SIM GLOBAL EDUCATION

UNIVERSITY OF WOLLONGONG AUSTRALIA

CSCI203 - Algorithms and Data Structures

Greedy Algorithm (Part 2)

Sionggo Japit
sjapit@uow.edu.au

2 January 2023

Outline

- A* Search

Objective

Upon completion of this topic, you should:

- Have an understanding of what a A* search algorithm is.
- Be able to use it to solve various types of problems.

Greedy Algorithms
A* Search

A* Search

- A* search is the combination of Dijkstra’s shortest path algorithm and best first search.
- It is a searching algorithm that makes use of heuristics function to help its searching.
- It can be used to solve various type of problems.
- A* search finds the shortest path, for example, through a search space to goal state using heuristics function.

A* Search

- This technique finds minimal cost solutions and is directed to a goal state called A* search.
- The A* algorithm also find the lowest cost path between the start and goal state, where changing from one state to another requires some cost.
- A* algorithm requires heuristics function to evaluate the cost of path that passes through the particular state.

A* Search

- The heuristics function is complete if the branching factor is finite and every action has fixed cost.
- For example, in TSP, one possible way to determine heuristics is in each city h(n) is the distance to the closest city not yet visited.

A* Search

- The function can be defined by the following formula:

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

where

$g(n)$: $The\ actual\ cost\ path\ from\ the\ start\ state\ to\ the\ current\ state.$

$h(n)$: $The\ actual\ cost\ path\ from\ the\ current\ state\ to\ goal\ state. We\ also\ call\ this\ a\ heuristic\ cost.$

$f(n)$: $The\ actual\ cost\ path\ from\ the\ start\ state\ to\ the\ goal\ state.$

A* Algorithm

1. **Initialize:** set OPEN=[s], CLOSED=[], g(s)=0, f(s)=h(s)
2. **Fail:** if OPEN=[], then terminate and fail
3. **Select:** Select a state with minimum cost n from OPEN and save in CLOSED.
4. **Terminate:** if $n \in G$ then terminate with success and return $f(s)$.
5. **Expand:** for each successors m of n,

   For each successor m, insert m in OPEN only if

   $$m \notin [\text{OPEN} \cup \text{CLOSED}]$$

   Set $g(m) = g(n) + C[n, m]$

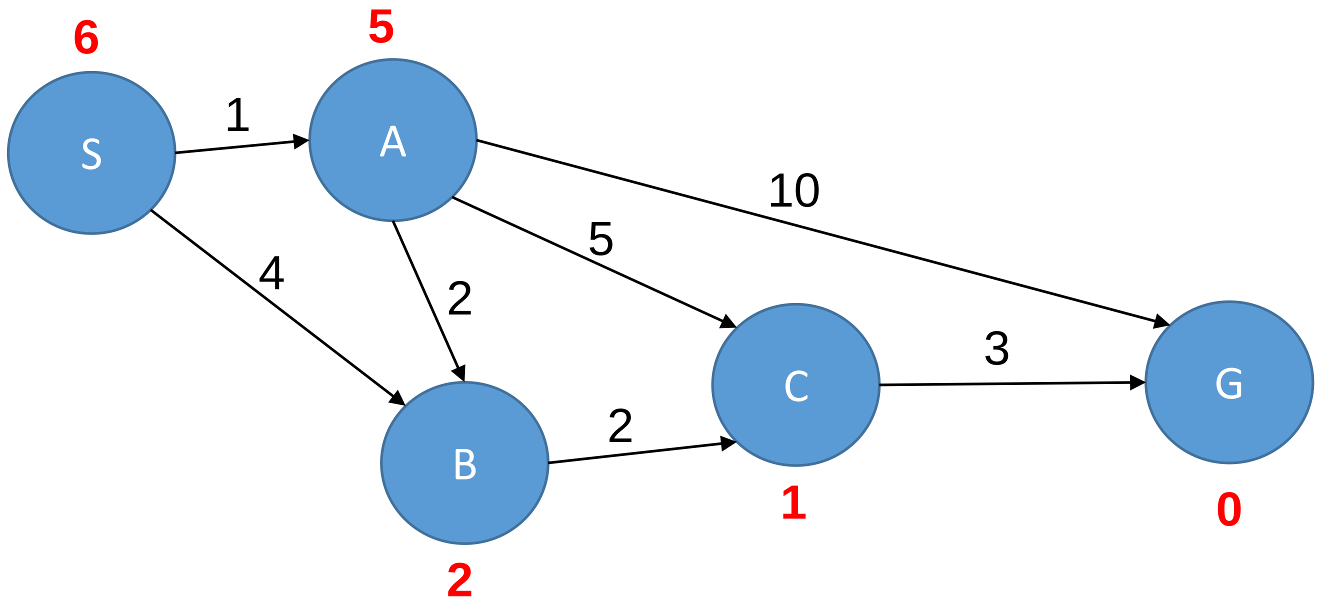
   Set $f(m) = g(m) + h(n)$

   If $f[m]$ has decreased and $m \in \text{CLOSED}$ move m to OPEN.
6. **Loop:** go to step 2

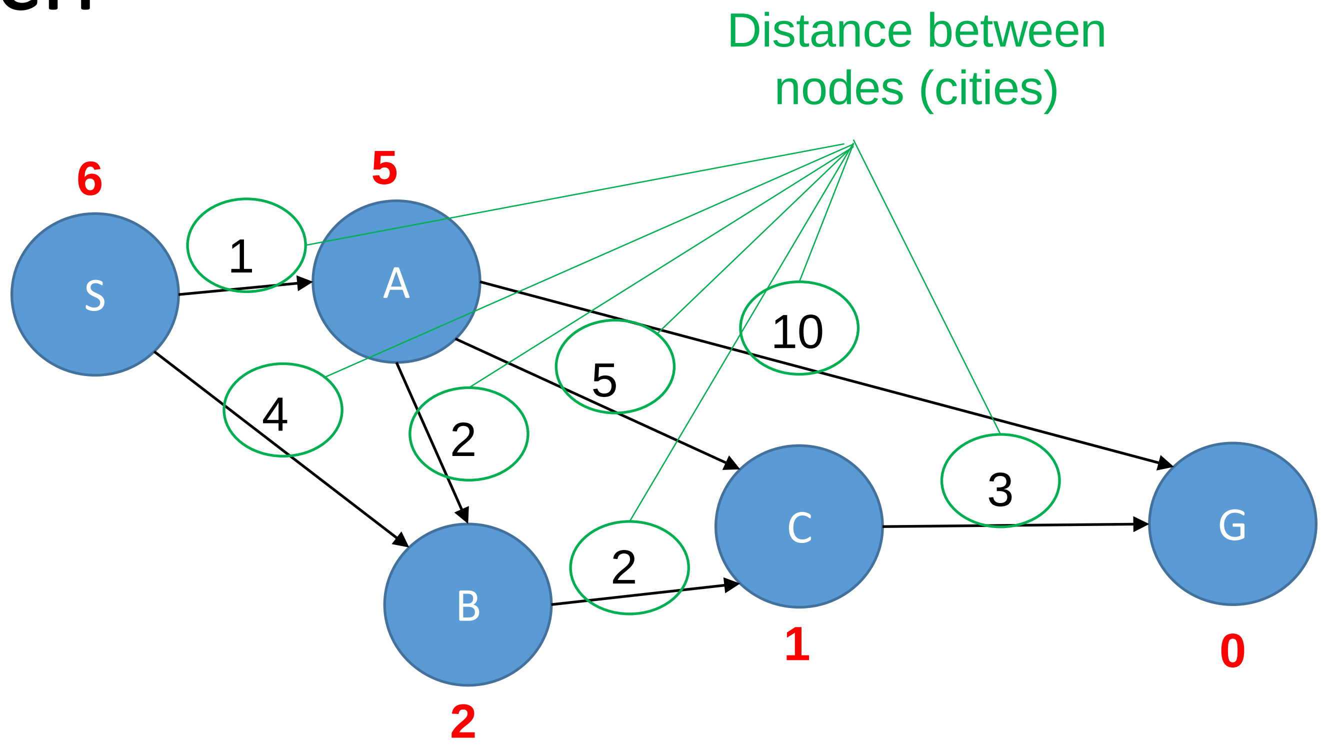
A* Search

Example:

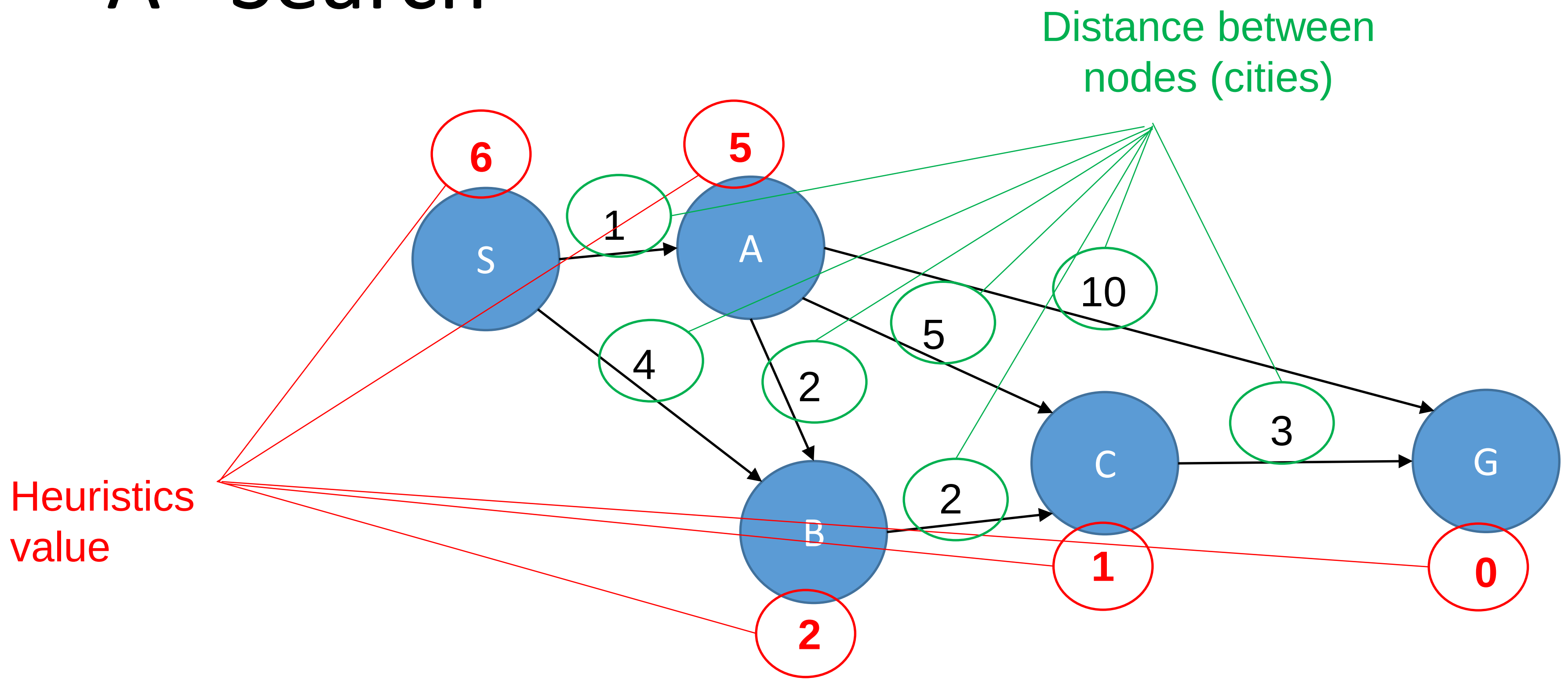
Consider a graph with multiple nodes, representing a network of cities that are inter-linked together. Starting from city 'S' we want to reach the targeted city 'G' with the shortest distance.



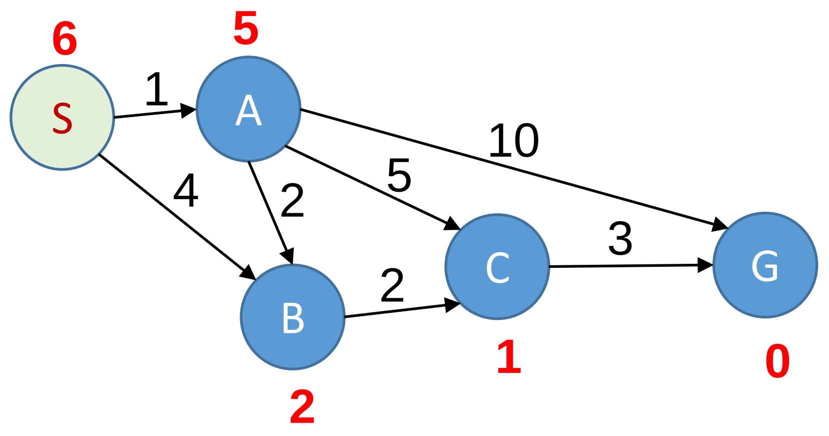
A* Search



A* Search


Distance between nodes (cities)
Heuristics value
S
A
B
C
G
6
5
2
1
0
1
4
2
5
10
2
3

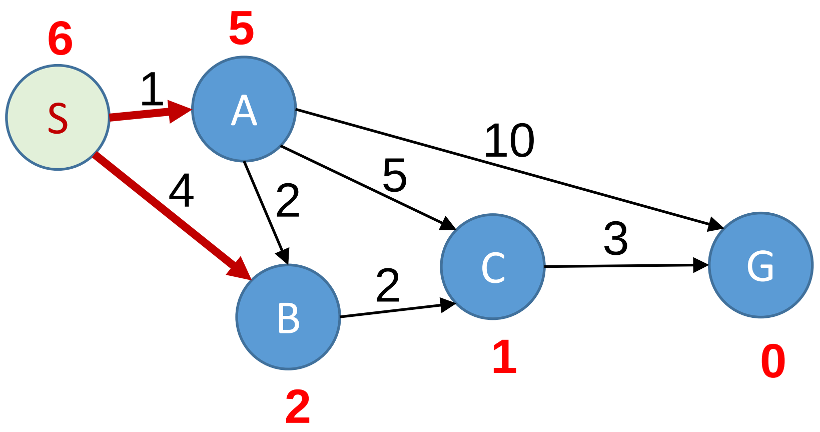
A* Search



Close set: { }

$$\textbf{S}, \mathrm{f}(S) = \mathrm{g}(\emptyset) + h(S)$$
$$= (0) + 6$$
$$= 6$$

A* Search



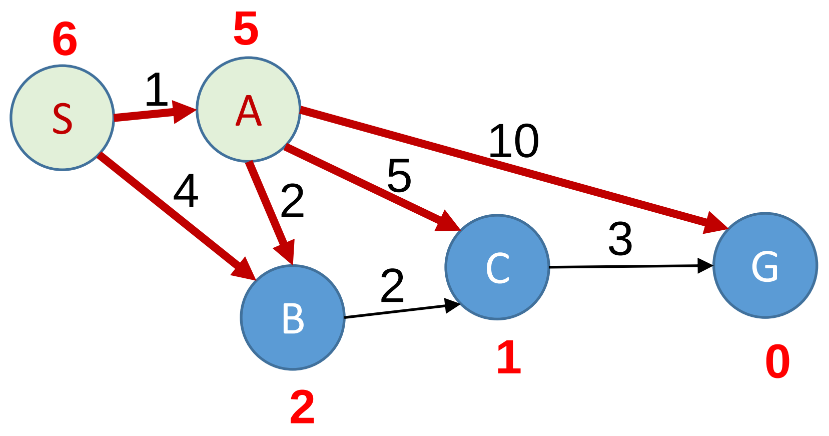
Close set: {S, }

$\mathbf{S}, \mathrm{f(S)} = \mathrm{g}(\emptyset) + h(S)$
$= (0) + 6$
$= 6$

$\mathbf{S - A}, \mathrm{f(A)} = \mathrm{g}(SA) + h(A)$
$= (0 + 1) + 5$
$= 6$

$\mathbf{S - B}, \mathrm{f(B)} = \mathrm{g}(SB) + h(B)$
$= (0 + 4) + 2$
$= 6$

A* Search



Close set: {S, A }

$\mathbf{S}, \mathrm{f(S)} = \mathrm{g}(\emptyset) + h(S)$
$= (0) + 6$
$= 6$

$\mathbf{S-A}, \mathrm{f(A)} = \mathrm{g}(SA) + h(A)$
$= (0+1) + 5$
$= 6$

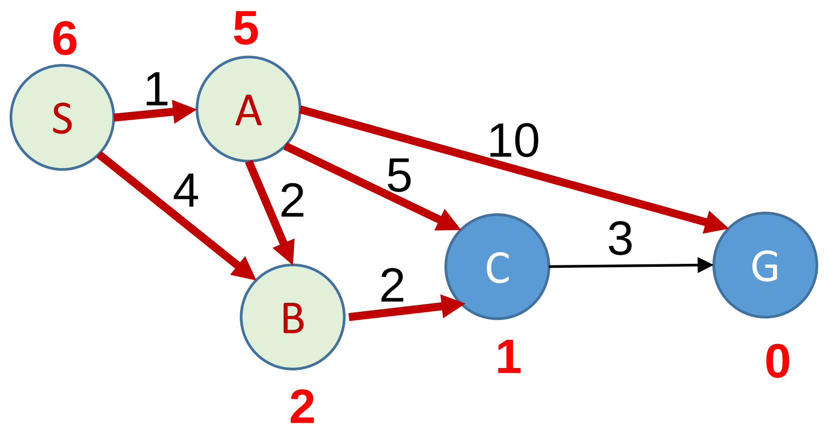
$\mathbf{S-B}, \mathrm{f(B)} = \mathrm{g}(SB) + h(B)$
$= (0+4) + 2$
$= 6$

$\mathbf{SA} - \mathrm{B}, \mathrm{f(B)} = \mathrm{g}(\mathrm{S}AB) + h(B)$
$= (1+2) + 2$
$= 5$

$\mathbf{SA-C}, \mathrm{f(C)} = \mathrm{g}(SAC) + h(C)$
$= (1+5) + 1$
$= 7$

$\mathbf{SA} - \mathrm{G}, \mathrm{f(G)} = \mathrm{g(SAG)} + h(G)$
$= (1+10) + 0$
$= 11$

A* Search

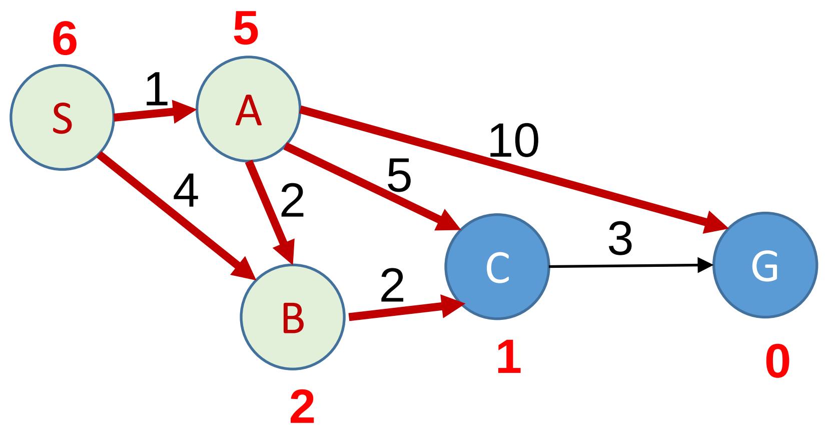


Close set: {S, A, B }

$$\textbf{SA} - \text{B}, \mathrm{f}(\mathrm{B}) = \mathrm{g}(SAB) + h(B)$$
$$= (1 + 2) + 2$$
$$= 5$$

$$\textbf{SAB} - \textbf{C}, \mathrm{f}(\mathrm{C}) = \mathrm{g}(\mathrm{SA}BC) + h(C)$$
$$= (3 + 2) + 1$$
$$= 6$$

A* Search



Close set: {S, A, B }

$$\mathbf{SA} - \mathrm{B}, \mathrm{f(B)} = \mathrm{g}(SAB) + h(B) \\ = (1+2)+2 \\ = 5$$

There are now four search-tree nodes that are Not processed yet, they are SAB-C, S-B, SA-C, and SA-G.

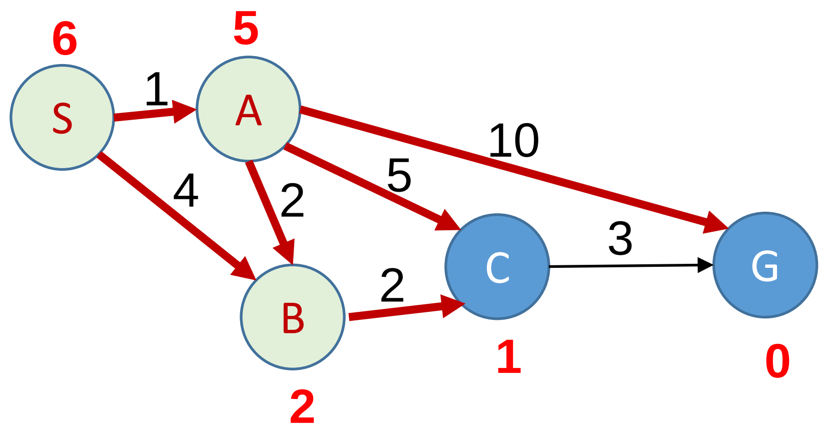
$$\mathbf{SAB} - \mathbf{C}, \mathrm{f(C)} = \mathrm{g}(\mathrm{SAB}C) + h(C) \\ = (3+2)+1 \\ = 6$$

$$\mathbf{S} - \mathbf{B}, \mathrm{f(B)} = \mathrm{g}(SB) + h(B) \\ = (0+4)+2 \\ = 6$$

$$\mathbf{SA} - \mathbf{C}, \mathrm{f(C)} = \mathrm{g}(SAC) + h(C) \\ = (1+5)+1 \\ = 7$$

$$\mathbf{SA} - \mathrm{G}, \mathrm{f(G)} = \mathrm{g(SAG)} + h(G) \\ = (1+10)+0 \\ = 11$$

A* Search



Close set: {S, A, B }

$$\mathbf{SA} - \mathrm{B}, \mathrm{f(B)} = \mathrm{g}(SAB) + h(B) \\ = (1+2)+2 \\ = 5$$

There are now four search-tree nodes that are Not processed yet, they are SAB-C, S-B, SA-C, and SA-G.

$$\mathbf{SAB} - \mathbf{C}, \mathrm{f(C)} = \mathrm{g}(\mathrm{SAB}C) + h(C) \\ = (3+2)+1 \\ = 6$$

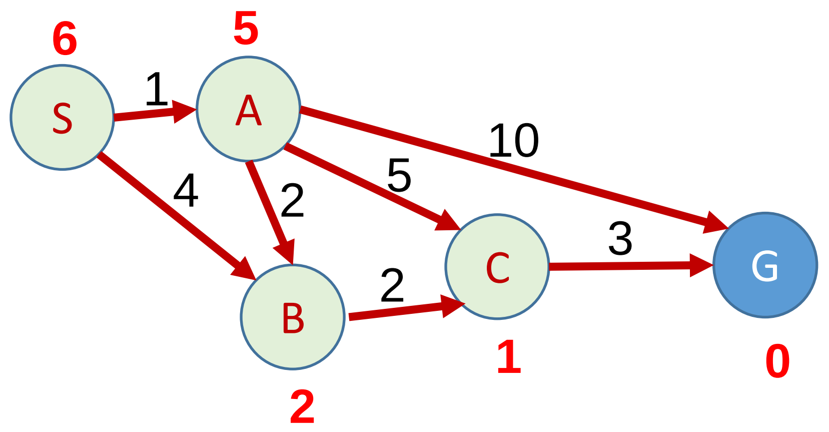
$$\mathbf{S} - \mathbf{B}, \mathrm{f(B)} = \mathrm{g}(SB) + h(B) \\ = (0+4)+2 \\ = 6$$

$$\mathbf{SA} - \mathbf{C}, \mathrm{f(C)} = \mathrm{g}(SAC) + h(C) \\ = (1+5)+1 \\ = 7$$

$$\mathbf{SA} - \mathrm{G}, \mathrm{f(G)} = \mathrm{g(SAG)} + h(G) \\ = (1+10)+0 \\ = 11$$

These two search-tree nodes tie. We may use alphabetical order as tie-breaker.

A* Search



Close set: {S, A, B, C }

$\mathbf{SA - B}, f(B) = g(SAB) + h(B)$
$= (1 + 2) + 2$
$= 5$

$\mathbf{S - B}, f(B) = g(SB) + h(B)$
$= (0 + 4) + 2$
$= 6$

$\mathbf{SAB - C}, f(C) = g(SABC) + h(C)$
$= (3 + 2) + 1$
$= 6$

$\mathbf{SA - C}, f(C) = g(SAC) + h(C)$
$= (1 + 5) + 1$
$= 7$

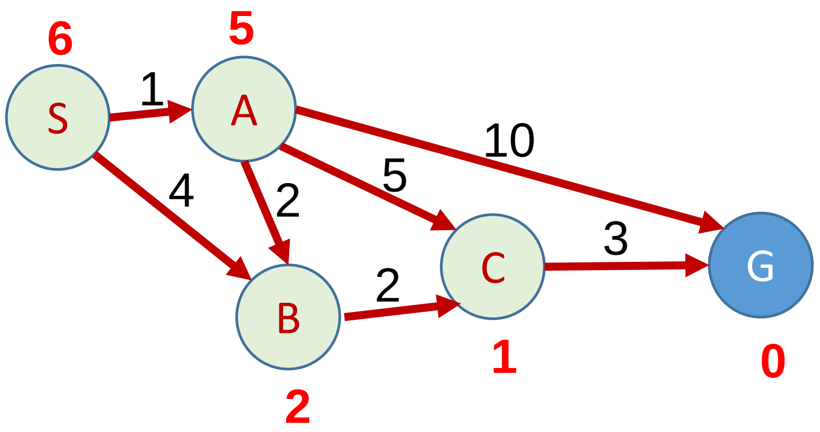
$\mathbf{SA - G}, f(G) = g(SAG) + h(G)$
$= (1 + 10) + 0$
$= 11$

$\mathbf{SABC - G}, f(G) = g(SABCG) + h(G)$
$= (5 + 3) + 0$
$= 8$

Next, the algorithm process the search-tree node S-B. But since node B is already in the ‘Close set’, the cannot be extended.

A* Search

Close set: {S, A, B, C }



$\mathbf{SA-B}, \mathrm{f(B)} = \mathrm{g}(SAB) + h(B)$
$= (1+2)+2$
$= 5$

$\mathbf{S-B}, \mathrm{f(B)} = \mathrm{g}(SB) + h(B)$
$= (0+4)+2$
$= 6$

Cannot proceed.

$\mathbf{SAB-C}, \mathrm{f(C)} = \mathrm{g}(SABC) + h(C)$
$= (3+2)+1$
$= 6$

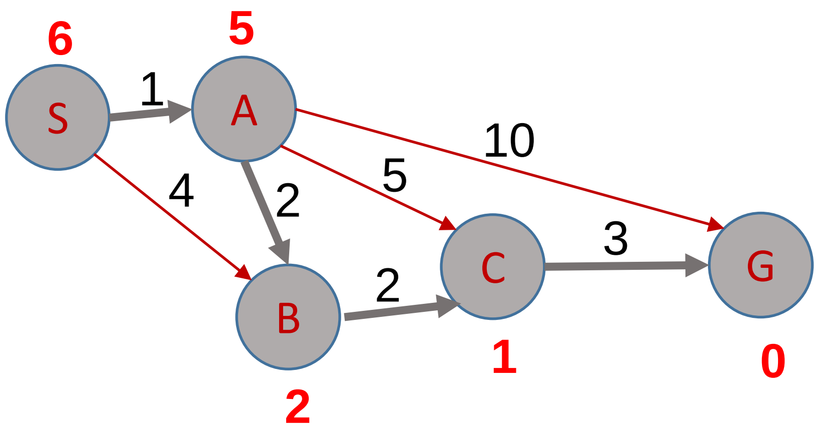
$\mathbf{SA-C}, \mathrm{f(C)} = \mathrm{g}(SAC) + h(C)$
$= (1+5)+1$
$= 7$

$\mathbf{SA-G}, \mathrm{f(G)} = \mathrm{g}(SAG) + h(G)$
$= (1+10)+0$
$= 11$

$\mathbf{SABC-G}, \mathrm{f(G)} = \mathrm{g}(SABCG) + h(G)$
$= (5+3)+0$
$= 8$

$\mathbf{SAC-G}, \mathrm{f(G)} = \mathrm{g}(SACG) + h(G)$
$= (6+3)+0$
$= 9$

A* Search



Close set: {S, A, B, C }

$\mathbf{SA-}\mathrm{B}, \mathrm{f(B)} = \mathrm{g}(SA) + h(B) = (1+2)+2 = 5$

$\mathbf{S-B}, \mathrm{f(B)} = \mathrm{g}(S) + h(B) = (0+4)+2 = 6$

Cannot proceed.

$\mathbf{SAB-C}, \mathrm{f(C)} = \mathrm{g}(\mathrm{SA}B) + h(C) = (3+2)+1 = 6$

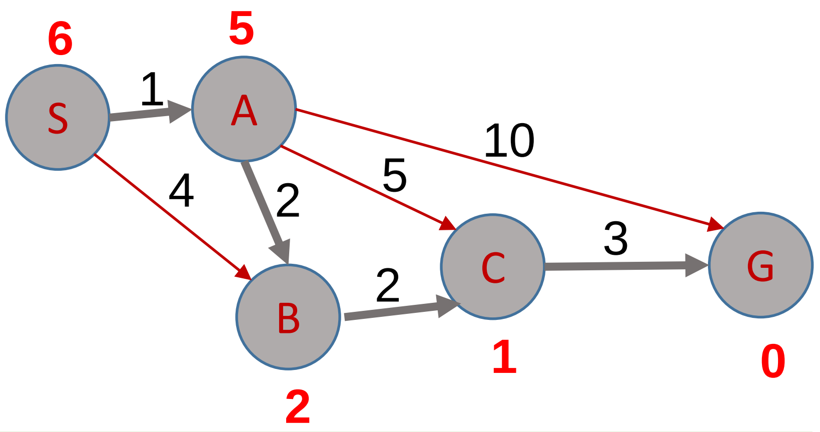
$\mathbf{SA-C}, \mathrm{f(C)} = \mathrm{g}(SA) + h(C) = (1+5)+1 = 7$

$\mathbf{SA-}\mathrm{G}, \mathrm{f(G)} = \mathrm{g}(\mathrm{SA}G) + h(G) = (1+10)+0 = 11$

$\mathbf{SABC-G}, \mathrm{f(G)} = \mathrm{g(SABCG)} + h(G) = (5+3)+0 = 8$

$\mathbf{SAC-}\mathrm{G}, \mathrm{f(G)} = \mathrm{g(SACG)} + h(G) = (6+3)+0 = 9$

A* Search



Close set: {S, A, B, C }

SA – I

ot
ed.

SAB – (

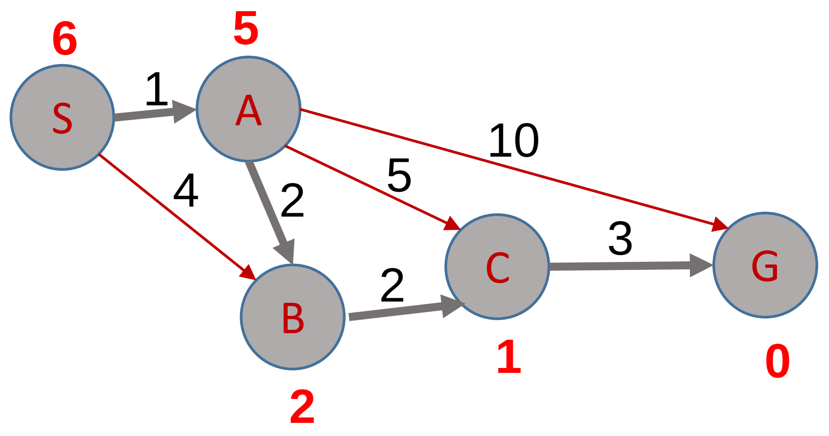
The three search-tree nodes are terminated at the goal state 'G', no further expansion can be done because the 'goal' has been achieved. Hence the shortest path starting from node 'S' to 'G' is SABCG, at a cost of 8.

$) + h(G)$
$= (1 + 10) + 0$
$= 11$

SABC – G, $f(G) = g(SABCG) + h(G)$
$= (5 + 3) + 0$
$= 8$

SAC – G, $f(G) = g(SACG) + h(G)$
$= (6 + 3) + 0$
$= 9$

A* Search



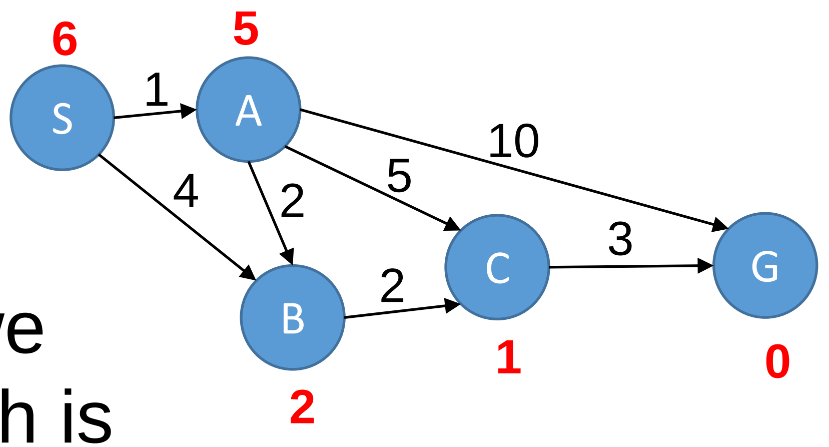
Is the solution SABCG at a cost of 8 optimal?

A* search algorithm guarantees to return an optimal solution if the heuristics h is admissible.

A* Search

- What is admissible mean?
- A* search algorithm is admissible under the following conditions:
  - Every node has a finite number of successors.
  - Every arc in the graph has a cost greater than some $\varepsilon > 0$.
  - For every node n, $\forall(s, s'): h(s) - h(s') \leq C(s, s')$.

A* Search

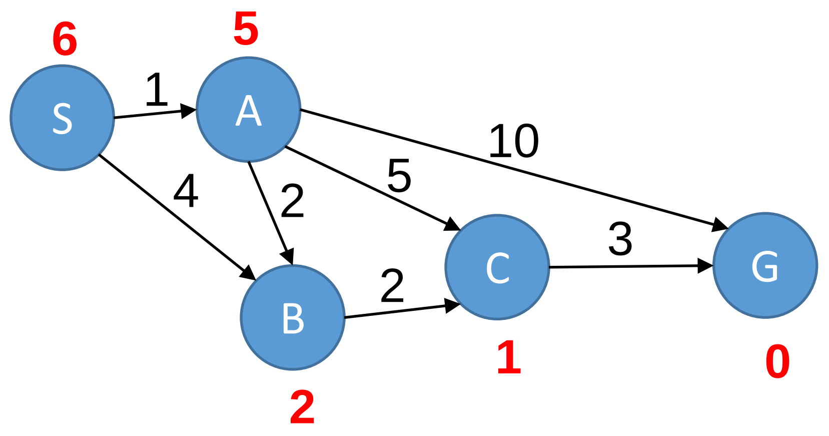


- In fact, the example we have just gone through is not admissible.

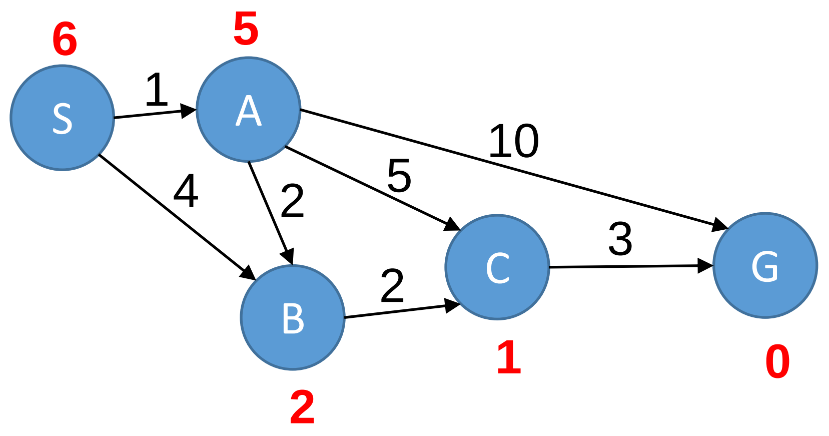
- Notice from the graph shown, $h(A) = 5$, and $h(B) = 2$. The cost $C(A, B) = 2$. Hence $h(A) - h(B) = 5 - 2 = 3$.
- 3 is not ≤ 2, the cost $C(A, B)$.

A* Search

What happen if it is not admissible?



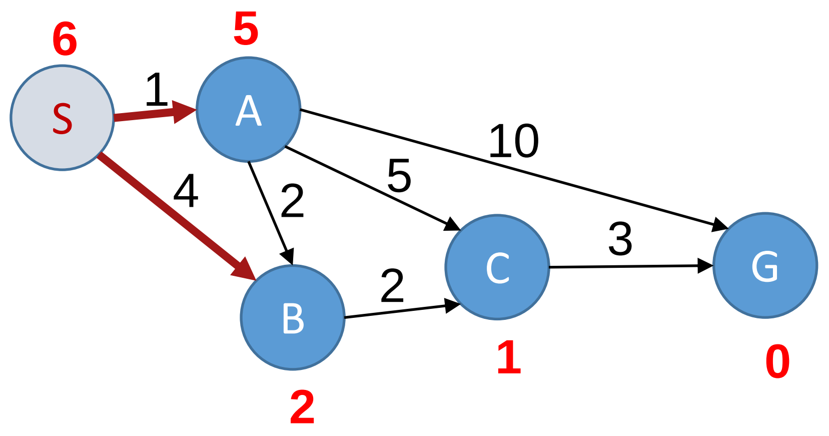
A* Search



What happen if it is not admissible?

A* search algorithm does not guarantee to return optimal solution if the heuristics is not admissible.

A* Search



For example, in our previous working of A* search, we have a 'tie' at this point:

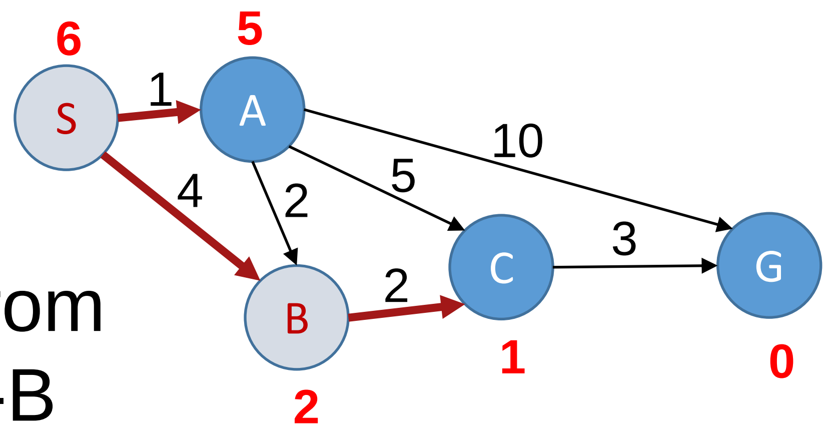
$$\mathbf{S}, \mathrm{f}(\mathrm{S}) = \mathrm{g}(\emptyset) + h(S) = (0) + 6 = 6$$

Close set: {S, }

$$\mathbf{S-A}, \mathrm{f}(\mathrm{A}) = \mathrm{g}(SA) + h(A) = (0+1) + 5 = 6$$

$$\mathbf{S-B}, \mathrm{f}(\mathrm{B}) = \mathrm{g}(SB) + h(B) = (0+4) + 2 = 6$$

A* Search



If we were to proceed from the search-tree node S-B instead of S-A, we have:

$\mathbf{S}, \mathrm{f}(\mathrm{S}) = \mathrm{g}(\emptyset) + h(S)$
$= (0) + 6$
$= 6$

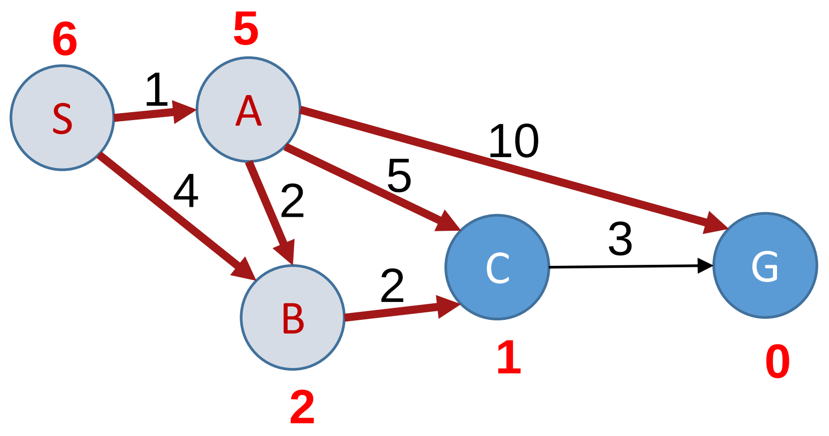
Close set: {S,B }

$\mathbf{S-A}, \mathrm{f}(\mathrm{A}) = \mathrm{g}(SA) + h(A)$
$= (0+1) + 5$
$= 6$

$\mathbf{S-B}, \mathrm{f}(\mathrm{B}) = \mathrm{g}(SB) + h(B)$
$= (0+4) + 2$
$= 6$

$\mathbf{SB} - \mathrm{C}, \mathrm{f}(\mathrm{B}) = \mathrm{g}(SBC) + h(B)$
$= (4+2) + 1$
$= 7$

A* Search



$\mathbf{S}, \mathrm{f}(\mathrm{S}) = \mathrm{g}(\emptyset) + h(S) = (0) + 6 = 6$

Close set: {S,B }

$\mathbf{S - A}, \mathrm{f}(\mathrm{A}) = \mathrm{g}(SA) + h(A) = (0 + 1) + 5 = 6$

$\mathbf{S - B}, \mathrm{f}(\mathrm{B}) = \mathrm{g}(SB) + h(B) = (0 + 4) + 2 = 6$

$\mathbf{SA - B}$

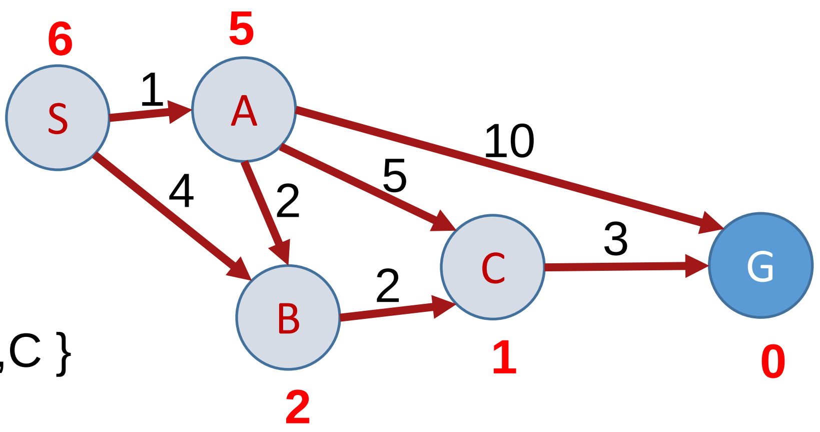
Cannot proceed.

$\mathbf{SA - C}, \mathrm{f}(\mathrm{C}) = \mathrm{g}(SAC) + h(C) = (1 + 5) + 1 = 7$

$\mathbf{SB} - \mathrm{C}, \mathrm{f}(\mathrm{B}) = \mathrm{g}(SBC) + h(B) = (4 + 2) + 1 = 7$

$\mathbf{SA} - \mathrm{G}, \mathrm{f}(\mathrm{G}) = \mathrm{g}(SAG) + h(G) = (1 + 10) + 0 = 11$

A* Search



Close set: {S,B,C }

$\mathbf{S}, \mathrm{f(S)} = \mathrm{g}(\emptyset) + h(S)$
$= (0) + 6$
$= 6$

$\mathbf{S-A}, \mathrm{f(A)} = \mathrm{g}(SA) + h(A)$
$= (0+1) + 5$
$= 6$

$\mathbf{S-B}, \mathrm{f(B)} = \mathrm{g}(SB) + h(B)$
$= (0+4) + 2$
$= 6$

$\mathbf{SB-C}, \mathrm{f(B)} = \mathrm{g}(SBC) + h(B)$
$= (4+2) + 1$
$= 7$

$\mathbf{SA-B}$

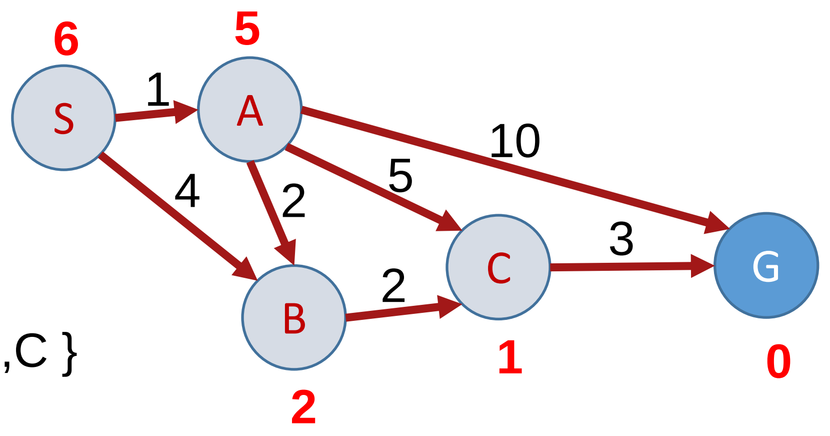
Cannot proceed.

$\mathbf{SA-C}, \mathrm{f(C)} = \mathrm{g}(SAC) + h(C)$
$= (1+5) + 1$
$= 7$

$\mathbf{SA-G}, \mathrm{f(G)} = \mathrm{g}(SAG) + h(G)$
$= (1+10) + 0$
$= 11$

$\mathbf{SBC-G}, \mathrm{f(G)} = \mathrm{g}(SBCG) + h(G)$
$= (6+3) + 0$
$= 9$

A* Search



Close set: {S,B,C }

S, $f(S) = g(\emptyset) + h(S) = (0) + 6 = 6$

S – A, $f(A) = g(S) + h(A) = (0 + 1) + 5 = 6$

S – B, $f(B) = g(S) + h(B) = (0 + 4) + 2 = 6$

SA – B

Cannot proceed.

SA – C, $f(C) = g(SA) + h(C) = (1 + 5) + 1 = 7$

SA – C

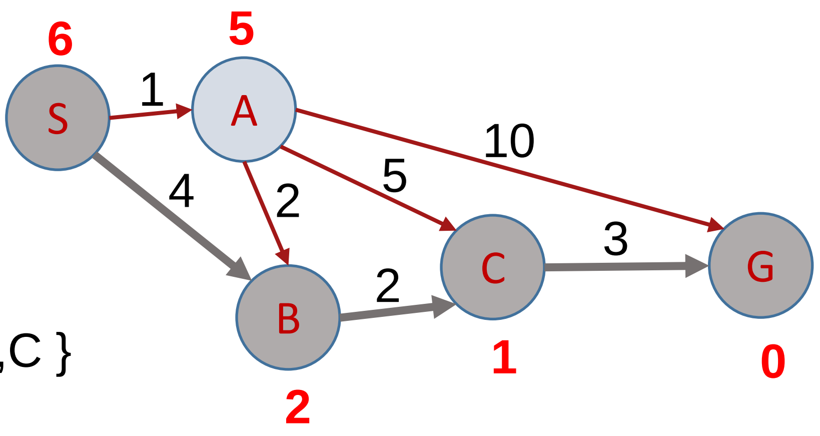
Cannot proceed.

SA – G, $f(G) = g(SA) + h(G) = (1 + 10) + 0 = 11$

SB – C, $f(B) = g(SB) + h(B) = (4 + 2) + 1 = 7$

SBC – G, $f(G) = g(SBC) + h(G) = (6 + 3) + 0 = 9$

A* Search



Close set: {S,B,C }

S, $f(S) = g(\emptyset) + h(S) = (0) + 6 = 6$

S – A, $f(A) = g(S) + h(A)$

S – B, $f(B) = g(S) + h(B) = (0 + 4) + 2$

From the candidate solutions, the path SBCG is optimal with a cost of 9.

$= (4 + 2) + 1$

$= 7$

SA – B

Cannot proceed.

SA – C, $f(C) = g(SA) + h(C) = (1 + 5) + 1 = 7$

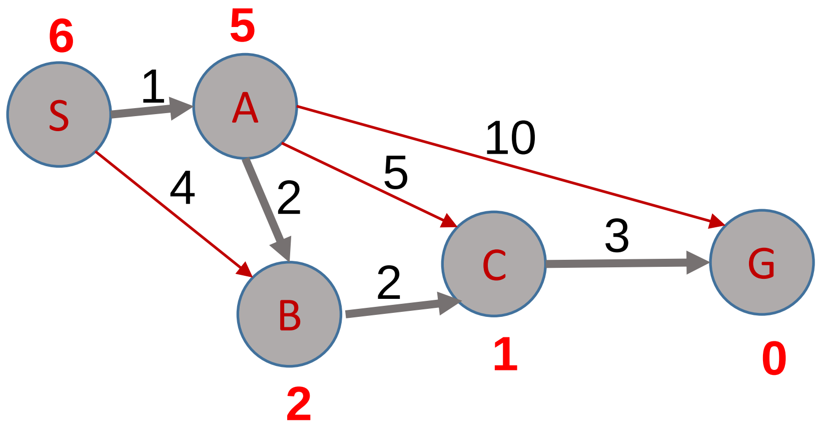
SA – C

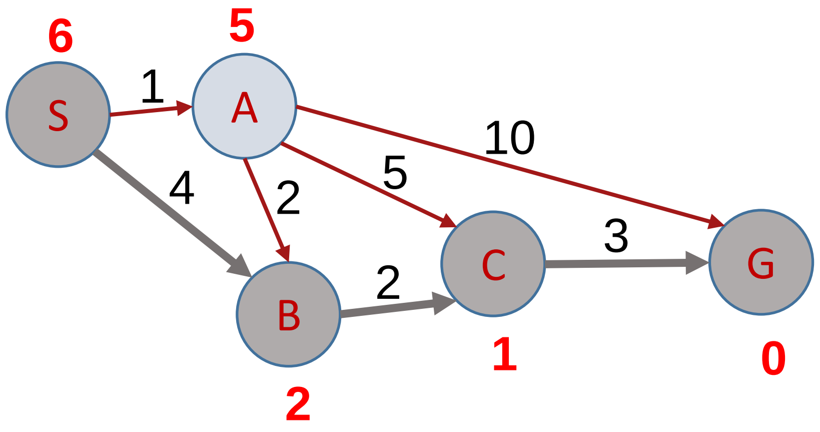
Cannot proceed.

SA – G, $f(G) = g(SA) + h(G) = (1 + 10) + 0 = 11$

SBC – G, $f(G) = g(SBC) + h(G) = (6 + 3) + 0 = 9$

A* Search

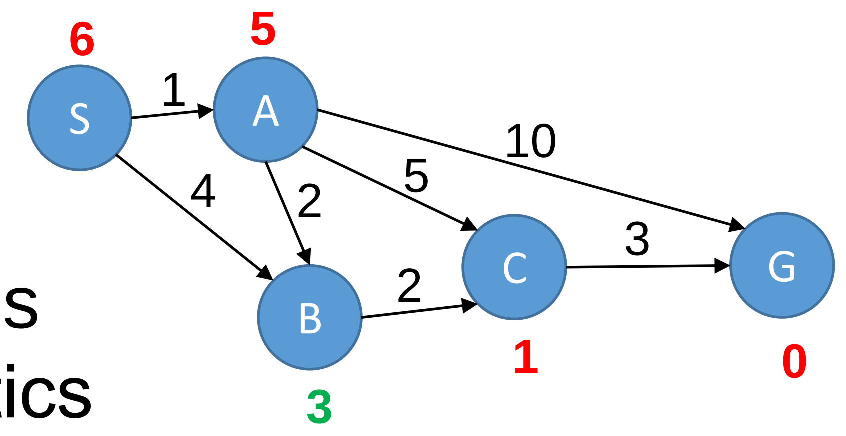




SABCG at the cost of 8. SBCG at the cost of 9.

- Huh… two different solutions.
- The reason is because the heuristics is not permissible.

A* Search

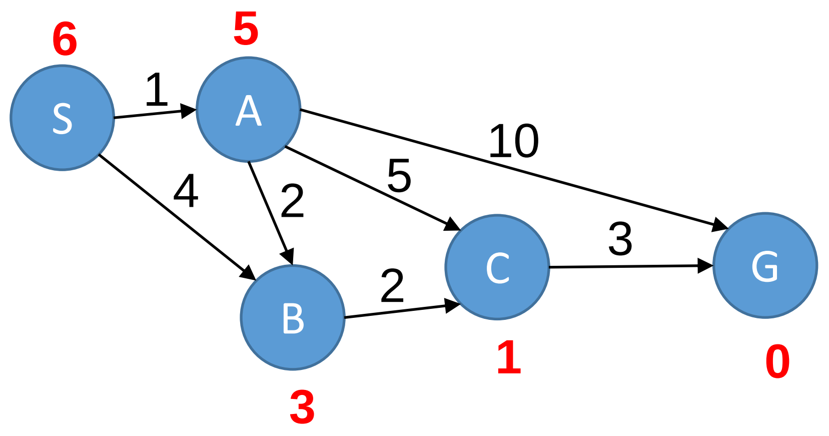


Correcting the heuristics for node ‘B’, the heuristics are now permissible.

States	$h(s) - h(s')$	$C(s,s')$	$h(s) - h(s') \leq C(s,s')$
S-A	$6 - 5 = 1$	$C(S,A) = 1$	true
S-B	$6 - 3 = 3$	$C(S,B) = 4$	true
A-B	$5 - 3 = 2$	$C(A,B) = 2$	true
A-C	$5 - 1 = 4$	$C(A,C) = 5$	true
A-G	$5 - 0 = 5$	$C(A,G) = 10$	true
B-C	$3 - 1 = 2$	$C(B,C) = 2$	true
C-G	$1 - 0 = 1$	$C(C,G) = 3$	true

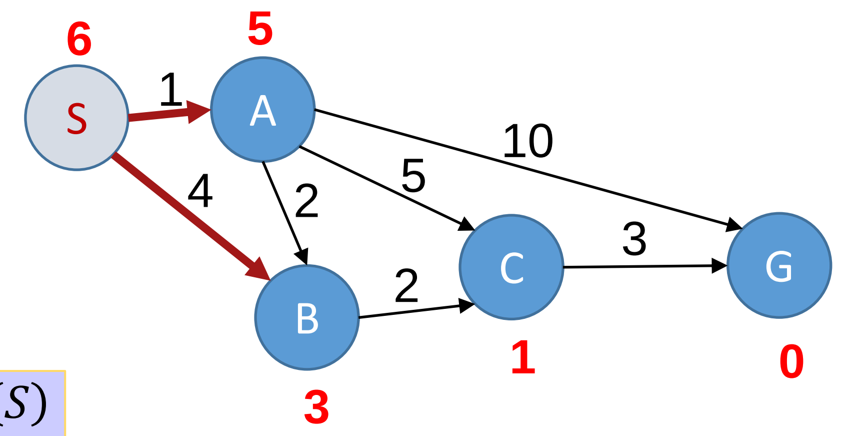
A* Search

Close set: { }



$$
\begin{aligned}
\mathbf{S}, \mathrm{f}(S) &= \mathrm{g}(\emptyset) + h(S) \\
&= (0) + 6 \\
&= 6
\end{aligned}
$$

A* Search



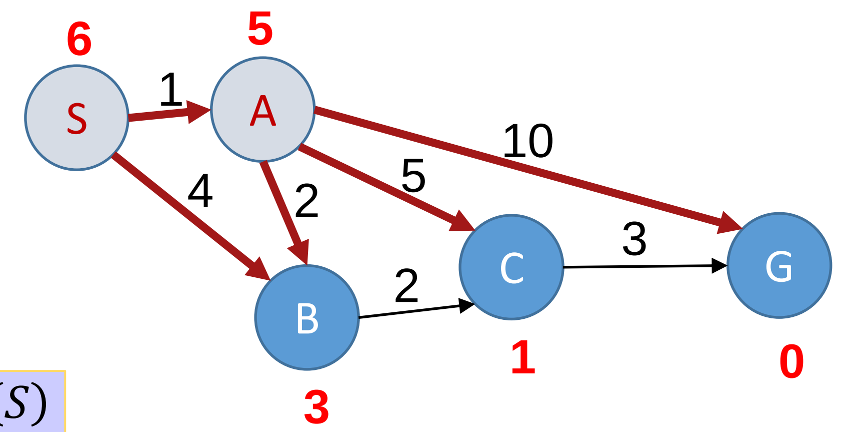
Close set: {S }

$\mathbf{S}$, $\mathrm{f(S)} = \mathrm{g}(\emptyset) + h(S)$
$= (0) + 6$
$= 6$

$\mathbf{S - A}$, $\mathrm{f(A)} = \mathrm{g}(SA) + h(A)$
$= (0 + 1) + 5$
$= 6$

$\mathbf{S - B}$, $\mathrm{f(B)} = \mathrm{g}(SB) + h(B)$
$= (0 + 4) + 3$
$= 7$

A* Search



Close set: {S, A }

$\mathbf{S}, \mathrm{f(S)} = \mathrm{g}(\emptyset) + h(S)$
$= (0) + 6$
$= 6$

$\mathbf{S - A}, \mathrm{f(A)} = \mathrm{g}(SA) + h(A)$
$= (0 + 1) + 5$
$= 6$

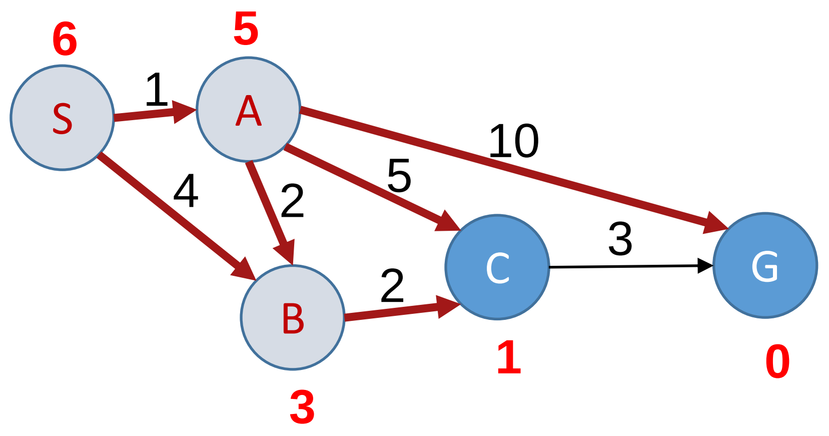
$\mathbf{S - B}, \mathrm{f(B)} = \mathrm{g}(SB) + h(B)$
$= (0 + 4) + 3$
$= 7$

$\mathbf{SA - B}, \mathrm{f(B)} = \mathrm{g}(SAB) + h(B)$
$= (1 + 2) + 3$
$= 6$

$\mathbf{SA - C}, \mathrm{f(C)} = \mathrm{g}(SAC) + h(C)$
$= (1 + 5) + 1$
$= 7$

$\mathbf{SA - G}, \mathrm{f(G)} = \mathrm{g}(SAG) + h(G)$
$= (1 + 10) + 0$
$= 11$

A* Search



Close set: {S, A, B }

$\mathbf{S - A}, \mathrm{f(A)} = \mathrm{g}(SA) + h(A)$
$= (0 + 1) + 5$
$= 6$

$\mathbf{S - B}, \mathrm{f(B)} = \mathrm{g}(SB) + h(B)$
$= (0 + 4) + 3$
$= 7$

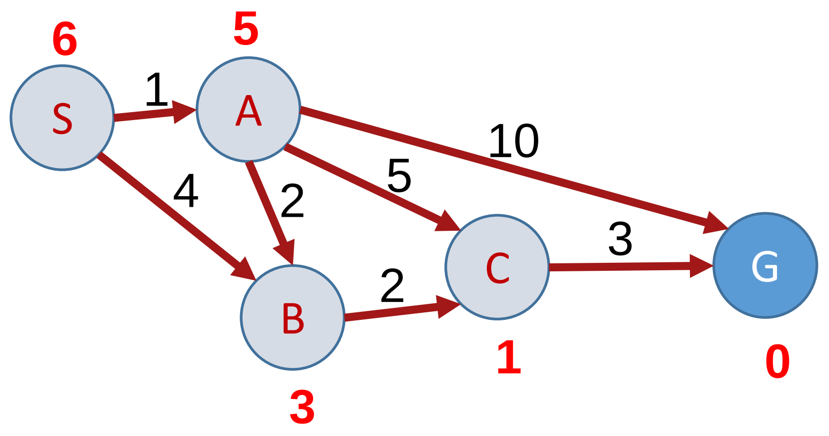
$\mathbf{SA - B}, \mathrm{f(B)} = \mathrm{g}(SAB) + h(B)$
$= (1 + 2) + 3$
$= 6$

$\mathbf{SA - C}, \mathrm{f(C)} = \mathrm{g}(SAC) + h(C)$
$= (1 + 5) + 1$
$= 7$

$\mathbf{SA - G}, \mathrm{f(G)} = \mathrm{g}(SAG) + h(G)$
$= (1 + 10) + 0$
$= 11$

$\mathbf{SAB - C}, \mathrm{f(C)} = \mathrm{g}(SABC) + h(C)$
$= (3 + 2) + 1$
$= 6$

A* Search



Close set: {S, A, B, C}

$$\mathbf{S-A}, f(A) = g(SA) + h(A) = (0+1)+5 = 6$$

$$\mathbf{S-B}, f(B) = g(SB) + h(B) = (0+4)+3 = 7$$

$$\mathbf{SA-B}, f(B) = g(SAB) + h(B) = (1+2)+3 = 6$$

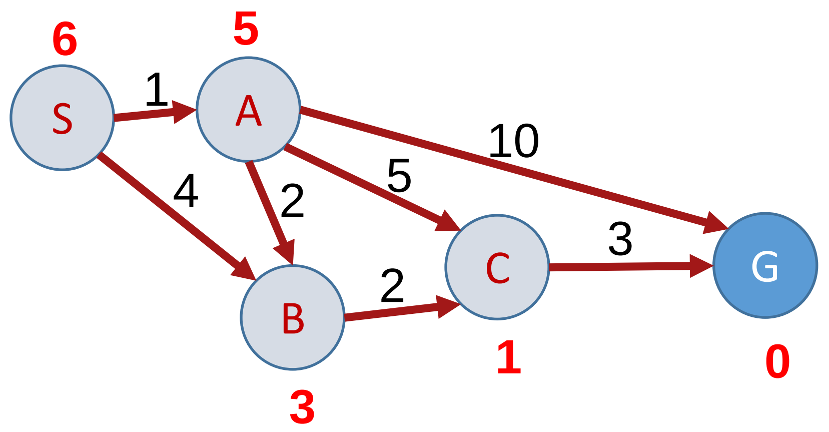
$$\mathbf{SA-C}, f(C) = g(SAC) + h(C) = (1+5)+1 = 7$$

$$\mathbf{SA-G}, f(G) = g(SAG) + h(G) = (1+10)+0 = 11$$

$$\mathbf{SAB-C}, f(C) = g(SABC) + h(C) = (3+2)+1 = 6$$

$$\mathbf{SABC-G}, f(G) = g(SABCG) + h(G) = (5+2)+1 = 8$$

A* Search



Close set: {S, A, B, C}

$\mathbf{S-A}$, $f(A) = g(SA) + h(A) = (0+1)+5 = 6$

$\mathbf{S-B}$, $f(B) = g(SB) + h(B) = (0+4)+3 = 7$

$\mathbf{SA-B}$, $f(B) = g(SAB) + h(B) = (1+2)+3 = 6$

$\mathbf{SA-C}$, $f(C) = g(SAC) + h(C) = (1+5)+1 = 7$

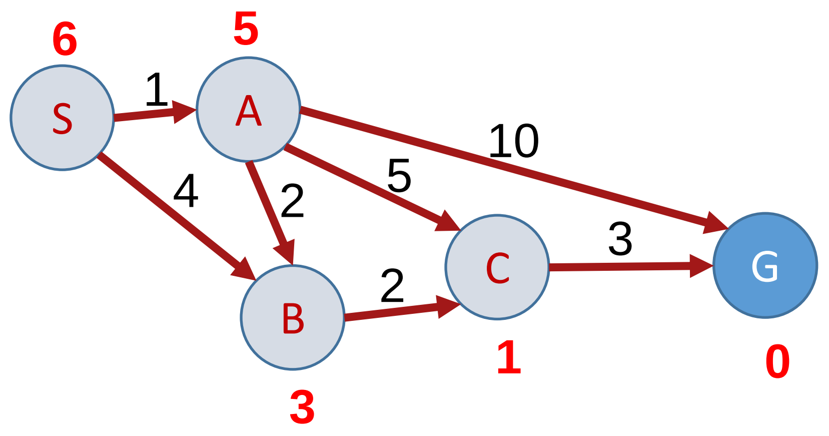
$\mathbf{SA-G}$, $f(G) = g(SAG) + h(G) = (1+10)+0 = 11$

$\mathbf{SAB-C}$, $f(C) = g(SABC) + h(C) = (3+2)+1 = 6$

$\mathbf{SAC-G}$, $f(G) = g(SACG) + h(G) = (6+3)+0 = 9$

$\mathbf{SABC-G}$, $f(G) = g(SABCG) + h(G) = (5+2)+1 = 8$

A* Search



Close set: {S, A, B, C}

$\mathbf{S-A}, \mathrm{f(A)} = \mathrm{g}(SA) + h(A) = (0+1)+5 = 6$

$\mathbf{S-B}, \mathrm{f(B)} = \mathrm{g}(SB) + h(B) = (0+4)+3 = 7$

SB-C

Cannot proceed

$\mathbf{SA-B}, \mathrm{f(B)} = \mathrm{g}(SAB) + h(B) = (1+2)+3 = 6$

$\mathbf{SA-C}, \mathrm{f(C)} = \mathrm{g}(SAC) + h(C) = (1+5)+1 = 7$

$\mathbf{SA-G}, \mathrm{f(G)} = \mathrm{g}(SAG) + h(G) = (1+10)+0 = 11$

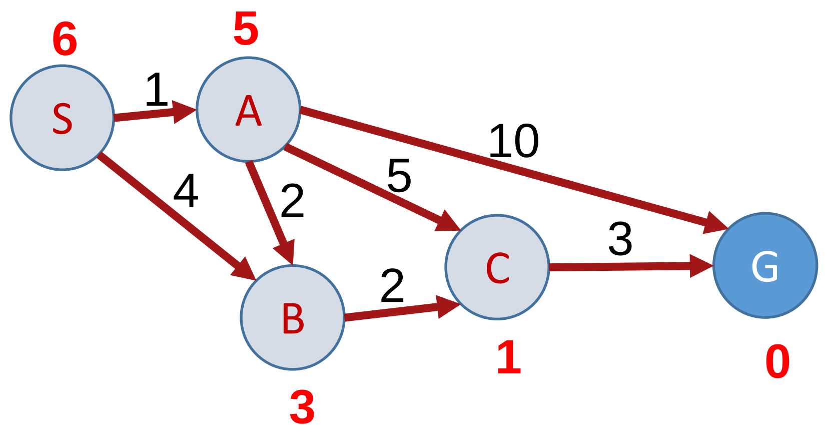
$\mathbf{SAB-C}, \mathrm{f(C)} = \mathrm{g}(SABC) + h(C) = (3+2)+1 = 6$

$\mathbf{SAC-G}, \mathrm{f(G)} = \mathrm{g}(SACG) + h(G) = (6+3)+0 = 9$

$\mathbf{SABC-G}, \mathrm{f(G)} = \mathrm{g}(SABCG) + h(G) = (5+2)+1 = 8$

A* Search

Close set: {S, A, B, C}



S − A, $f(A) = g(SA) + h(A)$
$= (0 + 1) + 5$

S − B, $f(B) = g(SB) + h(B)$
$= (0 + 4) + 3$

SB-C

Cannot

All remaining search-tree nodes are terminates at the goal 'G', hence, no need to further expand. The optimal solution is the path SABCG at the cost 8.

$= 6$

SA − C, $f(C) = g(SAC) + h(C)$
$= (1 + 5) + 1$
$= 7$

$= 11$

SAB − C, $f(C) = g(SABC) + h(C)$
$= (3 + 2) + 1$
$= 6$

SAC − G, $f(G) = g(SACG) + h(G)$
$= (6 + 3) + 0$
$= 9$

SABC − G, $f(G) = g(SABCG) + h(G)$
$= (5 + 2) + 1$
$= 8$

A* Search

Summary:

- **Admissible:** The A* search algorithm is admissible. This means that provided a solution exists, the first solution found by A* search algorithm is an optimal solution.
- A* search algorithm is admissible under the following conditions:
  - Every node has a finite number of successors.
  - Every arc in the graph has a cost greater than some $\varepsilon > 0$.
  - For every node n, $\forall(s, s')$: $h(s) - h(s') \leq C(s, s')$.

A* Search

- **Complete:** A* search algorithm is complete under the above conditions, that is, permissible.
- **Optimal:** A* search algorithm is optimally efficient for a given heuristics-of the optimal search algorithm that expand search paths from the root node, it can be shown that no other optimal algorithm will expand fewer nodes and find a solution.

A* Search

Advantages:

- It is complete and optimal.
- It is the best one from other techniques.
- It is used to solve very complex problems.
- It is optimally efficient, that is, there is no other optimal algorithm guaranteed to expand fewer nodes than A* search algorithm.

A* Search

Disadvantages:

- A* search algorithm is complete if the branching factor is finite and every action has fixed cost.
- The speed execution of A* search is highly dependent on the accuracy of the heuristic algorithm that is used to compute $h(n)$.
- It has complexity problems.