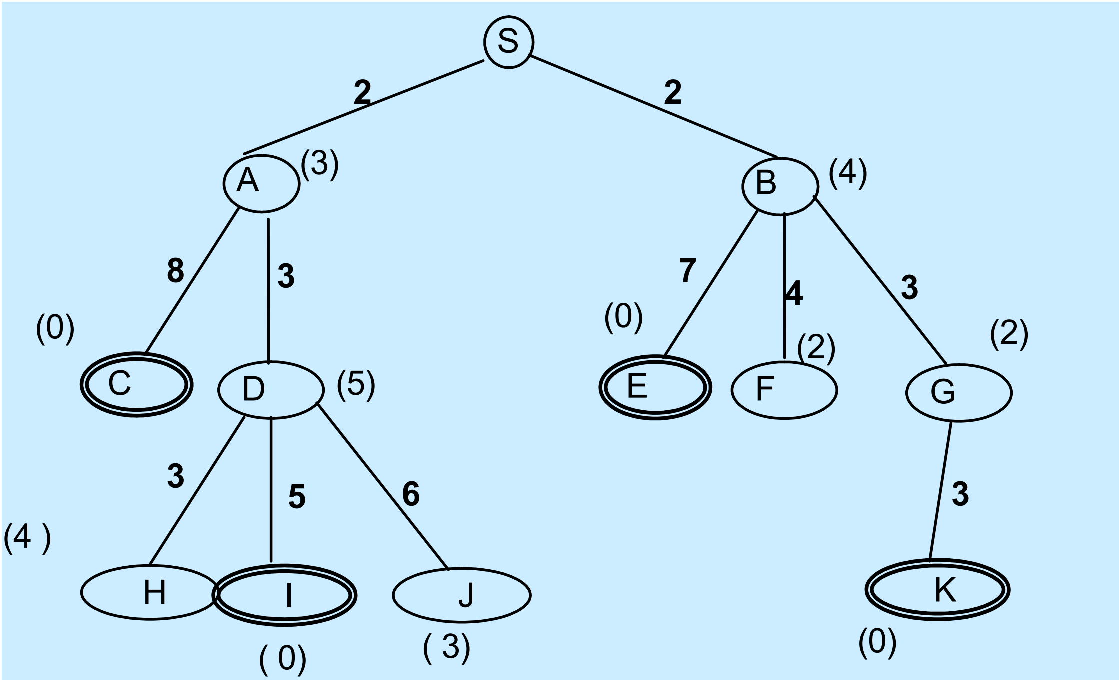
Solution

- **Fringe: (A, 5), (B, 6) //keep the fringe in sorted order**
- **Expanded: S**



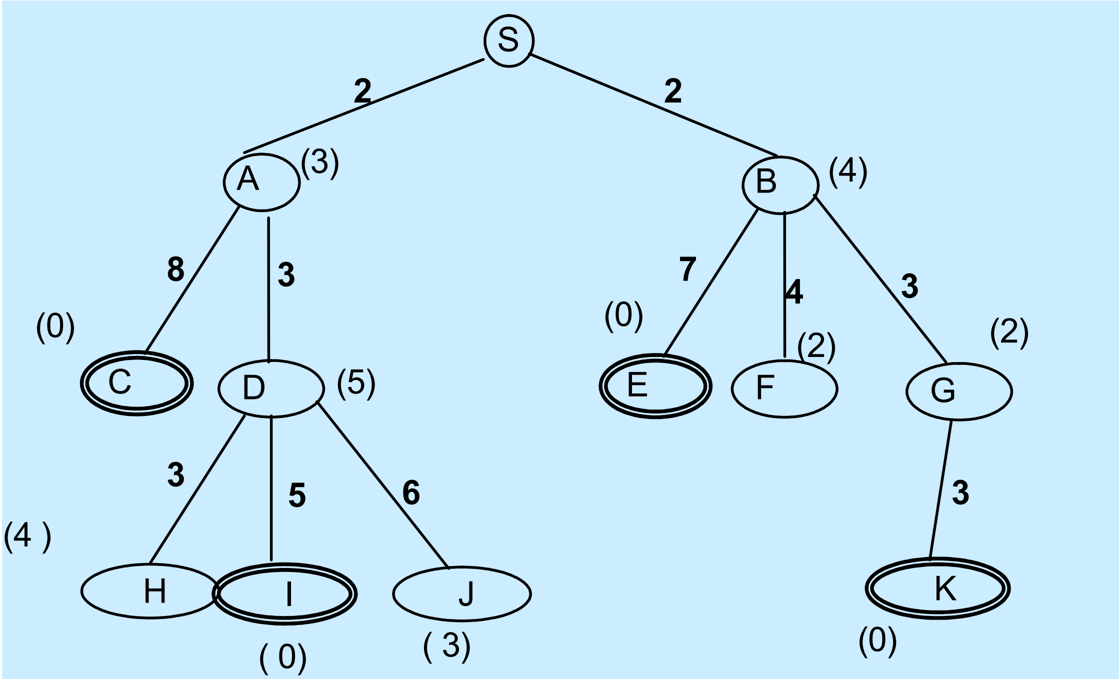
Solution

- **Fringe: (B, 6), (C, 10), (D, 10) //the added children are in blue**
- **Expanded: S, (A, 5)**



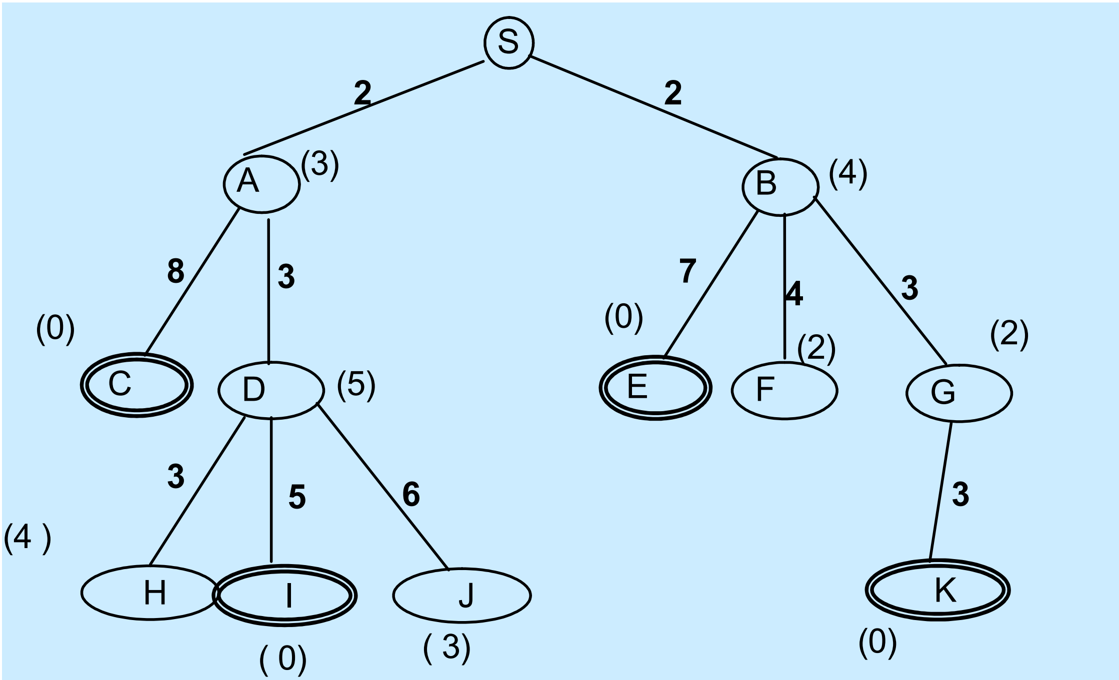
Solution

- **Fringe: (G, 7), (F, 8), (E, 9), (C, 10), (D, 10)**
- **Expanded: S, (A, 5), (B, 6)**



Solution

- **Fringe: (K, 8), (F, 8), (E, 9), (C, 10), (D, 10)**
- **Expanded: S, (A, 5), (B, 6), (G, 7)**

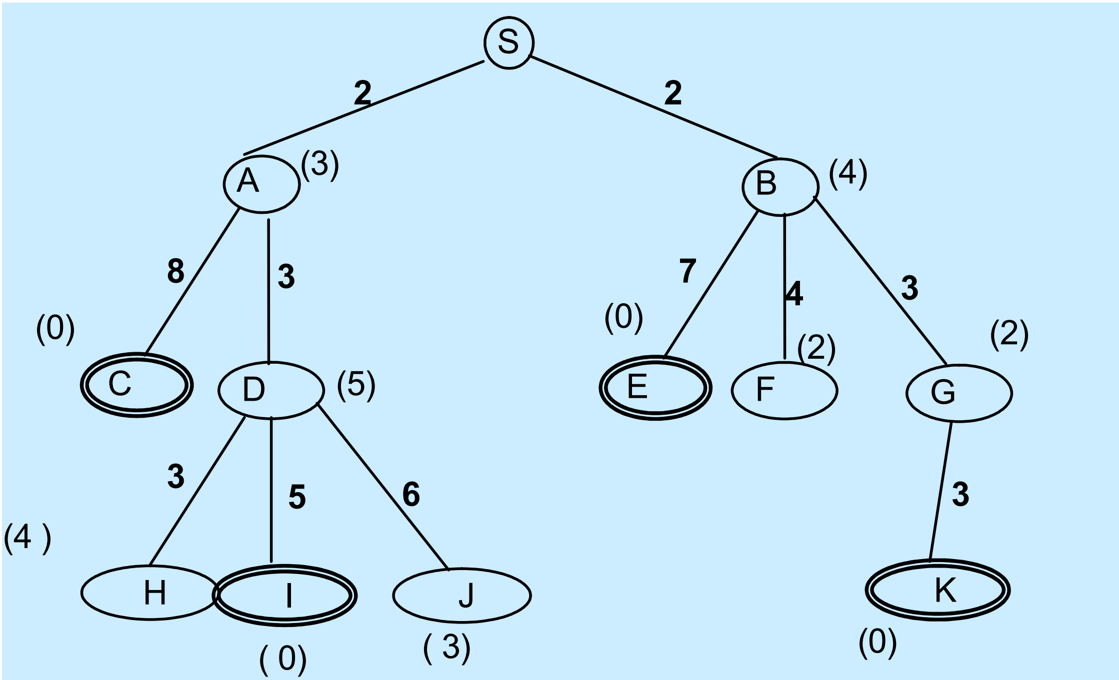


Solution

- **K is selected; Goal node? Yes => stop**

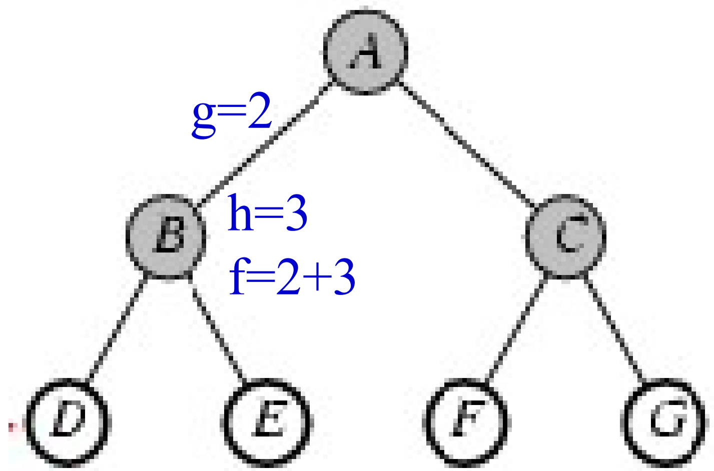
- **Expanded: S, A, B, G, K**
- **Solution path: SBGK, cost=8**

- **Is this the optimal solution=?**



A* and UCS

- **UCS is a special case of A* when *h(n)* =?**
- **In other words, when will A* behave as UCS?**

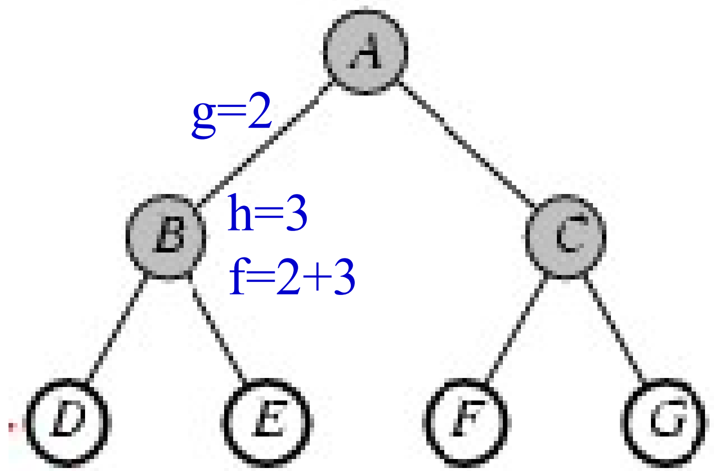


Hint:

- **UCS uses which cost?**
- **A* uses which cost?**
- **Relation between the 2 costs =?**

A* and UCS (Answer)

- UCS is a special case of A* when *h(n)* =?
- In other words, when will A* behave as UCS?

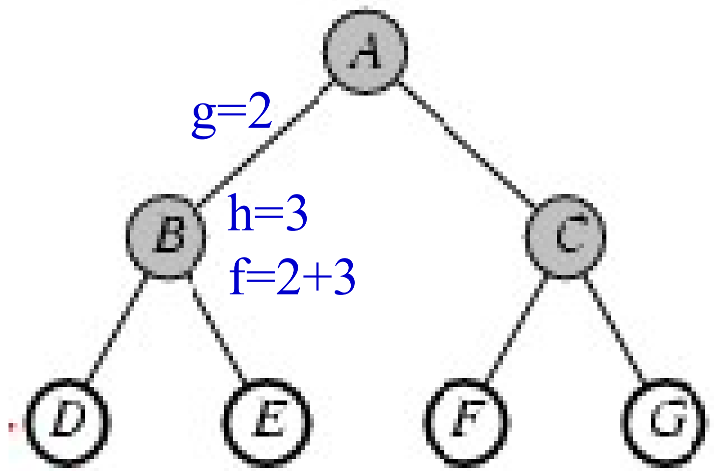


- UCS uses which cost?
- A* uses which cost?

- UCS: *g(n)*
- A*: *f(n)=g(n)+h(n)*
- if *h(n)=0* => *f(n)=g(n)*, i.e. A* becomes UCS

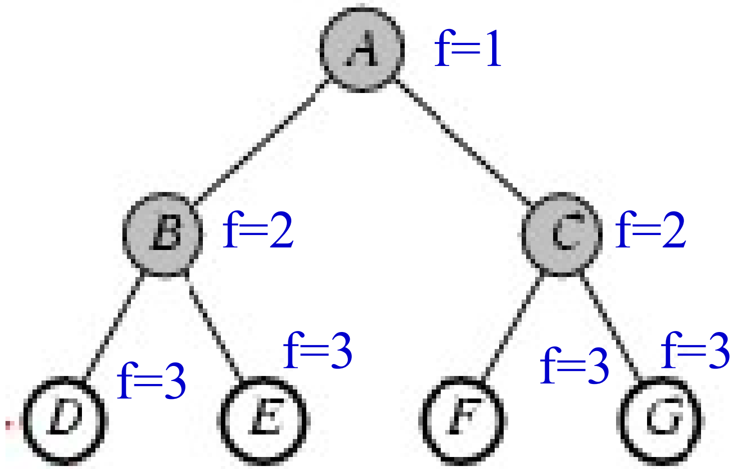
A* and BFS

- **BFS is a special case of A* when** ***f(n)*** **=?**
- **When will A* behave as BFS?**



A* and BFS (Answer)

- **BFS is a special case of A* when** ***f(n)*** **=?**

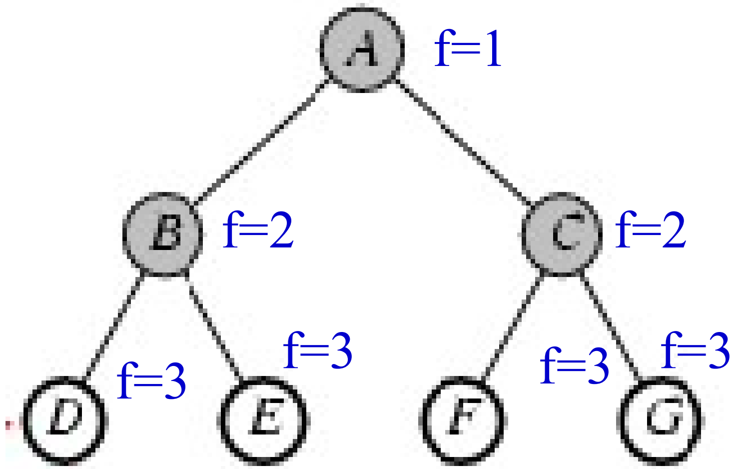


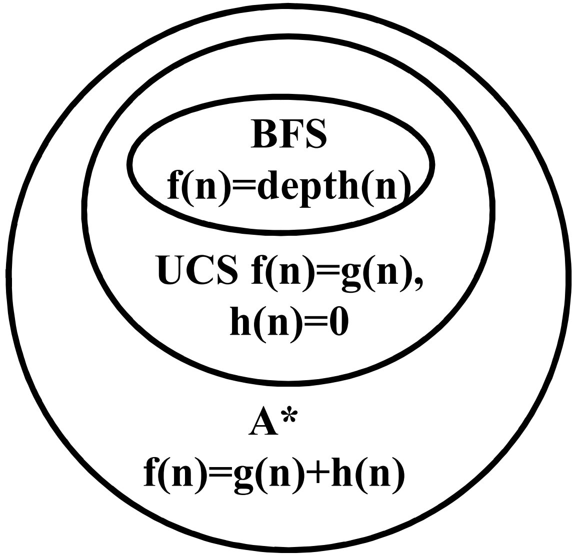
when ***f(n)=depth(n)***

- **And also when this assumption for resolving ties is true: among nodes with the same priority, the left most is expanded first**

BFS, UCS and A*

- **BFS is a special case of A* when *f(n)=depth(n)***
- **BFS is also a special case of UCS when *g(n)=depth(n)***
- **UCS is a special case of A* when *h(n)=0***





Admissible Heuristic

- **A heuristic *h(n)* is admissible if for every node *n*:**
  - **$h(n) \leq h^*(n)$ where $h^*(n)$ is the <u>true</u> cost to reach a goal from *n***
  - **i.e. the estimate to reach a goal is smaller than (or equal to) the true cost to reach a goal**

- **Admissible heuristics are *optimistic* – they think that the cost of solving the problem is less than it actually is!**
  - **e.g. the straight line distance heuristic $h_{SLD}(n)$ never overestimates the actual road distance (cost from *n* to goal) => it is admissible**

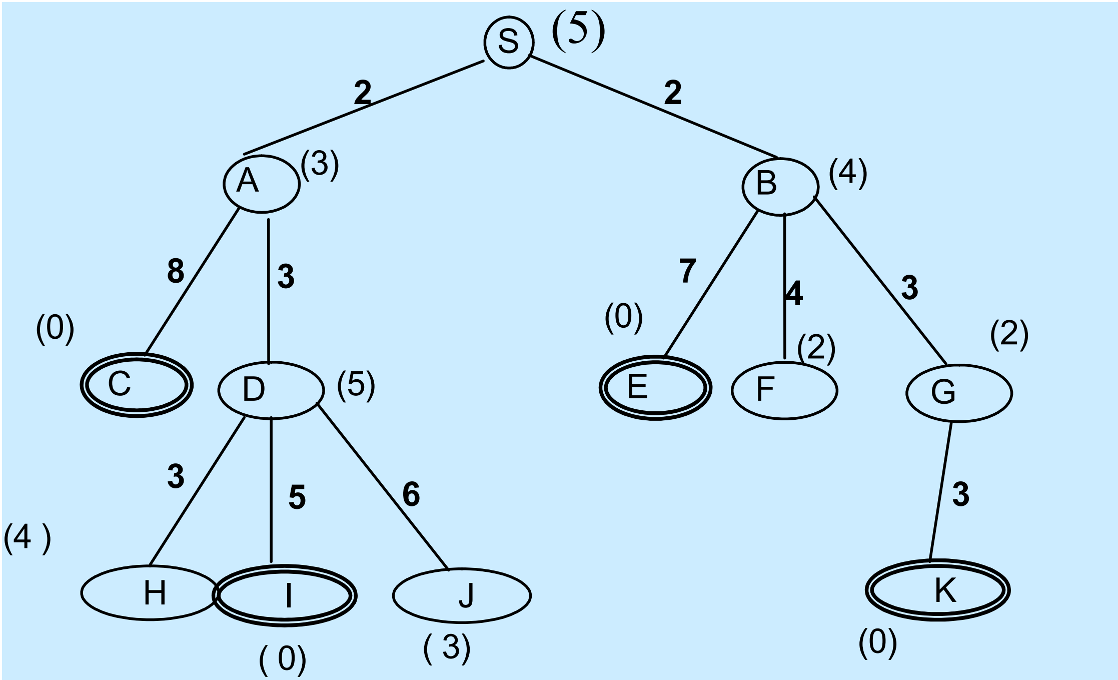
- **Theorem: If *h* is an *admissible heuristic*, then A* is complete and optimal**

Is *h* Admissible for Our Example?

- No need to check goal nodes (*h=0* for them) and nodes that are not on a goal path

- *h*(S)=5<=8 (shortest path from S to a goal, i.e. to goal K)
- *h*(B)=4<=6
- *h*(G)=2<=3
- *h*(A)=3<=8
- *h*(D)=5<=5

- => *h* is admissible



Optimality of A* - Proof

Optimal solution = the shortest (lowest cost) path to a goal node

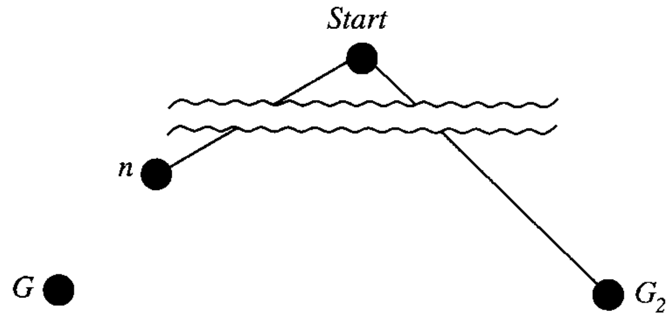
Idea: Suppose that some sub-optimal goal G_2 has been generated and it is in the fringe. We will show that G_2 can not be selected from the fringe.

<u>Given:</u>

G - the optimal goal

G_2 – a sub-optimal goal

h is admissible



<u>To prove:</u> G_2 can not be selected from the fringe for expansion

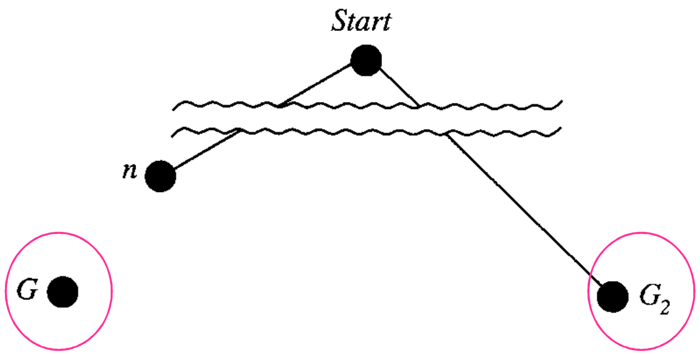
<u>Proof:</u>

Let n be an unexpanded node in the fringe such that n is <u>on</u> the optimal (shortest) path to G (there must be such a node). We will show that $f(n)<f(G2)$, i.e. n will be expanded, not G2

Optimality of A* - Proof (2)

Compare *f(G2)* and *f(G)*

1) ***f(G2)=g(G2)+h(G2) (by definition) = g(G2)* as *h(G2)=0*, *G2* is a goal**
2) ***f(G)=g(G)+h(G) (by definition) = g(G)* as *h(G)=0*, *G* is a goal**
3) ***g(G2)>g(G)* as *G2* is suboptimal**
4) ***<u>=> f(G2)>f(G)</u>* by substituting 1) and 2) into 3)**



Optimality of A* - Proof (3)

Compare $f(n)$ and $f(G)$

5) $f(n)=g(n)+h(n)$ (by definition)

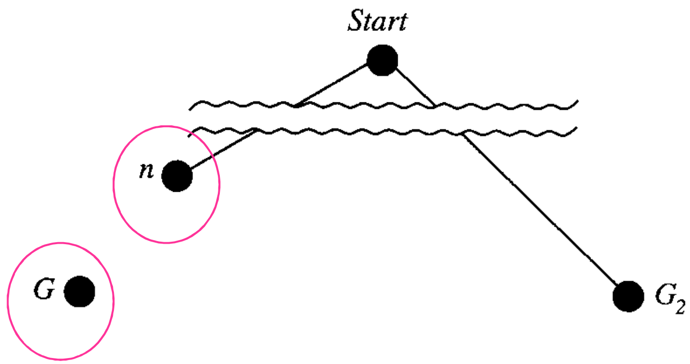
6) $h(n) <= h^*(n)$ where $h^*(n)$ is the true cost from n to G (as h is admissible)

7) => $f(n)<=g(n) + h^*(n)$ (5 & 6)

8) = g(G) path cost from S to G via n

9) $g(G) = f(G)$ as $f(G)=g(G)+h(G)=g(G)+0$ as h(G)=0, G is a goal

10) => $f(n)<=f(G)$ (7,8,9)



Thus $f(G) < f(G2)$ (4)

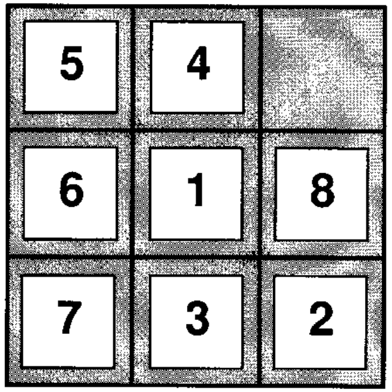
$f(n)<=f(G)$ (10)

11) $f(n)<=f(G)<f(G2)$ (10, 4)

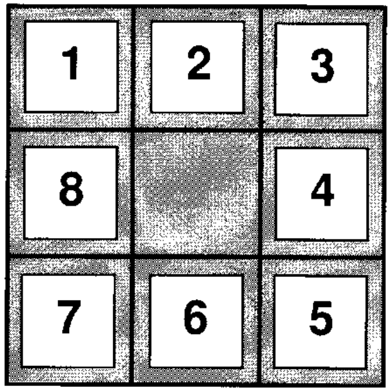
12) $f(n)<f(G2)$ => n will be expanded not $G2$; A* will not select G2 for expansion

Admissible Heuristics for 8-puzzle – h1

- ***$h_1(n)$*** **= number of misplaced tiles**
- ***$h_1(Start)$*** **= ?**
  - **7 (7 of 8 tiles are out of position)**
- **Why is** ***h_1*** **admissible?**
  - **recall: admissible heuristics are optimistic – they never overestimate the number of steps to the goal**
  - ***h_1:*** **any tile that is out of place must be moved once**
  - **true cost: higher; any tile that is out of place must be moved *at least* once**



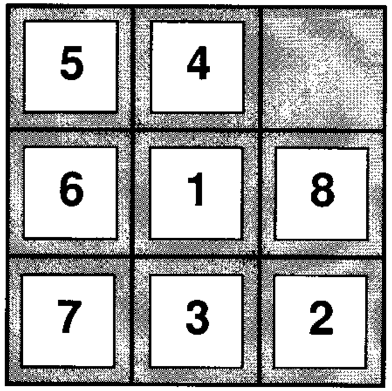
Start State



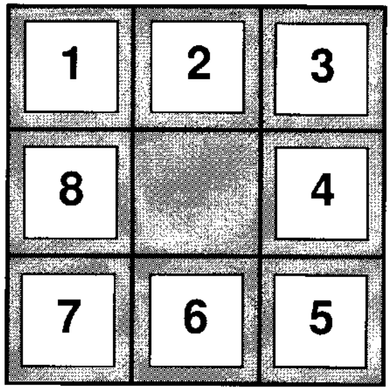
Goal State

Admissible Heuristics for 8-puzzle – h2

- $h_2(n)$ = the sum of the distances of the tiles from their goal positions (Manhattan distance)
  - note: tiles can move only horizontally and vertically
- $h_2(Start)$ = ?
  - 18 (2+3+3+2+4+2+0+2)
- Why is h_2 admissible?
  - *h2:* at each step move a tile to an adjacent position so that it is 1 step closer to its goal position and you will reach the solution in h2 steps, e.g. move tile 1 up, then left
  - True cost: higher as moving a tile to an adjacent position is not always possible; depends on the position of the blank tile



Start State



Goal State

Dominance

- **Definition of a *dominant heuristic*:**
  - **Given 2 admissible heuristics h_1 and h_2 ,**
  - **h_2 *dominates* h_1 if for all nodes n $h_2(n) \geq h_1(n)$**
- **Theorem: A* using h_2 will expand fewer nodes than A* using h_1 (i.e. h_2 is better for search)**
  - **$\forall$ *n* with f(n) < f* will be expanded (f*=cost of optimal solution path)**
  - **=> $\forall$ *n* with h(n) < f*- g(n) will be expanded**
  - **but $h_2(n) \geq h_1(n)$**
  - **=> $\forall$ n expanded by A*using h_2 will also be expanded by h_1 and h_1 may also expand other nodes**

- **Typical search costs for 8-puzzle with d=14:**

  IDS = 3 473 941 nodes, A*(h1) = 539 nodes, A*(h2) = 113 nodes
- **Dominant heuristics give a better estimate of the true cost to a goal G**

Question

- **Suppose that *h1* and *h2* are two admissible heuristics for a given problem. We define two other heuristics:**
  - ***h3=min(h1, h2)***
  - ***h4= max(h1, h2)***
- **Q1. Is *h3* admissible?**
- **Q2. Is *h4* admissible?**
- **Q3. Which one is a better heuristic - *h3* or *h4*?**

Answer

- **Suppose that *hl* and *h2* are two admissible heuristics for a given problem. We define two other heuristics:**
  - ***h3=min(hl, h2)***
  - ***h4= max(hl, h2)***
- **Q1. Is *h3* admissible?**
- **Q2. Is *h4* admissible?**
- **Q2. Which one is a better heuristic - *h3* or *h4*?**

Answer:

- **Q1 and Q2: Both *h3* and *h4* are admissible as their values are never greater than an admissible value *h1* or *h2***
- **Q3: *h4* is a better heuristic since it is closer to the real cost, i.e. *h4* is a dominant heuristic since $h4(n) \geq h3(b)$**

How to Invent Admissible Heuristics?

- **By formulating a *relaxed* version of the problem and finding the *exact* solution. This solution is an admissible heuristic.**
- **Relaxed problem – a problem with fewer restrictions on the actions**
- **8-puzzle relaxed formulation 1:**
  - **a tile can move *anywhere***
  - **How many steps do we need to reach the goal state from the initial state? (=solution)**
  - **solution = the number of misplaced tiles = $h1(n)$**
- **8-puzzle relaxed formulation 2:**
  - **a tile can move to *any adjacent square***
  - **solution = Manhattan distance = $h_2(n)$**

Admissible Heuristics from Relaxed Problems

- **<u>Theorem:</u> The optimal solution to a relaxed problem is an admissible heuristic for the original problem**
- **Intuitively, this is true because:**

  The optimal solution to the original problem is also a solution to the relaxed version (by definition) => it must be at least as expensive as the optimal solution to the relaxed version => the solution to the relaxed version is less or equally expensive than the solution to the original problem => it is an admissible heuristic for the original problem

Constructing Relaxed Problems Automatically

- **Relaxed problems can be constructed automatically if the problem definition is written in a formal language**
  - **Problem:**

    A tile can move from square A to square B if

    A is adjacent to B and B is blank
  - **3 relaxed problems generated by removing 1 or both conditions:**

    1) A tile can move from square A to square B if A is adjacent to B

    2) A tile can move from square A to square B if B is blank

    3) A tile can move from square A to square B **(always, no conditions)**
- **ABSOLVER (1993) is a program that can generate heuristics automatically using the "relaxed problem" method and other methods**
  - **Generated a new heuristic for the 8-puzzle that was better than any existing heuristic**
  - **Found the first useful heuristic for the Rubik's cube puzzle**

No Single Clearly Best Heuristic?

- **Often we can't find a single heuristic that is clearly the best (i.e. dominant)**
- **We have a set of heuristics *h1*, *h2*, …, *hm* but none of them dominates any of the others**
- **Which should we choose?**

- **Solution: define a composite heuristic:**

  h(n)=max{h1(n), h2(n), …, hm(n)}

  At a given node, it uses whichever heuristic is most accurate (dominant)
- **Is *h(n)* admissible?**

  Yes, because the individual heuristics are admissible

Learning Heuristics from Experience

- **Example: 8-puzzle**
- **Experience = many 8-puzzle solutions (paths from A to B)**
- **Each previous solution provides a set of examples to learn *h***
- **Each example is a pair (state, associated *h*)**
  - ***h* is known for each state, i.e. we have a *labelled* dataset**
- **The state is suitably represented as a set of useful features, e.g.**
  - ***f1* = number of misplaced tiles**
  - ***f2* = number of adjacent tiles that should not be adjacent**
  - ***h* is a function of the features but we don't know how exactly it depends on them, we will learn this relationship from the data**
- **We can generate e.g. 100 random 8-puzzle configurations and record the values of *f1*, *f2* and *h* to form a *training set* of examples. Using this training set, we build a classifier.**

training data

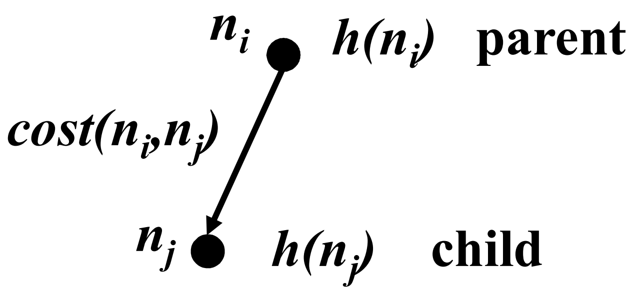
Ex.#	f1	f2	h
Ex1	7	8	14
…			
Ex100	5	2	5

- **We use this classifier on new data, i.e. given *f1* and *f2*, to predict *h* which is unknown. No guarantee that the learned heuristic is admissible or consistent.**

Back to A* and another property of the heuristics…

Consistent (Monotonic) Heuristic

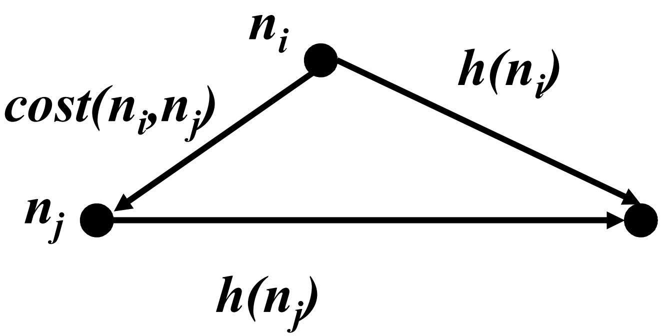
- Consider a pair of nodes *ni* and *nj,* where *ni* is the parent of *nj*

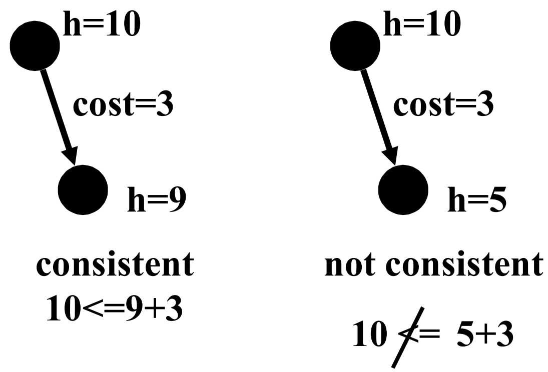


- *h* is a *consistent (monotonic) heuristic*, if for all such pairs in the search graph the following triangle inequality is satisfied:

$h(ni) \leq cost(ni,nj) + h(nj)$ for all n

parent child



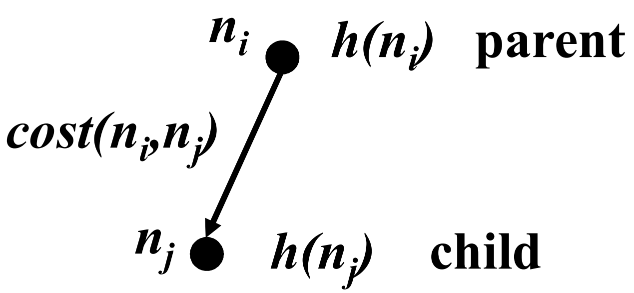


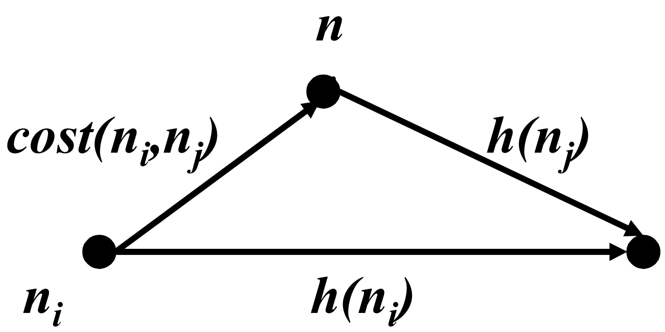
Another Interpretation of the Triangle Inequality

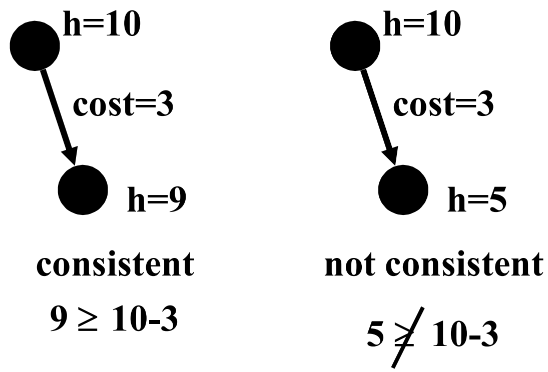
$h(ni) \leq cost(ni,nj) + h(nj)$ **for all** n

parent **child**

- $\Rightarrow h(n_j) \geq h(n_i) - cost(n_i,n_j)$ **, i.e. along any path our estimate of the remaining cost to the goal cannot decrease by more than the arc cost**







Consistency Theorems

- **Theorem 1: If *h(n)* is consistent, then *f(nj)* $\geq$ *f(ni)*, i.e. *f* is non-decreasing along any path**

  child parent

Given: h(ni) $\leq$ c(ni, nj)+h(nj)

To prove: f(nj) $\geq$ f(ni)

Proof: f(nj)=g(nj)+h(hj)=

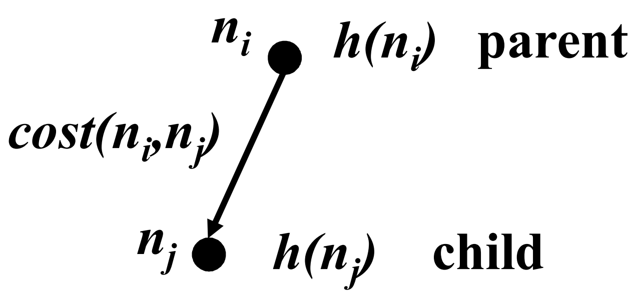
=g(ni)+c(ni,nj)+h(nj)=

$\geq$ g(ni) + h(ni) =

=f(ni)

=>f(nj) $\geq$ f(ni)

deff. h(n) consistent



- **Theorem 2: If *f(nj)* $\geq$ *f(ni)*, i.e. *f* is non-decreasing along any path, then *h(n)* is consistent**

Admissibility and Consistency

- **Consistency is the stronger condition**
- **Theorems:**
  - **If a heuristic is consistent, it is also admissible**

    consistent => admissible

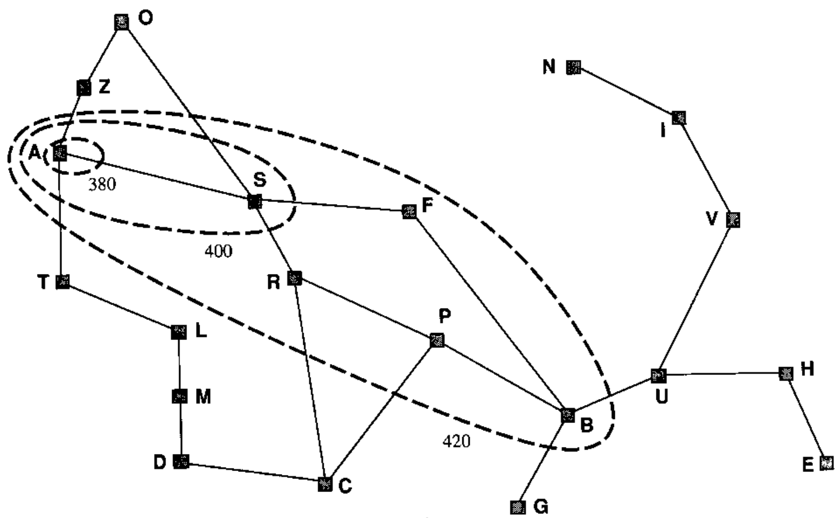
  - **If a heuristic is admissible, there is no guarantee that it is consistent**

    admissible ≠> consistent

Completeness of A* with Consistent Heuristic – Intuitive Idea

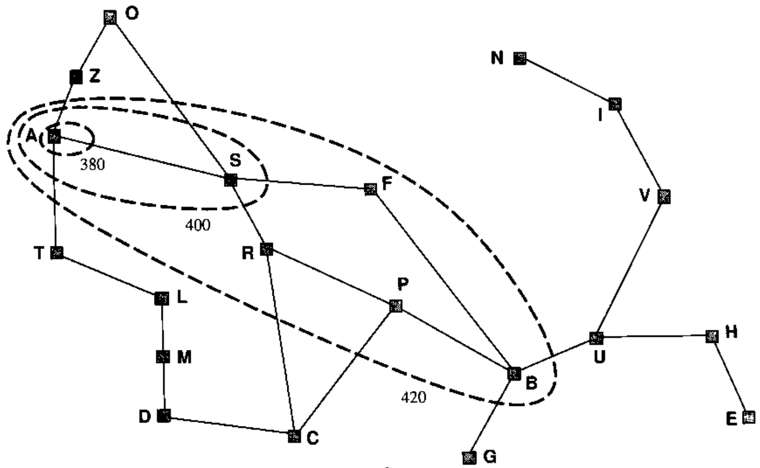
- A* uses the f-cost to select nodes for expansion
- If *h* is consistent, the f-costs are non-decreasing => we can draw f-contours in the state space
- A* expands nodes in order of increasing f-values, i.e.
  - It gradually adds f-contours of nodes
  - Nodes inside a contour have f-cost less than or equal to the contour value

- Completeness – as we add bands of increasing f, we must eventually reach a band where f=h(G)+g(G)=h(G)



<u>Optimality</u> of A* with Consistent Heuristic – Intuitive Idea

- **A* finds the optimal solution, i.e. the one with smallest path cost g(n) among all solutions**
- **The first solution must be the optimal one, as subsequent contours will have higher f-cost, and thus higher g-cost (h(n)=0 for goal nodes):**
  - **Bands f1<f2<f3…**
  - **Compare 2 solutions at band 2 and 3: G2 and G3 (G2 will be found first)**
  - **f(G2)=g(G2)+h(G2), f(G3)=g(G3)+h(G3)**
  - **But f(G2)<f(G3) and h(G2)=h(G3)=0 => g(G2)<g(G3), i.e. the first solution found is the optimal**



A* with Consistent Heuristic is Optimally Efficient

- **Theorem: If *h* is a consistent heuristic, then A* is *optimally efficient* among all optimal search algorithms using *h***
  - **no other optimal algorithm using *h* is guaranteed to expand fewer nodes than A***

- **Which are the optimal algorithms we have studied so far?**
- **Which are the optimal heuristic algorithms we have studied so far?**

Properties of A*

- <u>Complete?</u> Yes, unless there are infinitely many nodes with $f \leq f(G)$, G – optimal goal state
- <u>Optimal?</u> Yes, with admissible heuristic
- <u>Time?</u> Exponential $O(b^d)$
- <u>Space?</u> Exponential, keeps all nodes in memory

- For most problems, the number of nodes which have to be expanded is exponential
- Both time and space are problems for A* but space is the bigger problem - A* runs out of space long before it runs out of time; solution: Iterative Deepening A* (IDA*) or Simplified Memory-Bounded A* (SMA*)

Summary of A*

- **An admissible heuristic never overestimates the true distance to a goal**
- **A consistent (monotonic) heuristic satisfies the triangle equation**

- ***h(n)* satisfies the triangle equation <=> *f(n)* does not decrease along any path**

- **Admissible ≠> consistent**
- **Consistent => admissible**

- **Dominant heuristic**
  - **given 2 admissible heuristics *h*1 and *h2*, *h2* is dominant if it gives a better estimate of the true cost to a goal node**
  - **A* with a dominant heuristic will expand fewer nodes**

Summary of A* (2)

- **If *h(n)* is admissible, A* is optimal**
- **If *h(n)* is consistent, A* is optimally efficient - A* will expand less or equal number of nodes than any other optimal algorithm using *h(n)***

- **However, theoretical completeness and optimality do not mean practical completeness and optimality if it takes too long to get the solution (time and space are exponential)**
- **=> If we can't design an accurate admissible or consistent heuristic, it may be better to settle for a <u>non-admissible</u> heuristic that works well in practice or for a local search algorithm (next lecture) even though completeness and optimality are no longer guaranteed.**
- **=> Also, although dominant (i.e. good) heuristics are better, they may need a lot of time to compute; it may be better to use a simpler heuristic - more nodes will be expanded but overall the search may be faster.**