

Exercises: Artificial Intelligence

A*

A*

A* ALGORITHM

A* Algorithm

- ***Input:***

- **QUEUE:** Path only containing root

- ***Algorithm:***

- **WHILE** (QUEUE not empty && first path not reach goal) **DO**

- Remove ***first path*** from QUEUE

- Create paths to all children

- Reject paths with loops

- Add paths and sort QUEUE (by $f = \text{cost} + \text{heuristic}$)

- **IF** QUEUE contains paths: P, Q

- AND** P ends in node N_i && Q contains node N_i

- AND** cost P \geq cost Q

- THEN** remove P

- **IF** goal reached **THEN** success **ELSE** failure

A*

FIRST EXAMPLE ON A*

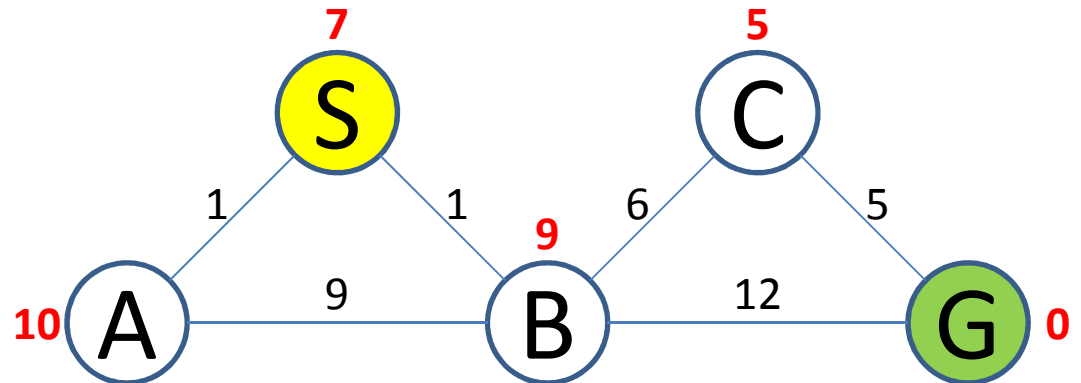
A* algorithm by Example



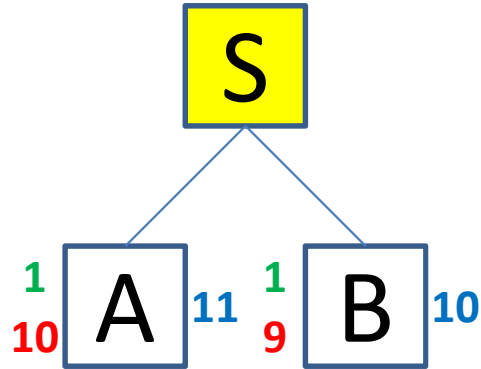
$f = \text{accumulated path cost} + \text{heuristic}$

QUEUE = *path containing root*

QUEUE: <S>



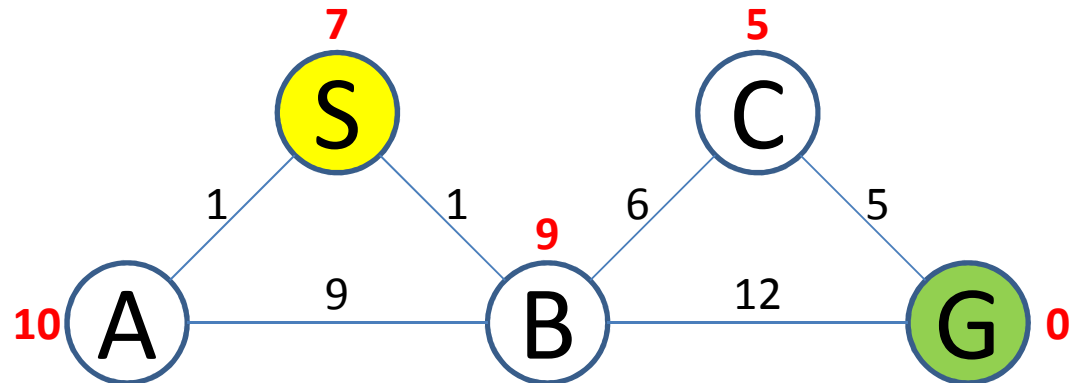
A* algorithm by Example



$f = \text{accumulated path cost} + \text{heuristic}$

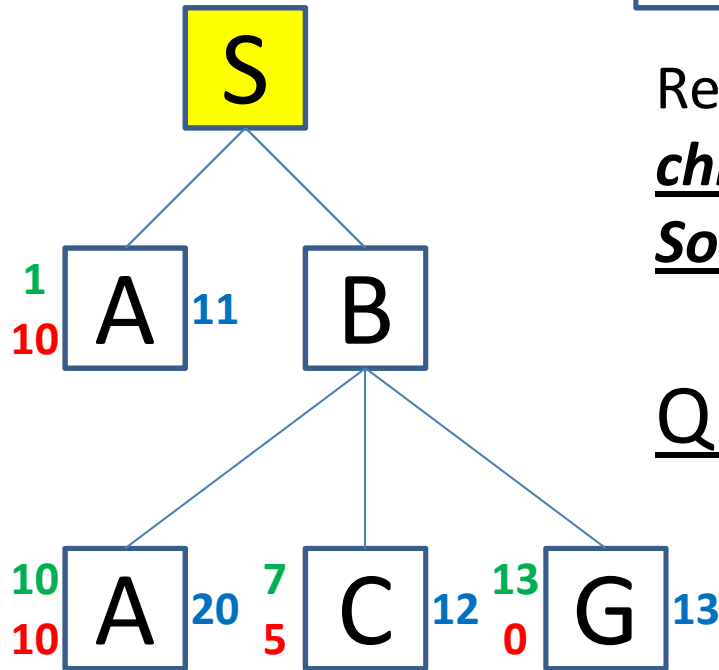
Remove ***first path***, Create ***paths to all children***, Reject ***loops*** and ***Add paths***.
Sort QUEUE by f

QUEUE: <SB,SA>



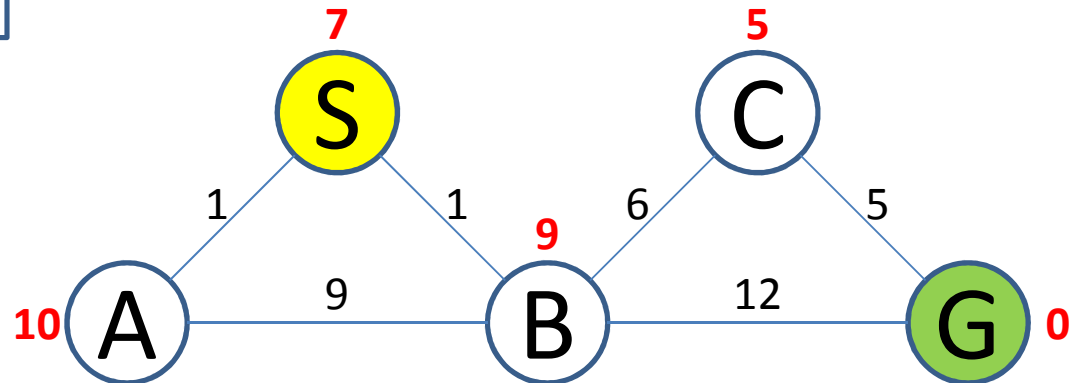
A* algorithm by Example

$$f = \text{accumulated path cost} + \text{heuristic}$$



Remove ***first path***, Create ***paths to all children***, Reject ***loops*** and ***Add paths***.
Sort QUEUE by f

QUEUE: <SA, SBC, SBG, SBA>

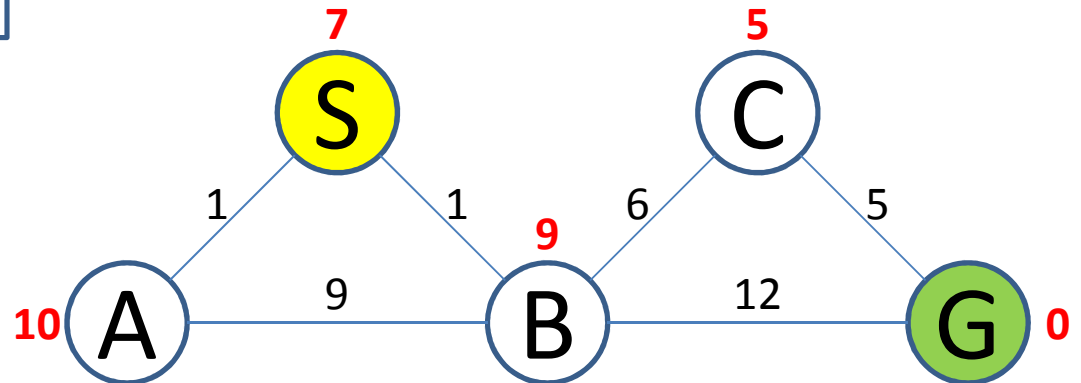
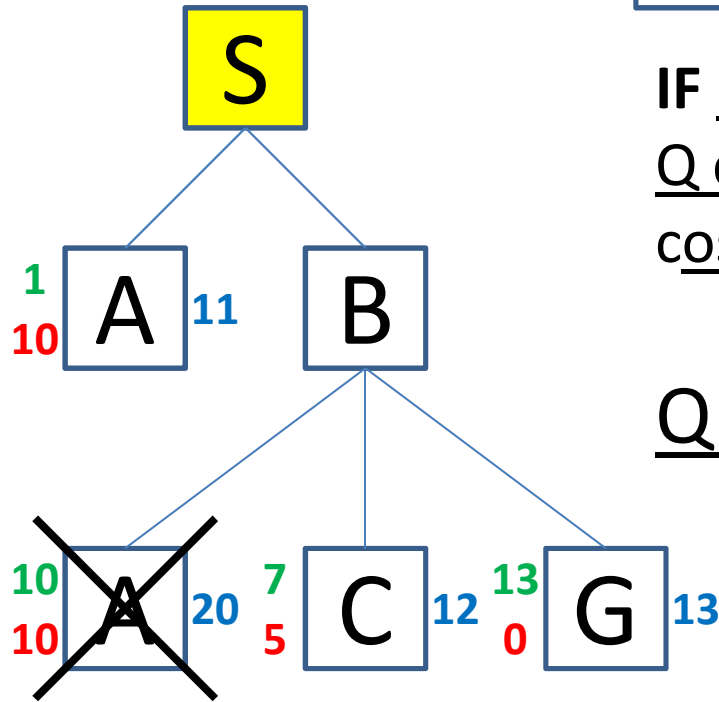


A* algorithm by Example

$f = \text{accumulated path cost} + \text{heuristic}$

IF P terminating in I with cost P &&
Q containing I with cost Q **AND**
cost P \geq cost Q **THEN** remove P

QUEUE: <SA, SBC, SBG, SBA>

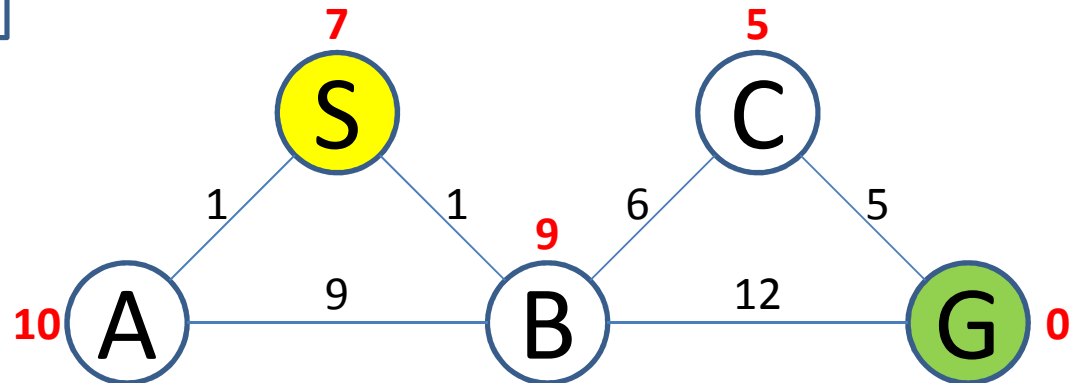
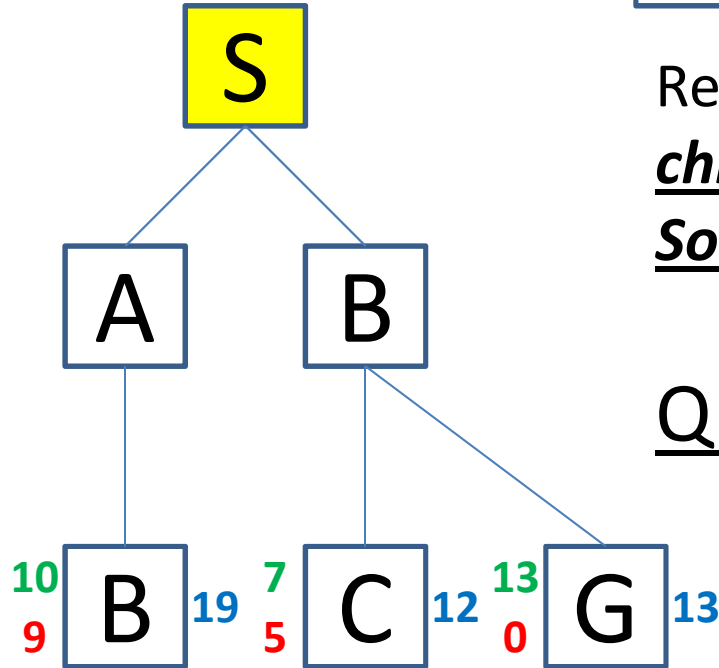


A* algorithm by Example

$f = \text{accumulated path cost} + \text{heuristic}$

Remove ***first path***, Create ***paths to all children***, Reject ***loops*** and ***Add paths***.
Sort QUEUE by f

QUEUE: <SBC,SBG,SAB>

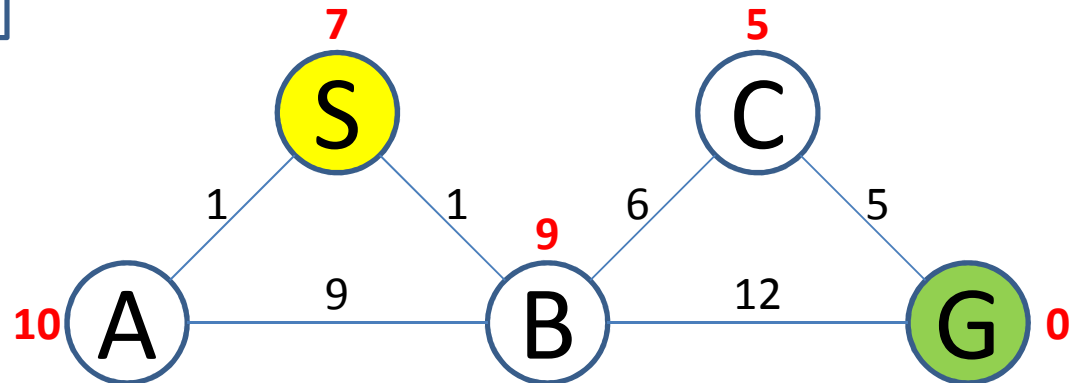
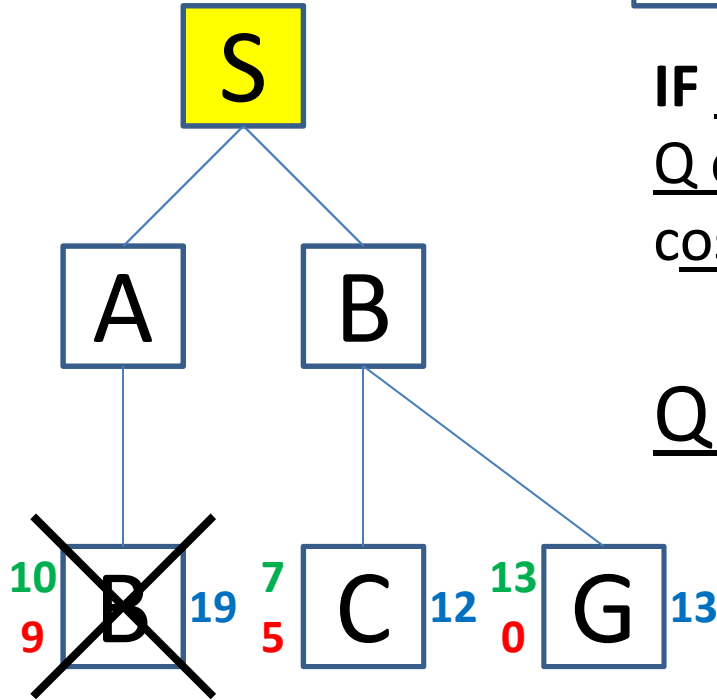


A* algorithm by Example

$$f = \text{accumulated path cost} + \text{heuristic}$$

IF P terminating in I with cost P &&
Q containing I with cost Q **AND**
cost P \geq cost Q **THEN** remove P

QUEUE: <SBC,SBG,**SAB**>

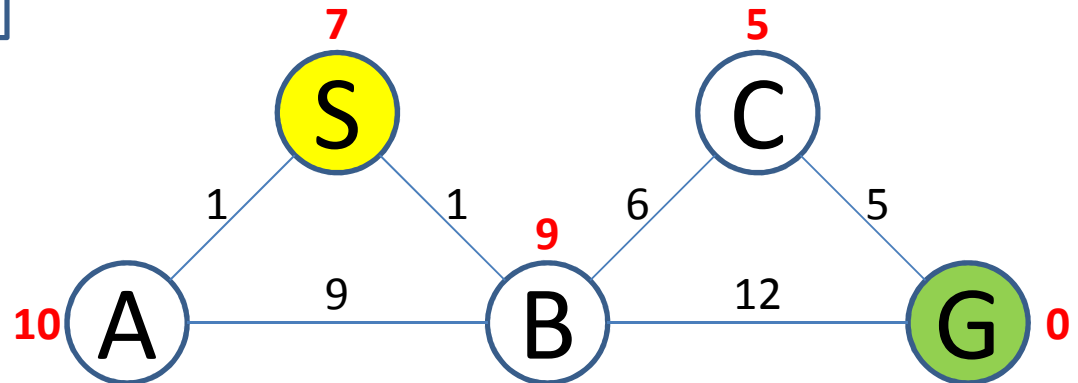
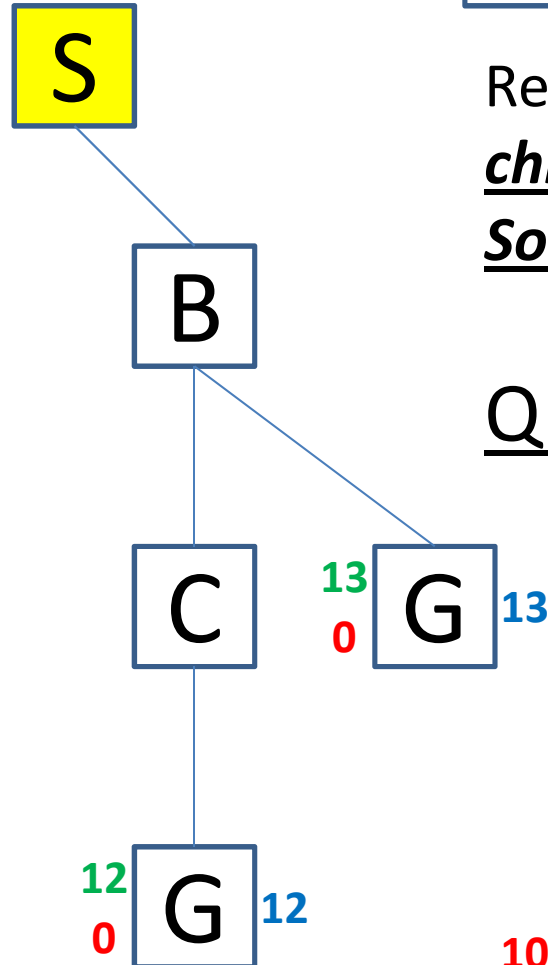


A* algorithm by Example

$f = \text{accumulated path cost} + \text{heuristic}$

Remove first path, Create paths to all children, Reject loops and Add paths.
Sort QUEUE by f

QUEUE: <SBCG, SBG>

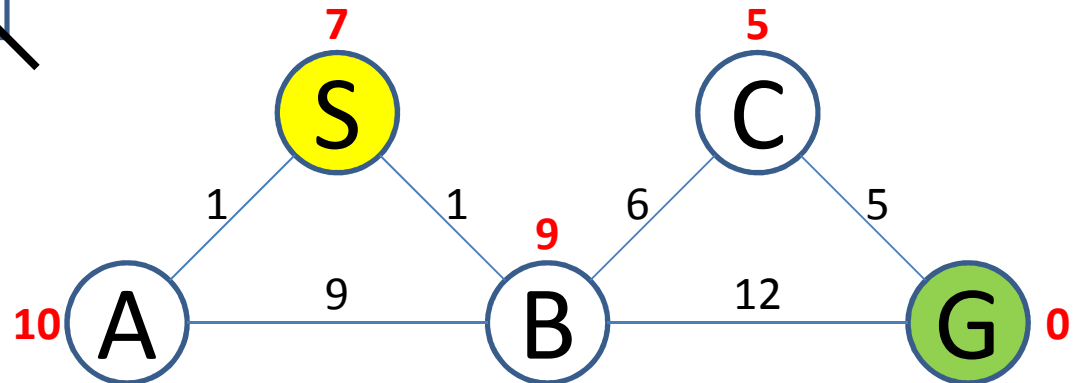
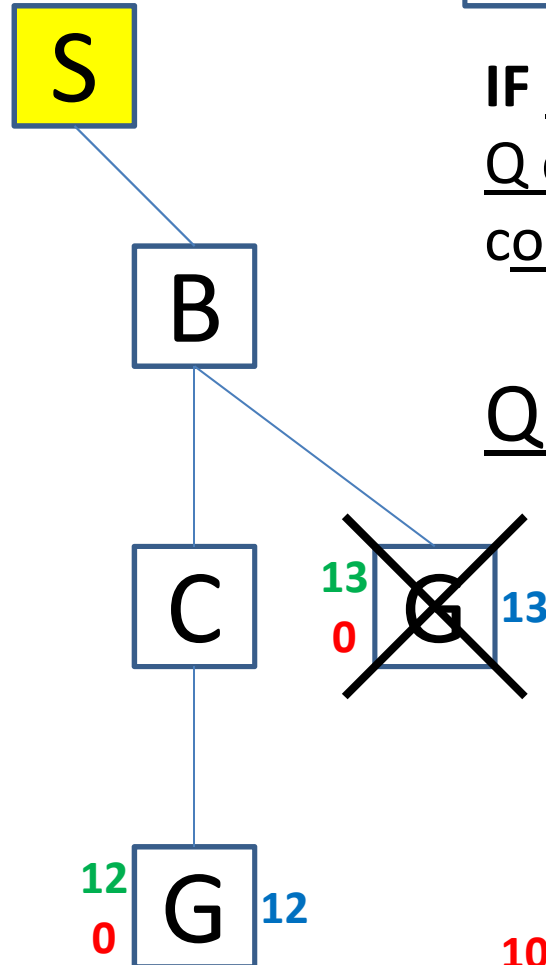


A* algorithm by Example

$f = \text{accumulated path cost} + \text{heuristic}$

IF P terminating in I with cost P &&
Q containing I with cost Q **AND**
cost P \geq cost Q **THEN** remove P

QUEUE: <SBCG,SBG>

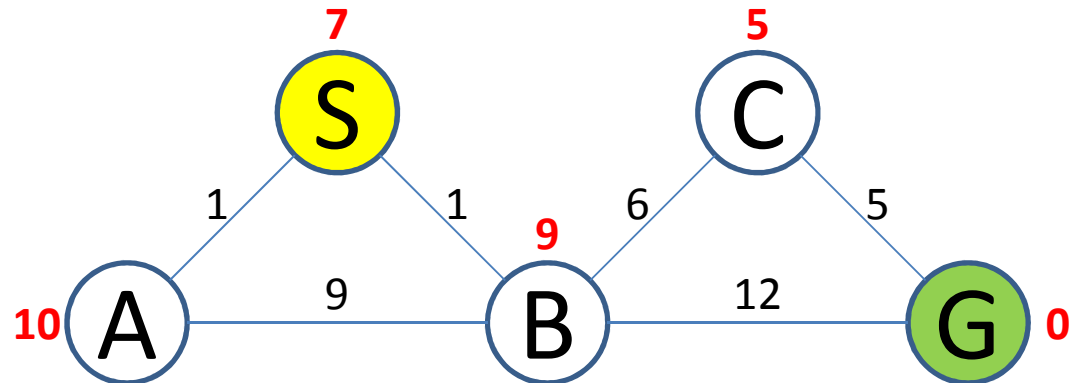
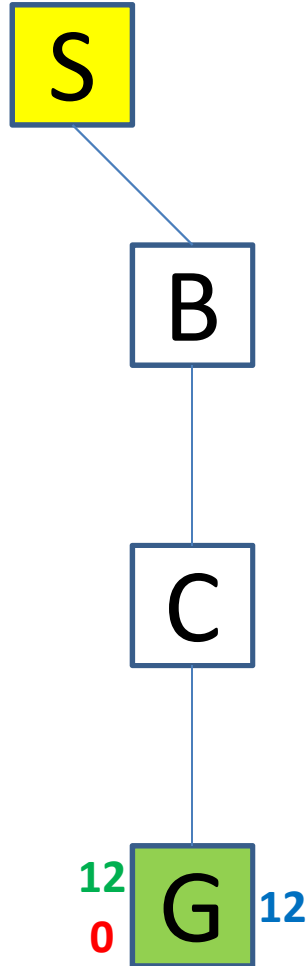


A* algorithm by Example

$f = \text{accumulated path cost} + \text{heuristic}$

SUCCESS

QUEUE: <SBCG>

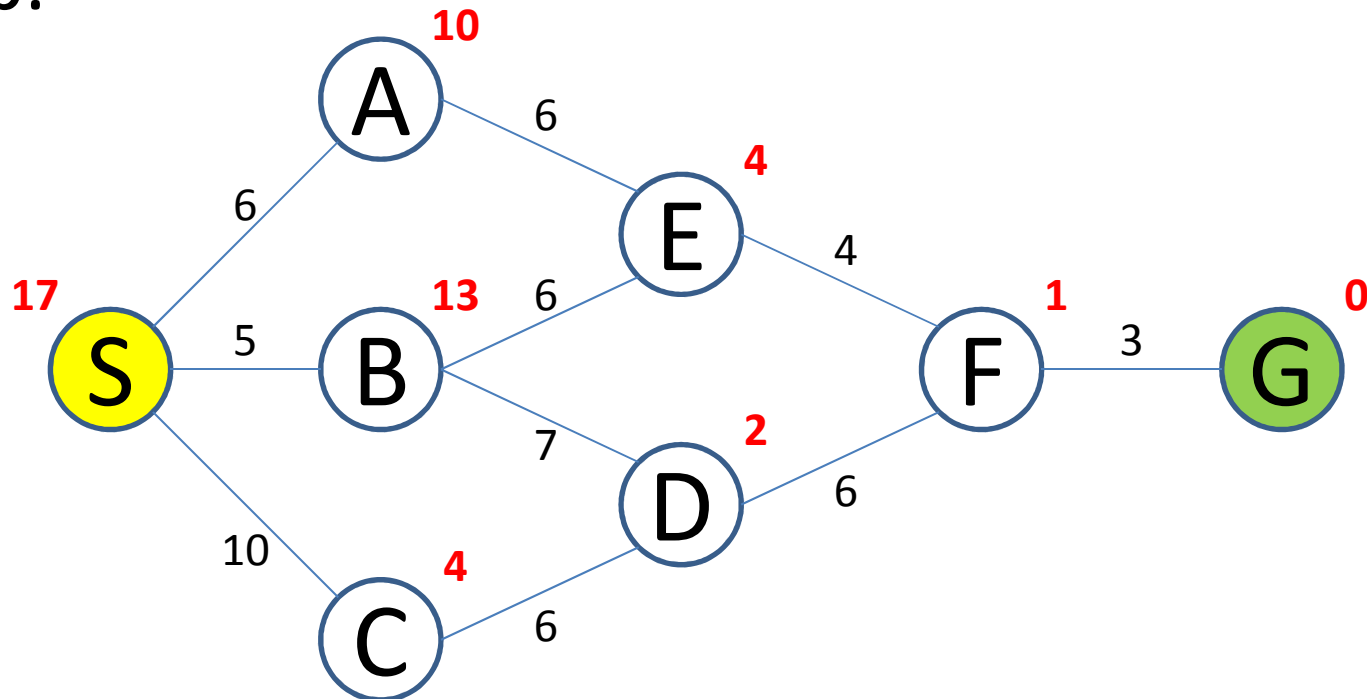


A*

PROBLEM

Problem

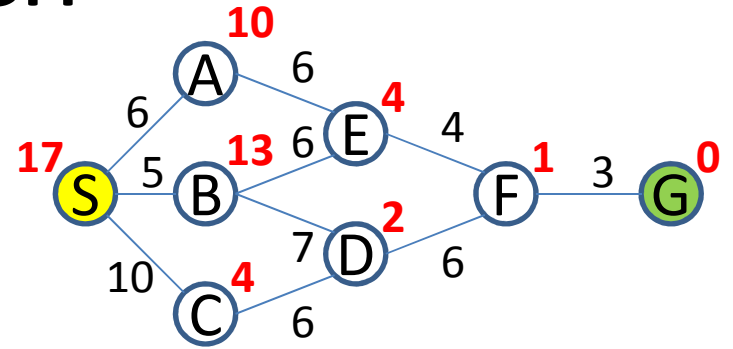
- Perform the A^* Algorithm on the following figure. Explicitly write down the queue at each step.



A*

A* SEARCH

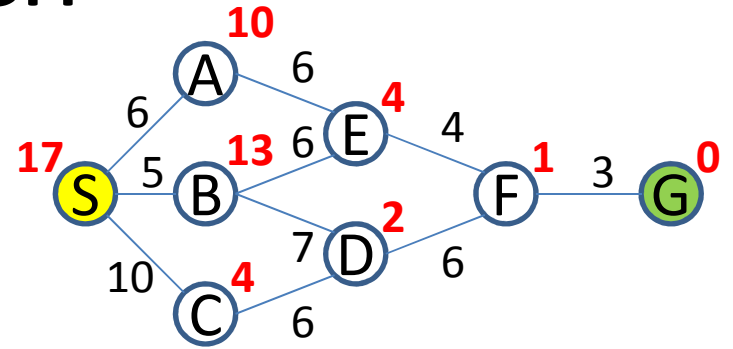
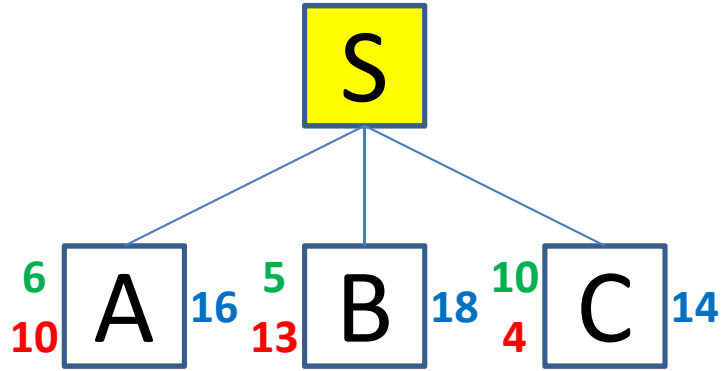
A* Search



QUEUE:

S

A* Search



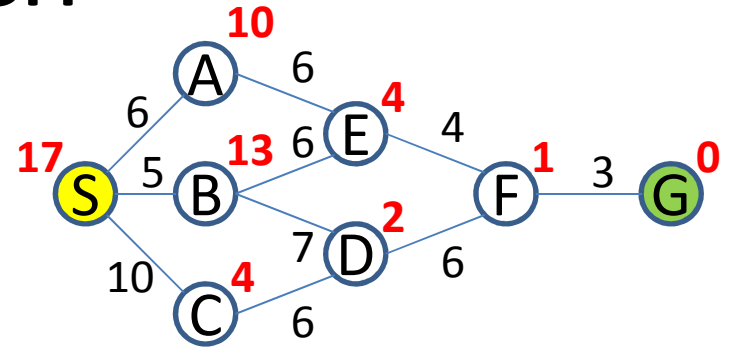
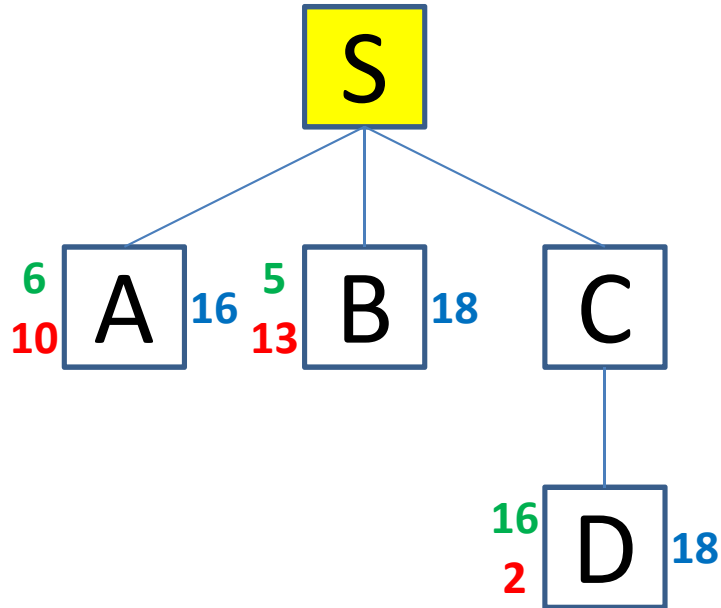
QUEUE:

SC

SA

SB

A* Search



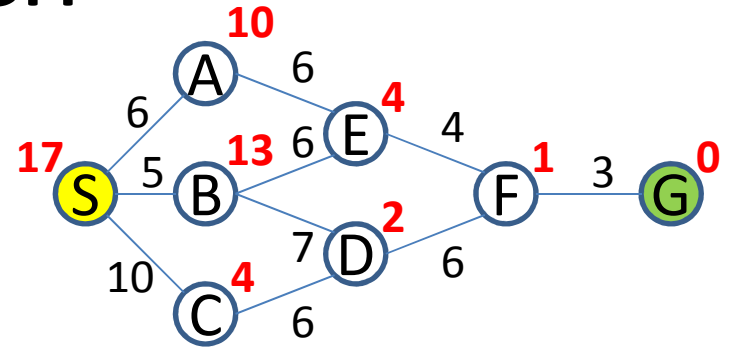
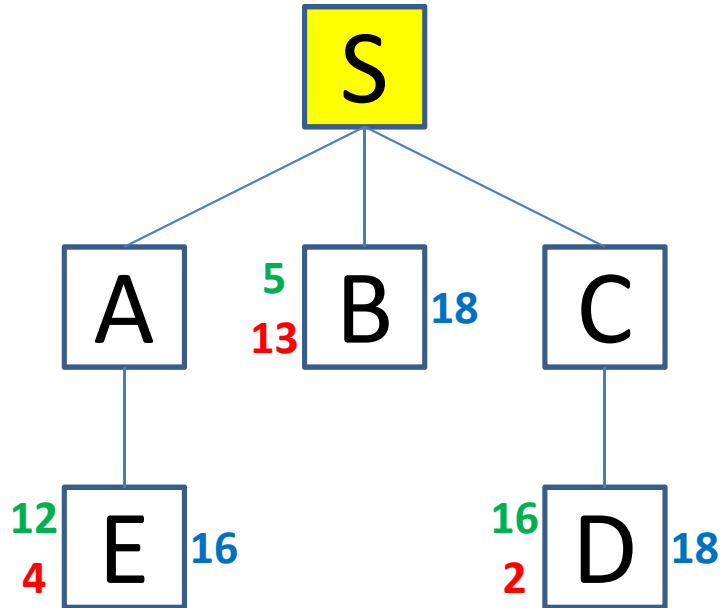
QUEUE:

SA

SCD

SB

A* Search



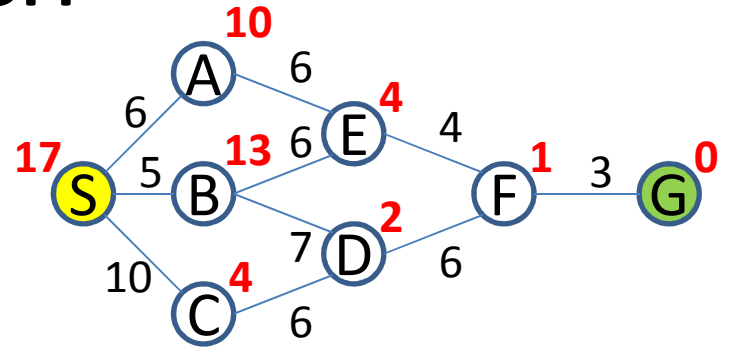
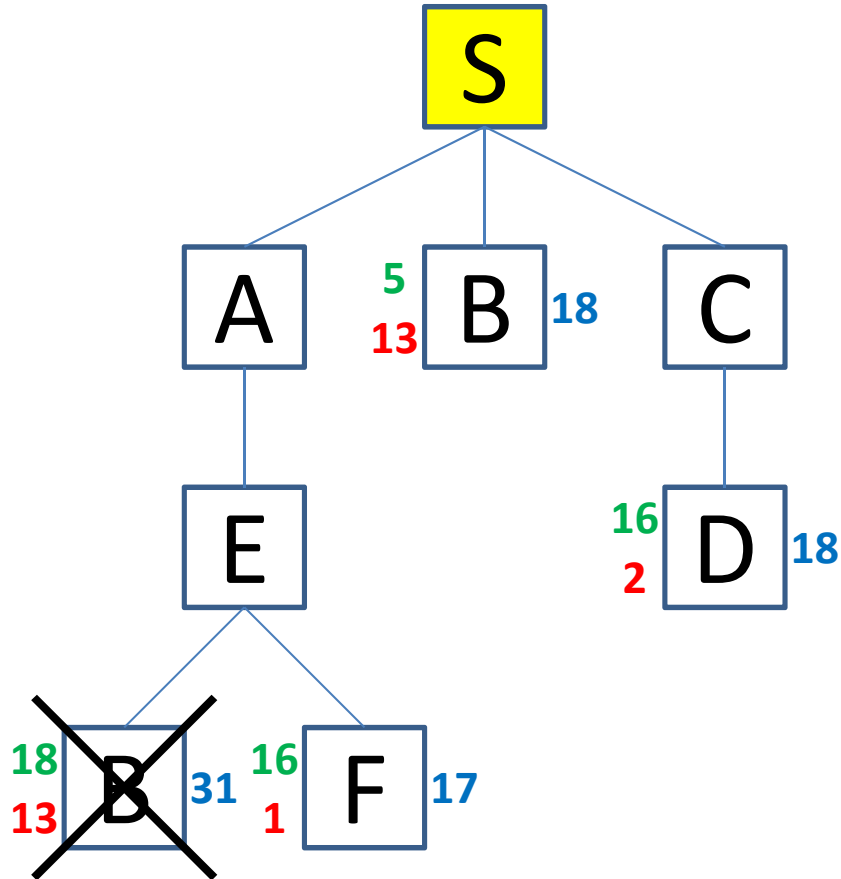
QUEUE:

SAE

SCD

SB

A* Search



QUEUE:

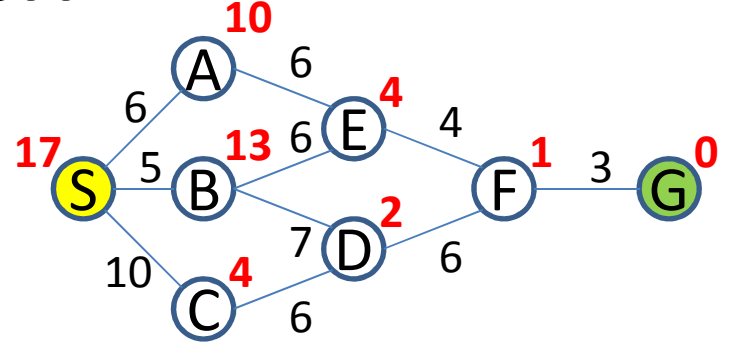
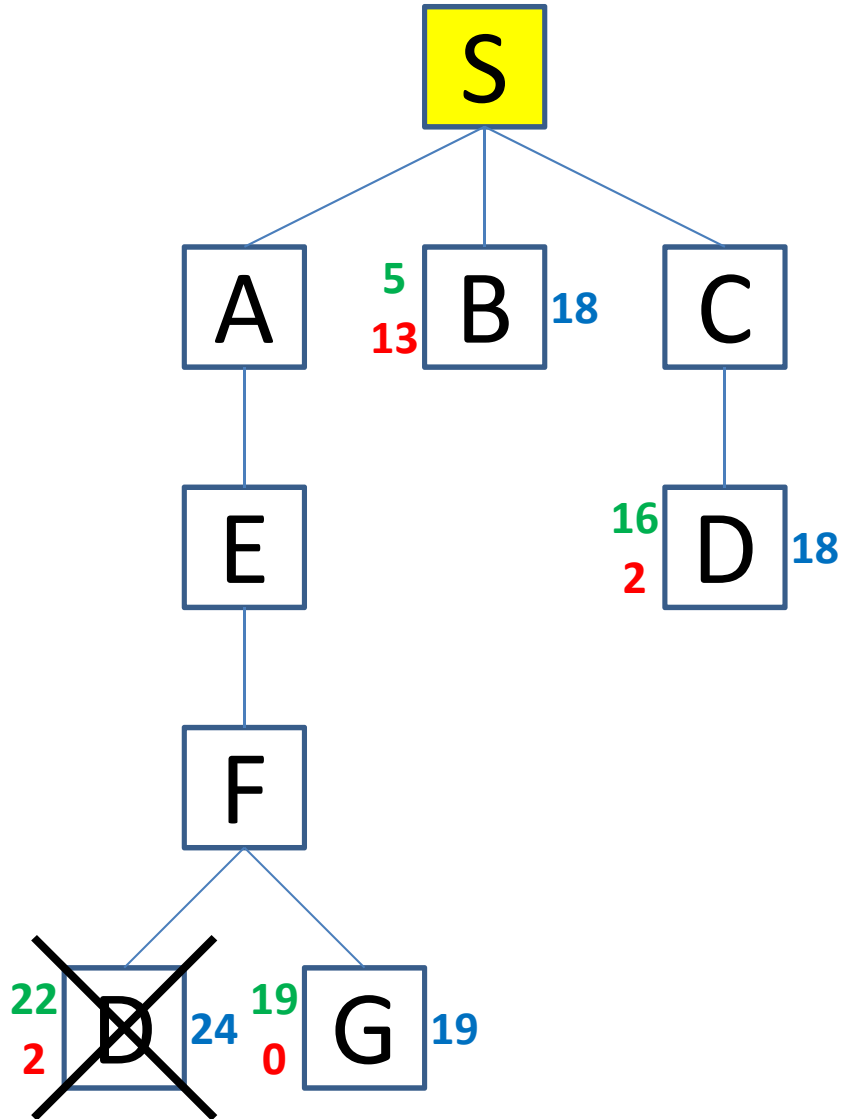
SAEF

SCD

SB

SAEB

A* Search



QUEUE:

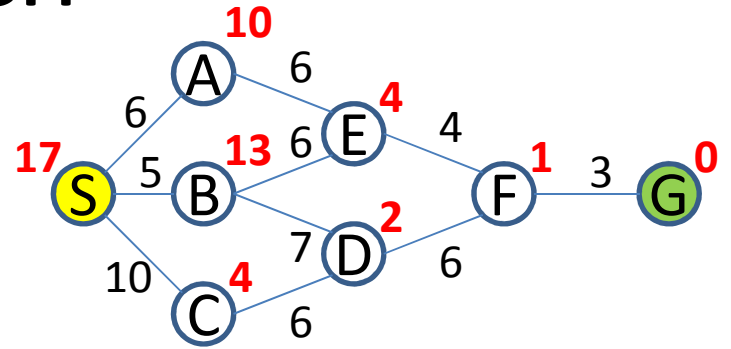
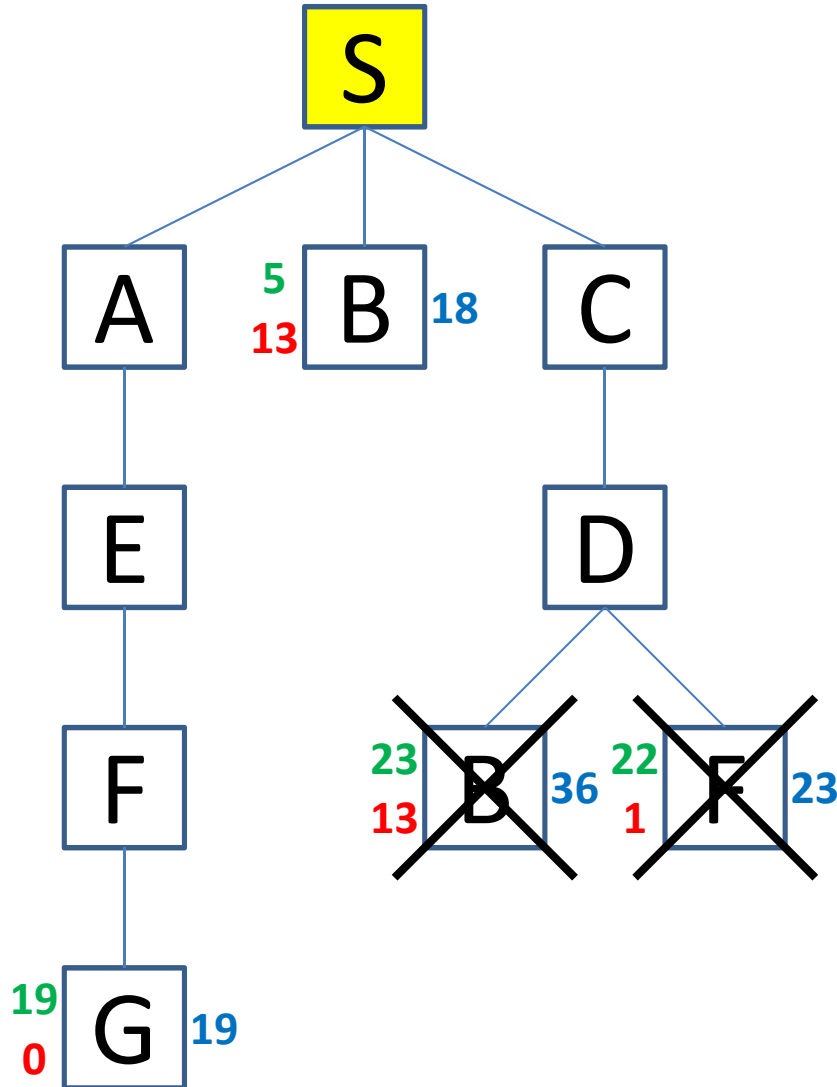
SCD

SB

SAEFG

SAEFD

A* Search



QUEUE:

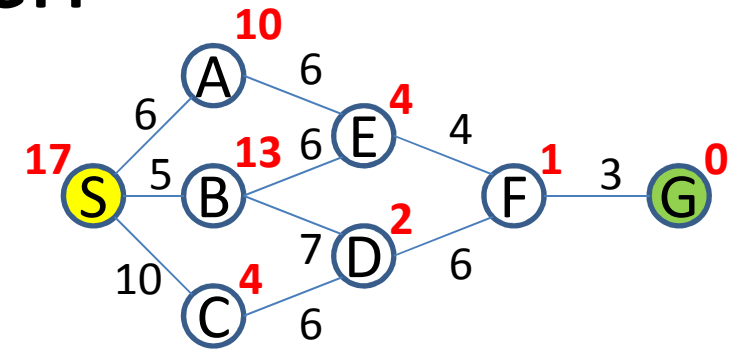
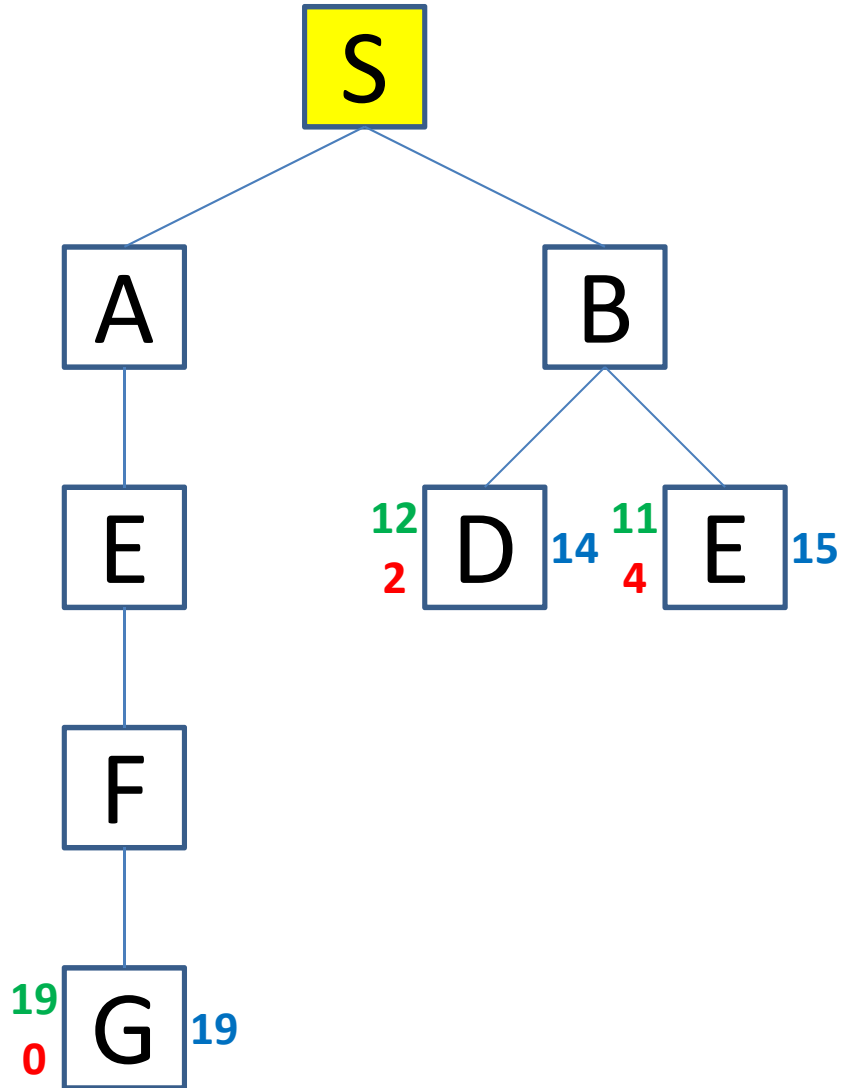
SB

SAEFG

SCDF

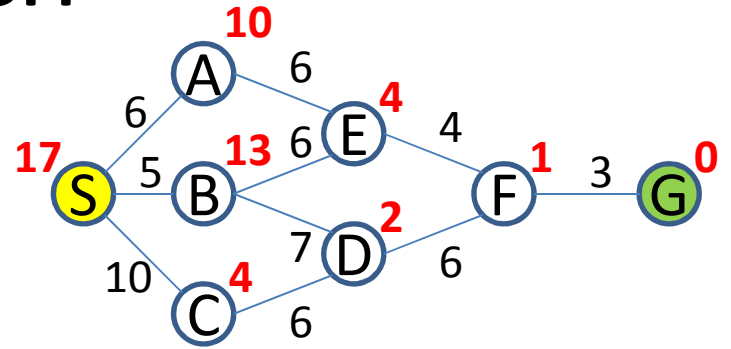
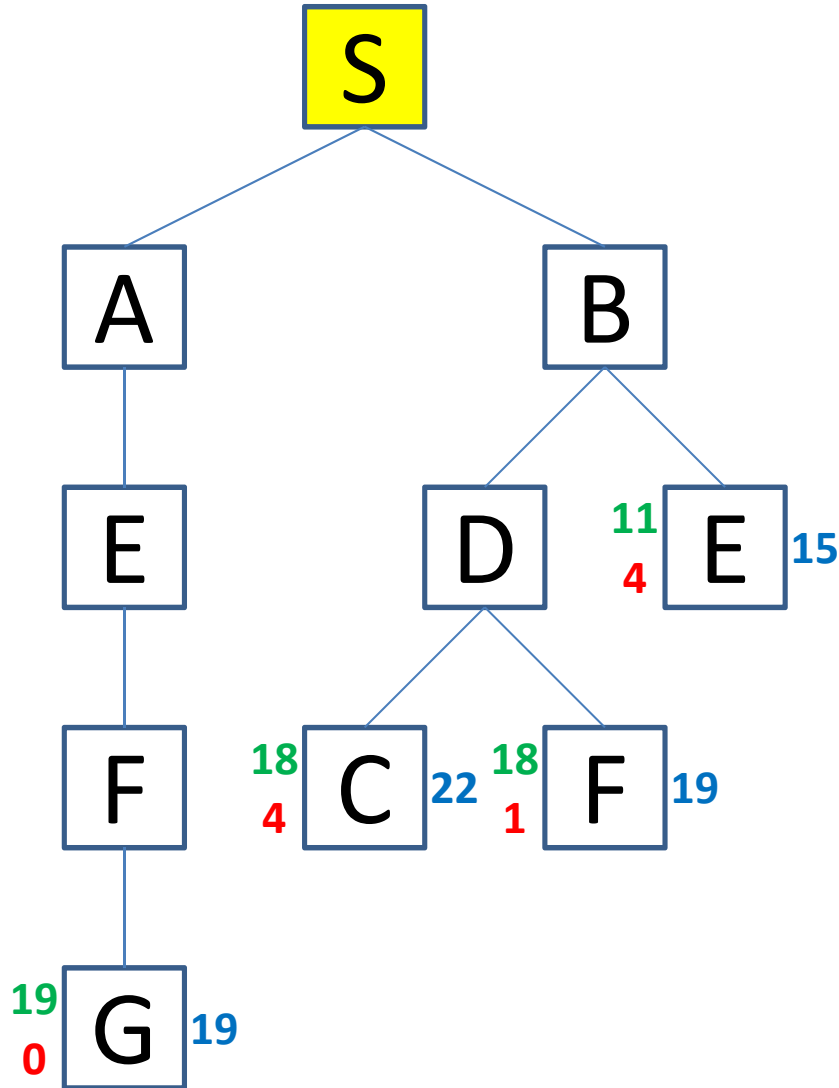
SCDB

A* Search



QUEUE:
SBD
SBE
SAEFG

A* Search



QUEUE:

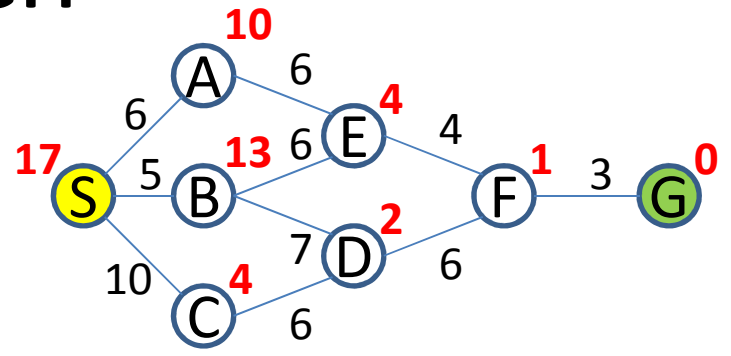
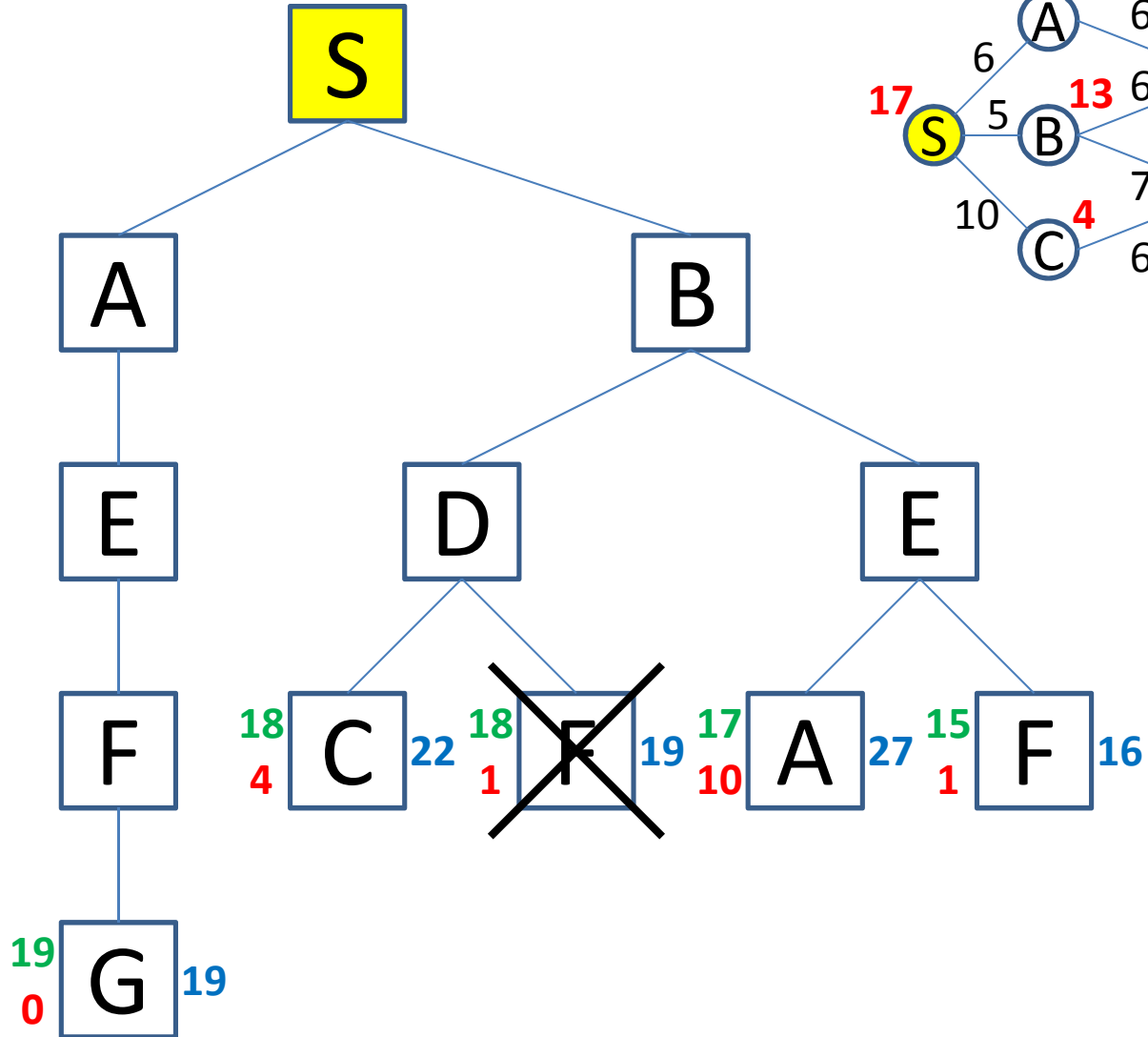
SBE

SBDF

SAEFG

SBDC

A* Search



QUEUE:

SB EF

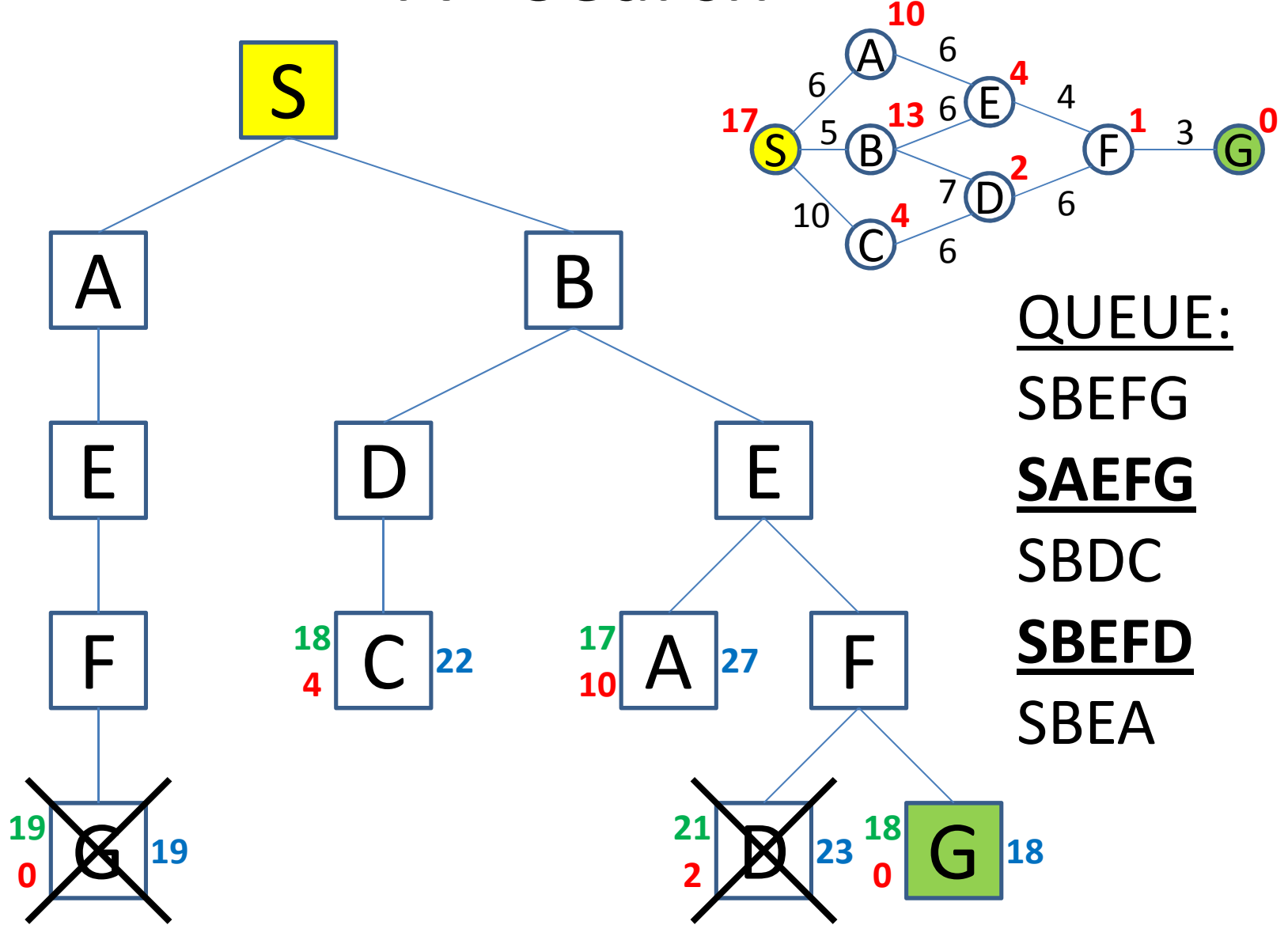
SA EFG

SBDF

SBDC

SBEA

A* Search



QUEUE:

SBEFG

SAEFG

SBDC

SBEFD

SBEA

Exercises: Artificial Intelligence

Iterated Deepening A*

Iterated Deepening A*

IDA* ALGORITHM

IDA* Algorithm

- $f\text{-bound} \leftarrow f(S)$
- **Algorithm:**
 - **WHILE** (goal is not reached) **DO**
 - $f\text{-bound} \leftarrow f\text{-limited_search}(f\text{-bound})$
 - Perform **f-limited search** with $f\text{-bound}$
(See next slide)

f-limited Search Algorithm

- ***Input:***

- QUEUE \leftarrow Path only containing root
- f-bound \leftarrow Natural number
- f-new $\leftarrow \infty$

- ***Algorithm:***

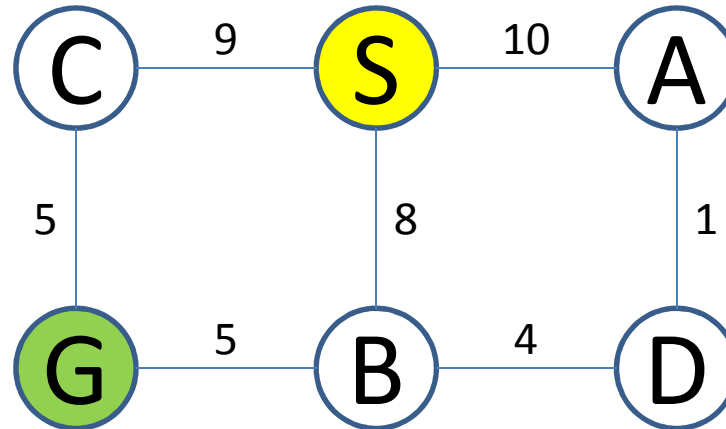
- **WHILE** (QUEUE not empty && goal not reached) **DO**
 - Remove **first path** from QUEUE
 - Create paths to children
 - Reject paths with loops
 - Add paths with **f(path) \leq f-bound** to **front** of QUEUE (*depth-first*)
 - f-new \leftarrow minimum({f-new} \cup {f(P) | P is rejected path})
- **IF** goal reached **THEN** success **ELSE** report f-new

Iterated Deepening A*

PROBLEM

Problem

- Perform the IDA* Algorithm on the following figure.

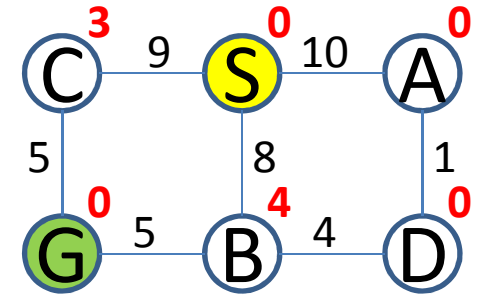


	S	A	B	C	D	G
heuristic	0	0	4	3	0	0

Iterated Deepening A*

IDA* SEARCH

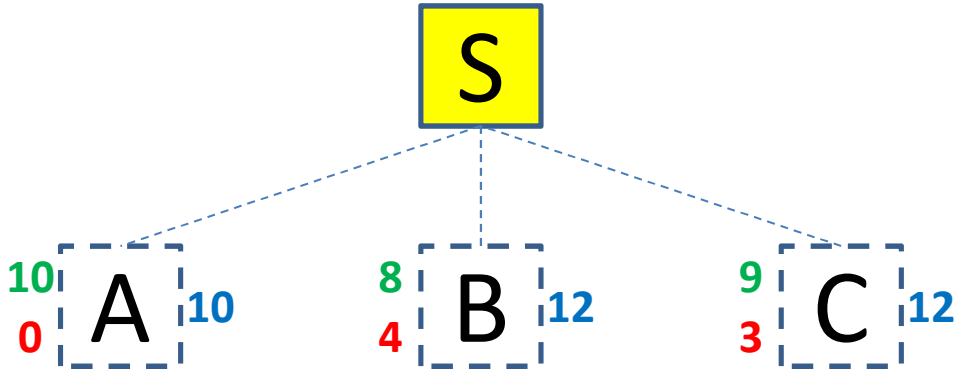
IDA* Search



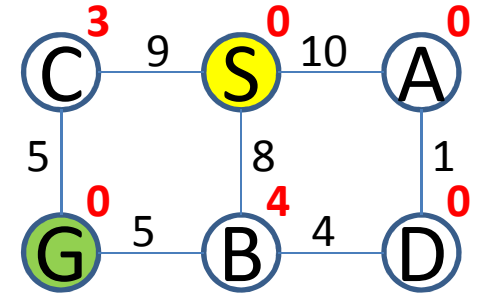
f-bound = 0

f-new = ∞

IDA* Search

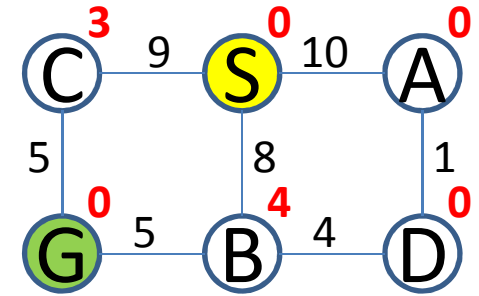


Children are explored depth-first!



f-bound = 0
f-new = 10

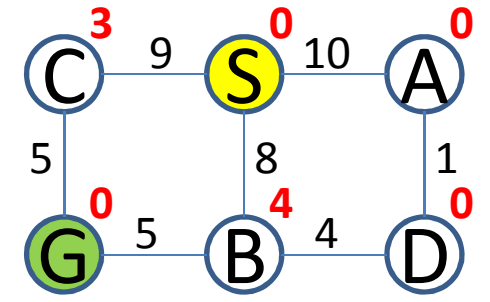
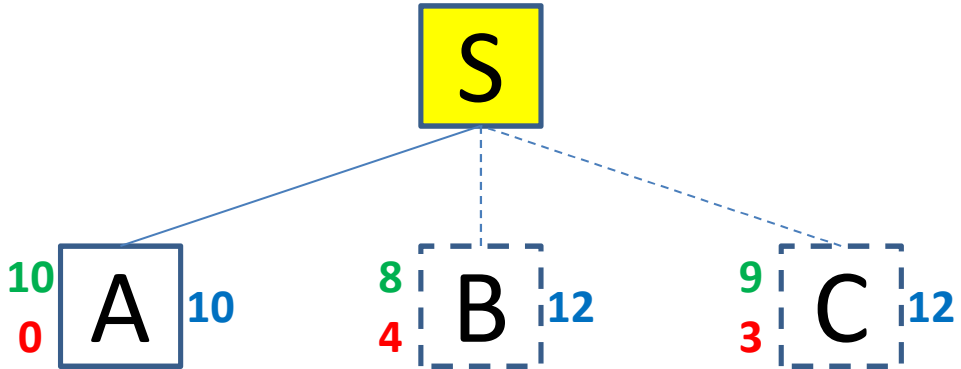
IDA* Search



f-bound = 10

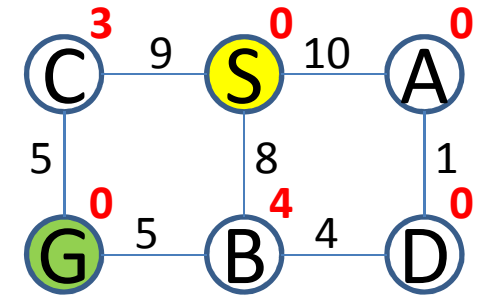
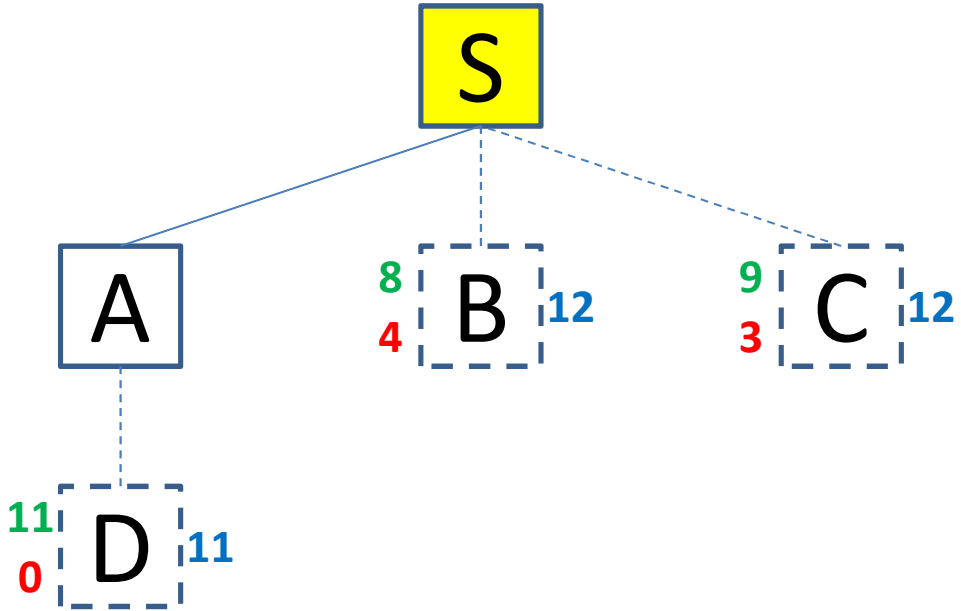
f-new = ∞

IDA* Search



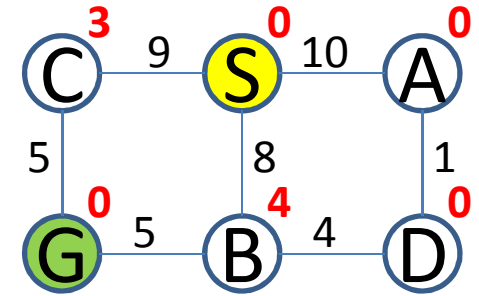
f-bound = 10
f-new = 12

IDA* Search



f-bound = 10
f-new = 11

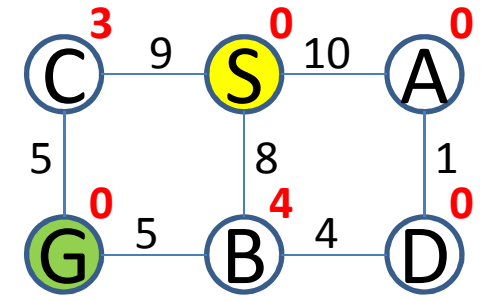
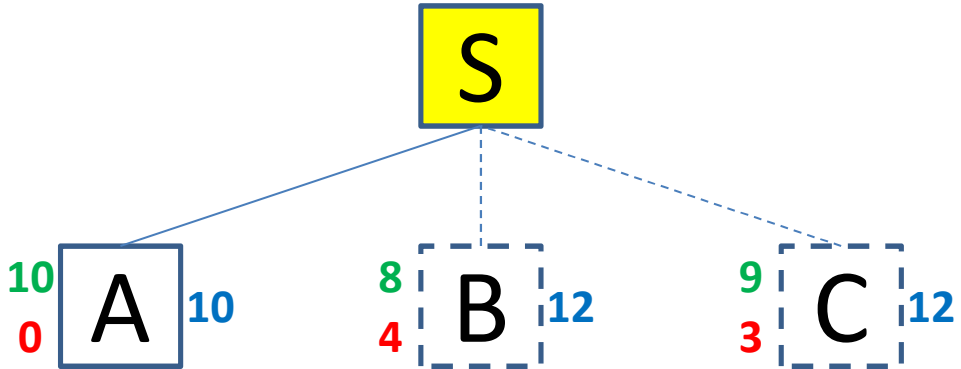
IDA* Search



f-bound = 11

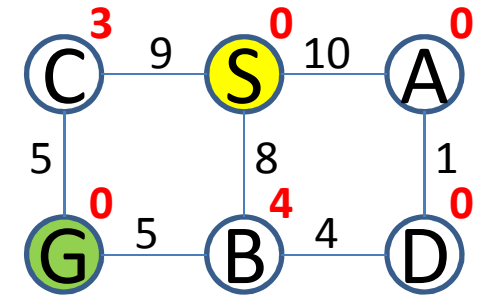
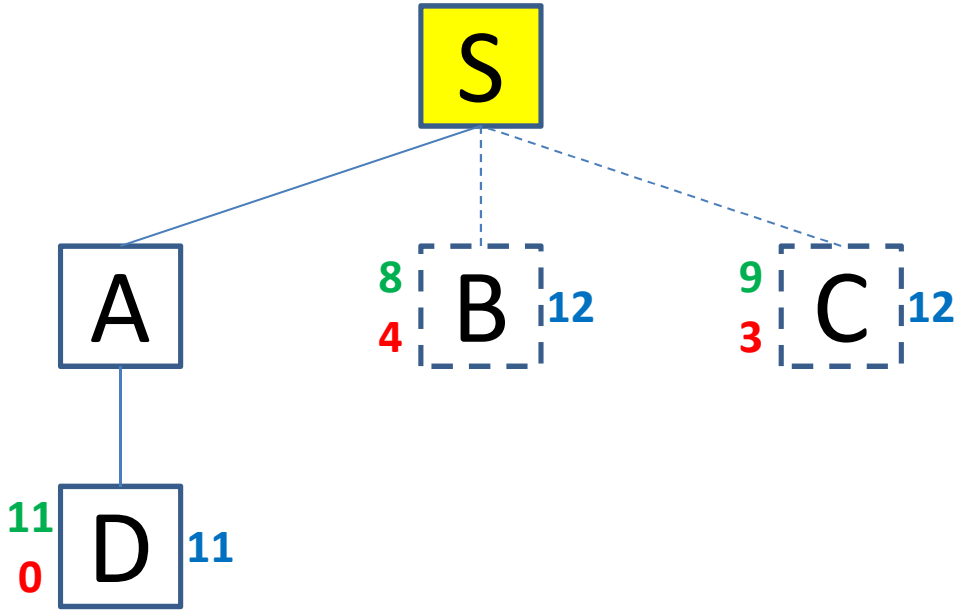
f-new = ∞

IDA* Search



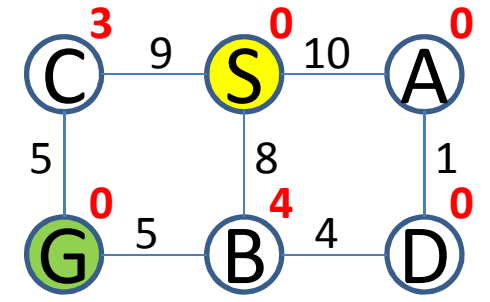
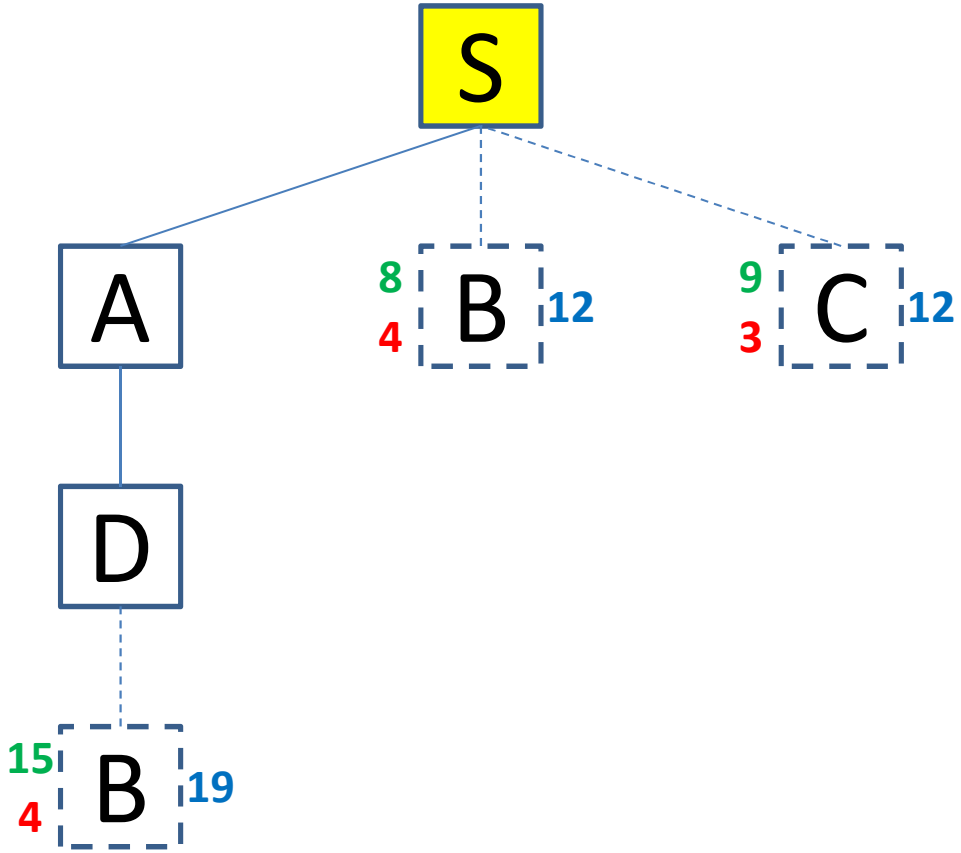
f-bound = 11
f-new = 12

IDA* Search



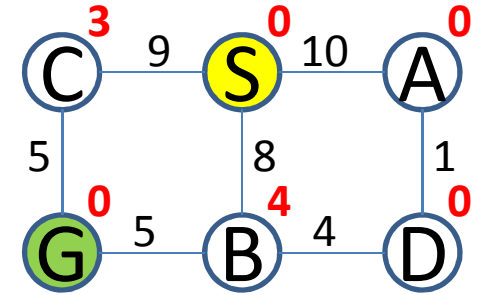
f-bound = 11
f-new = 12

IDA* Search



f-bound = 11
f-new = 12

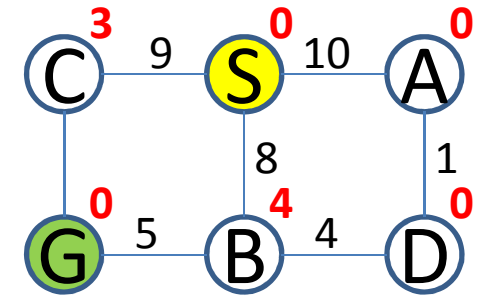
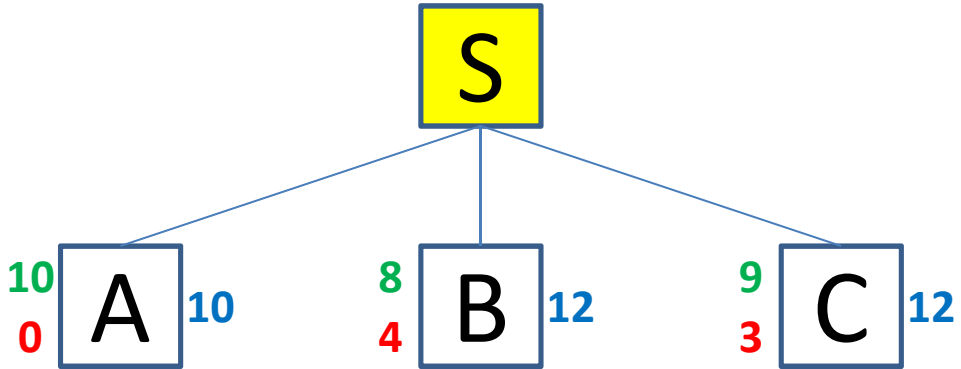
IDA* Search



f-bound = 12

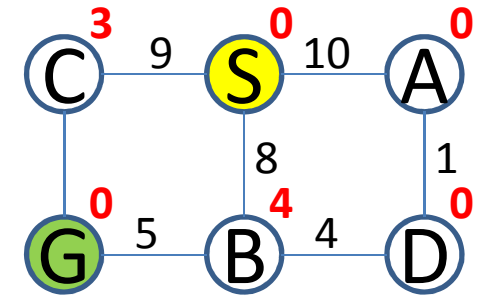
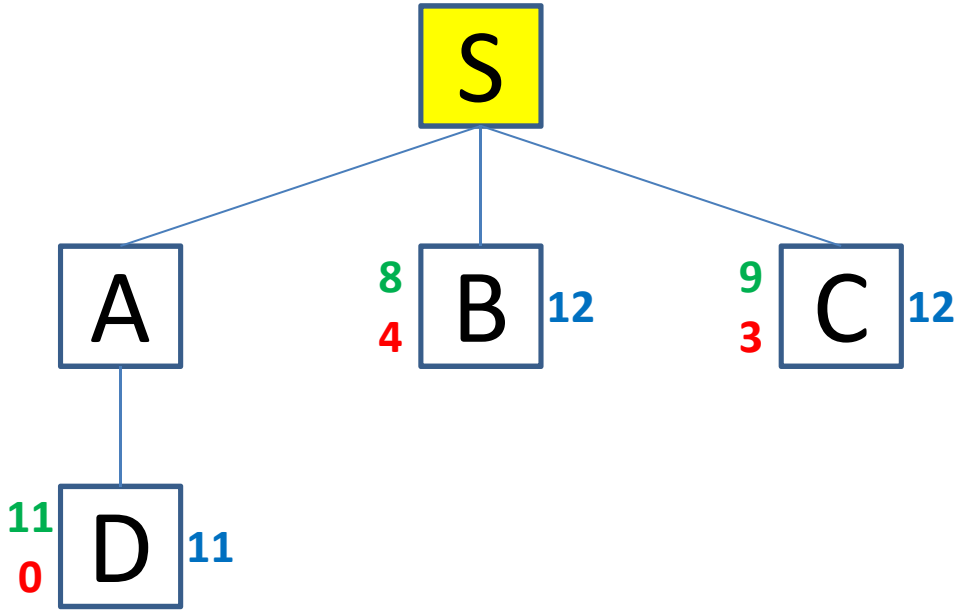
f-new = ∞

IDA* Search



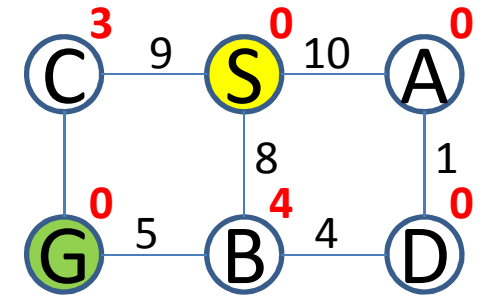
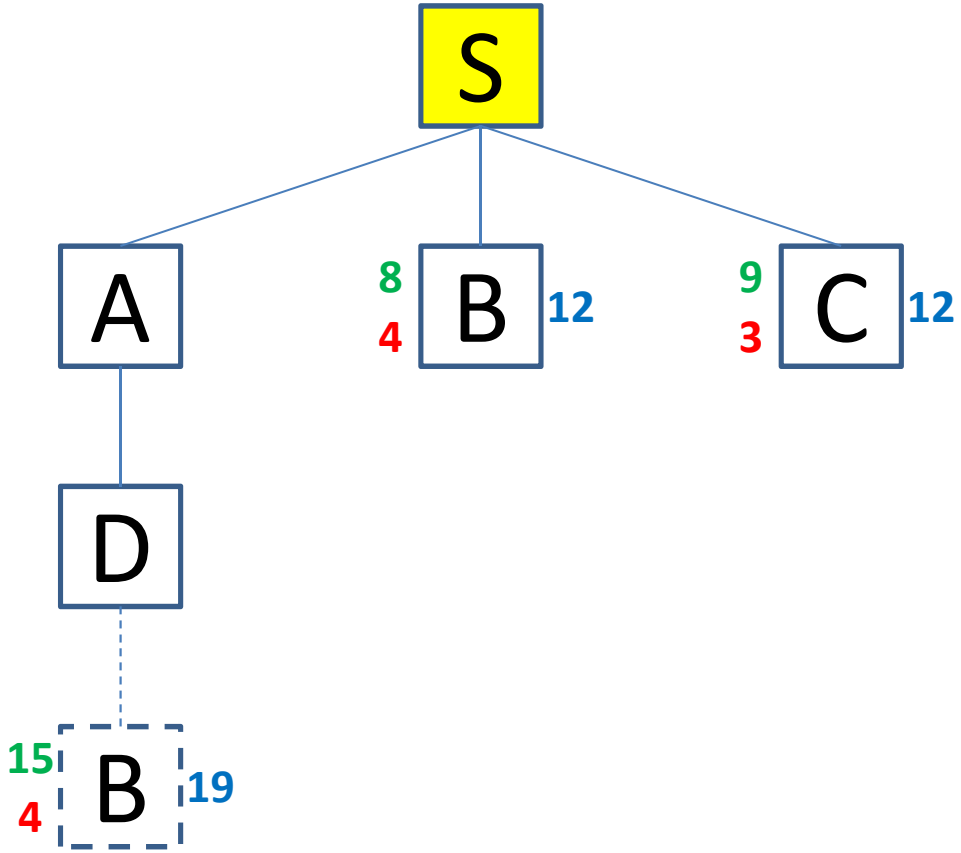
f-bound = 12
f-new = ∞

IDA* Search



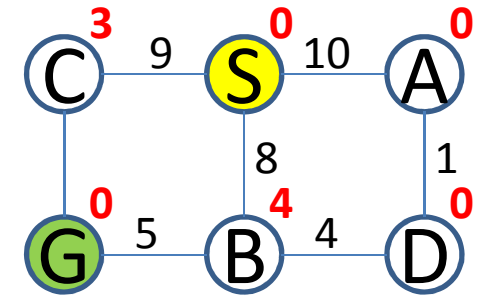
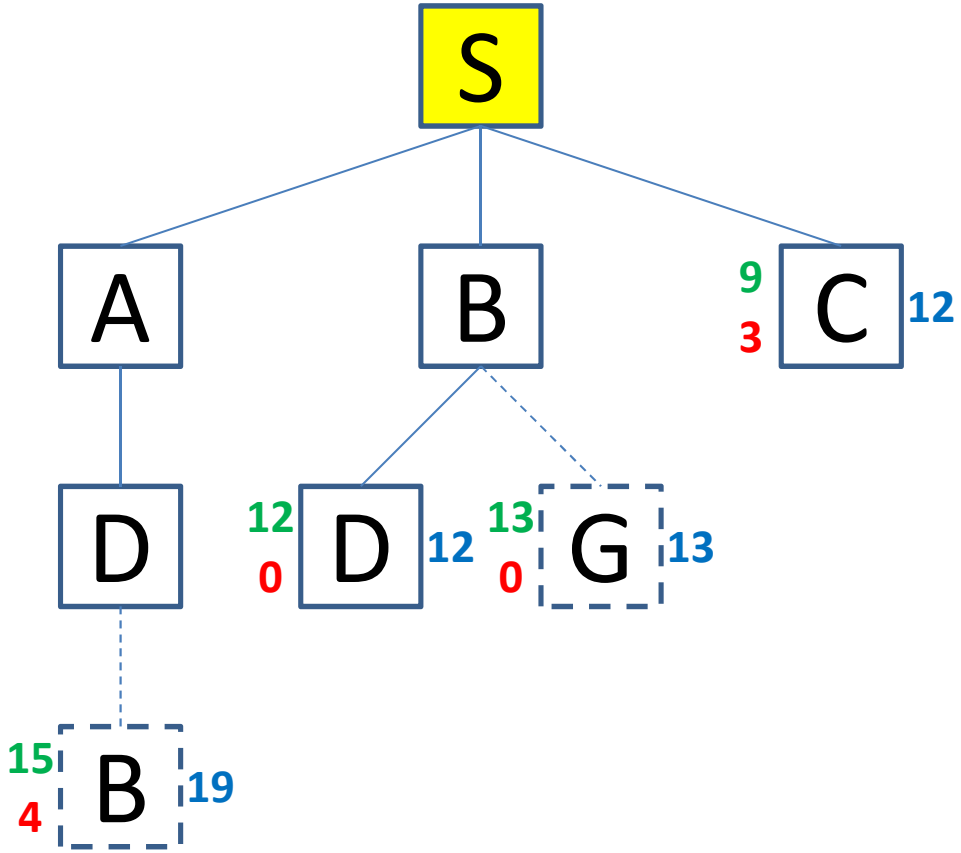
f-bound = 12
f-new = ∞

IDA* Search



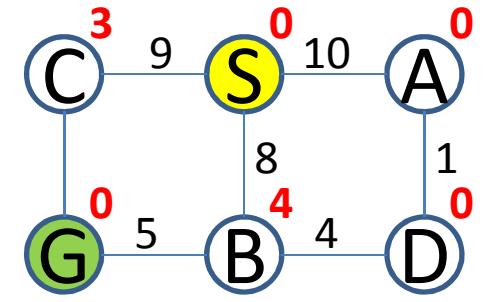
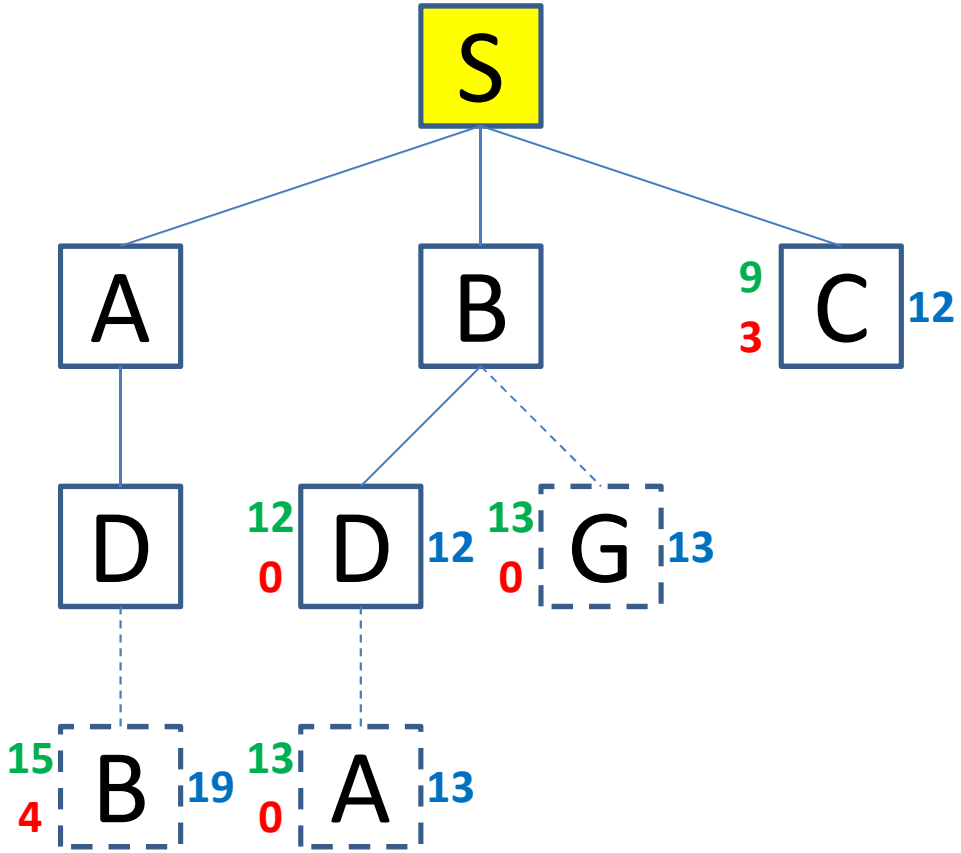
f-bound = 12
f-new = 19

IDA* Search



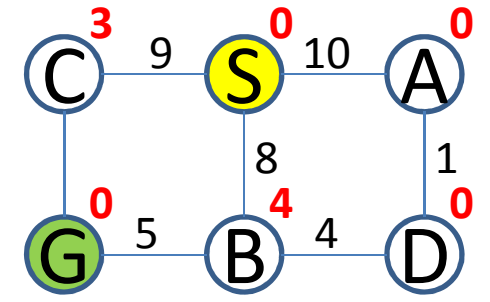
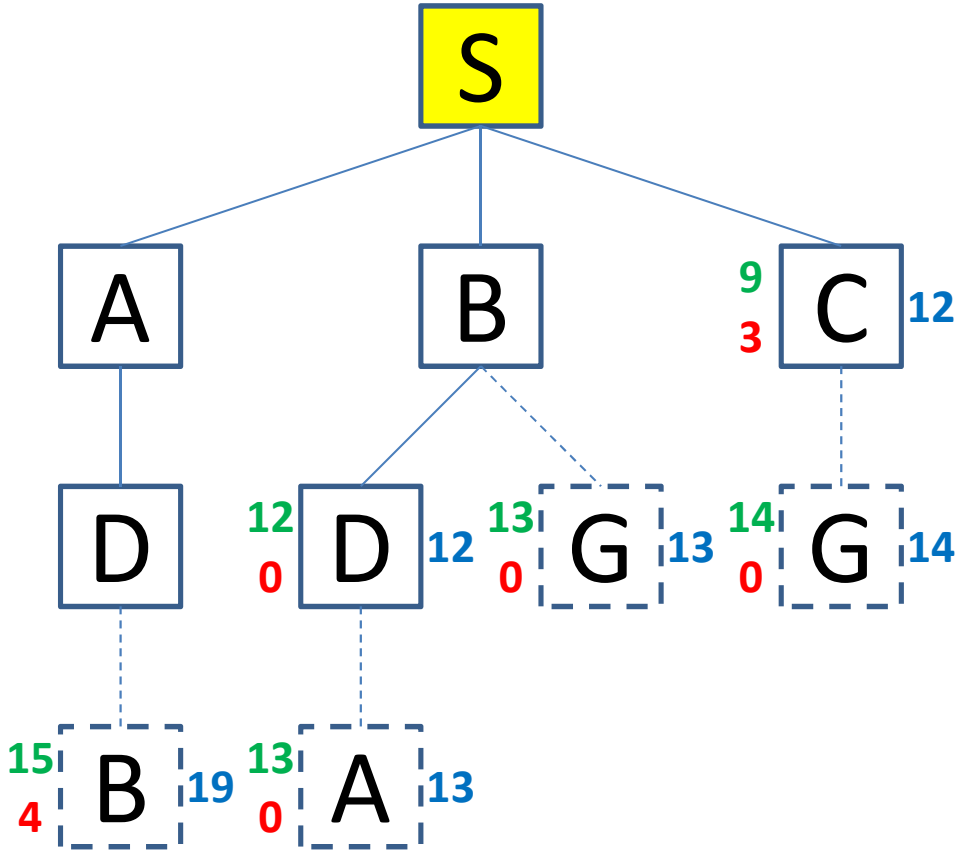
f-bound = 12
f-new = 13

IDA* Search



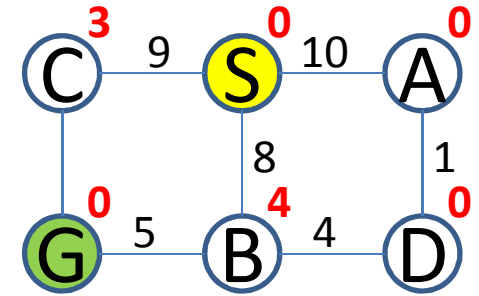
f-bound = 12
f-new = 13

IDA* Search



f-bound = 12
f-new = 13

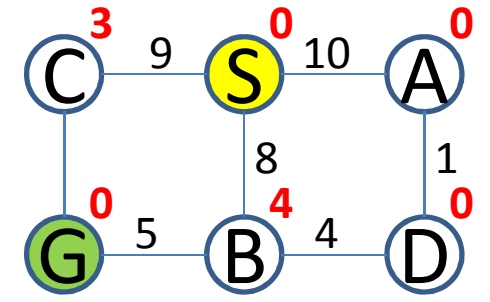
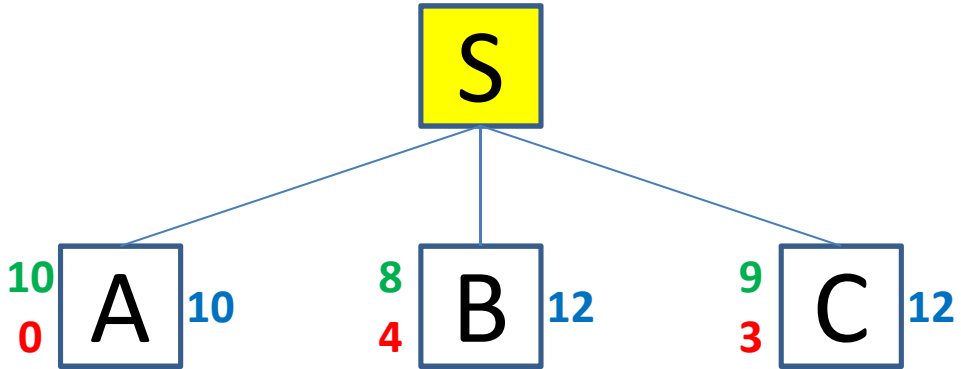
IDA* Search



f-bound = 13

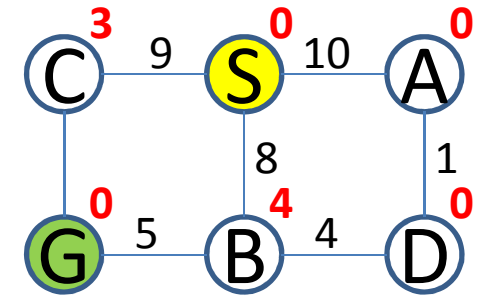
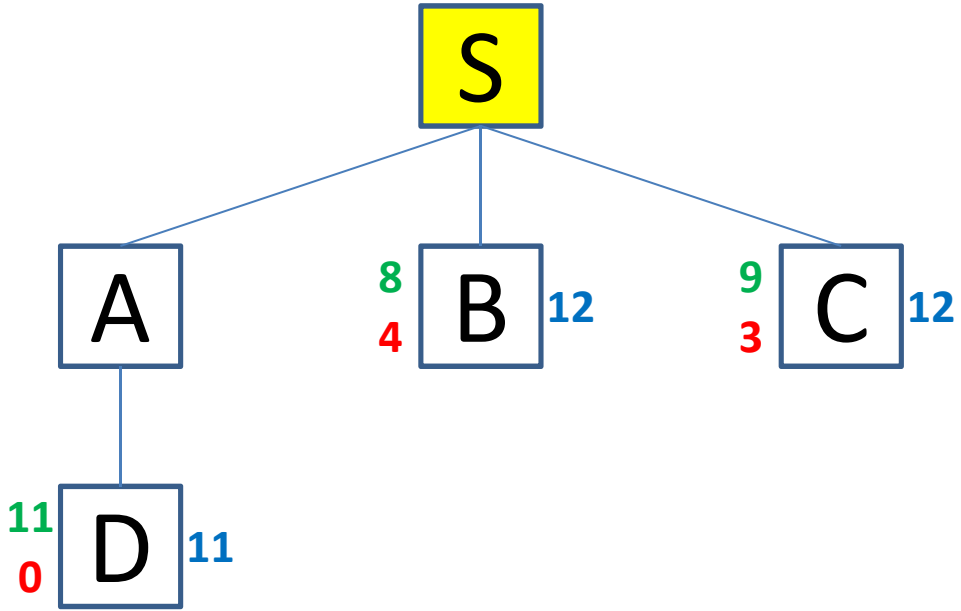
f-new = ∞

IDA* Search



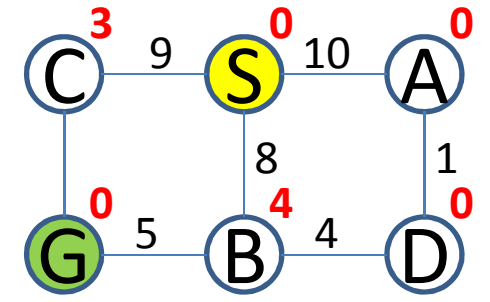
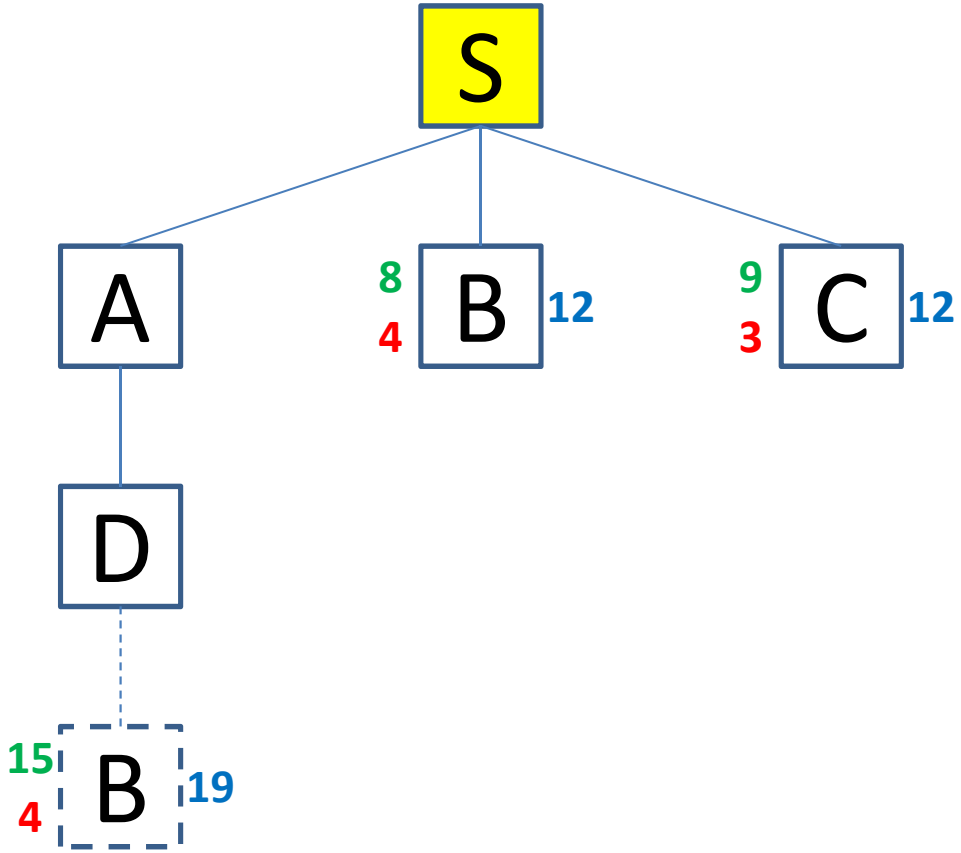
f-bound = 13
f-new = ∞

IDA* Search



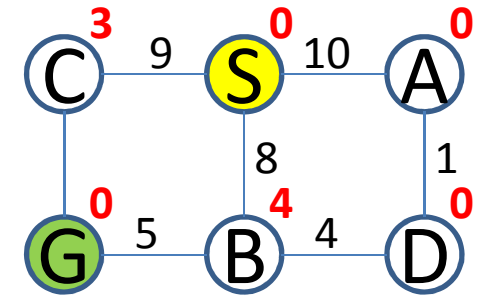
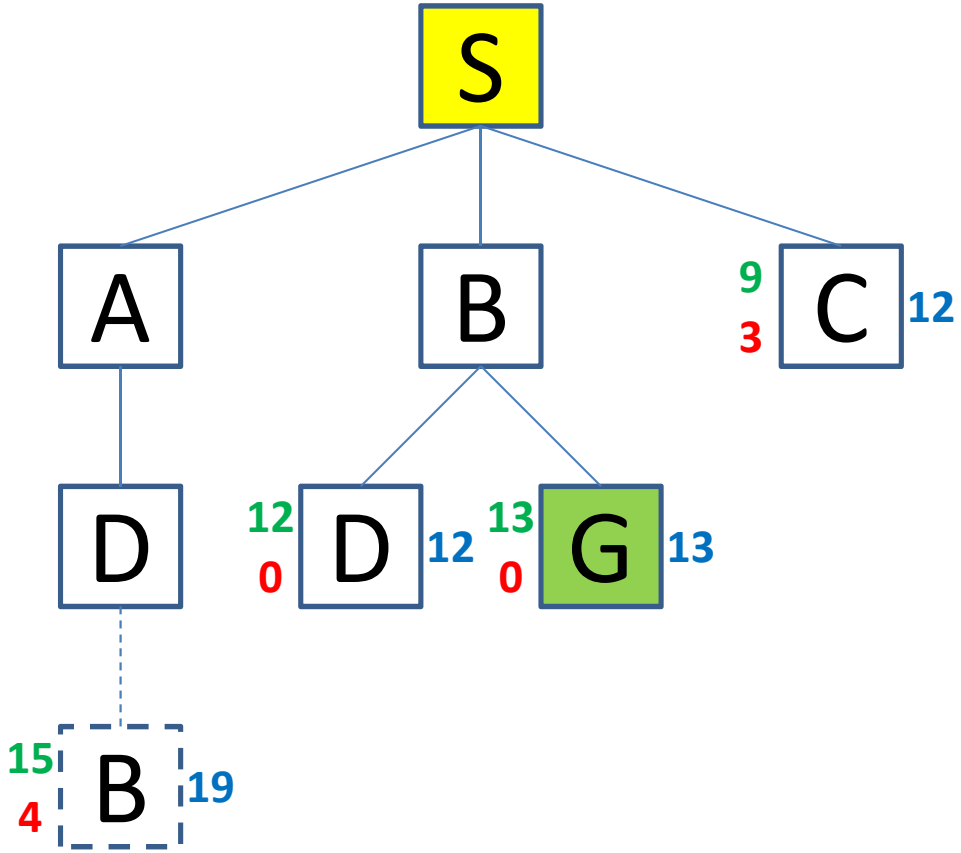
f-bound = 13
f-new = ∞

IDA* Search



f-bound = 13
f-new = 19

IDA* Search



f-bound = 13
f-new = 19

Exercises: Artificial Intelligence

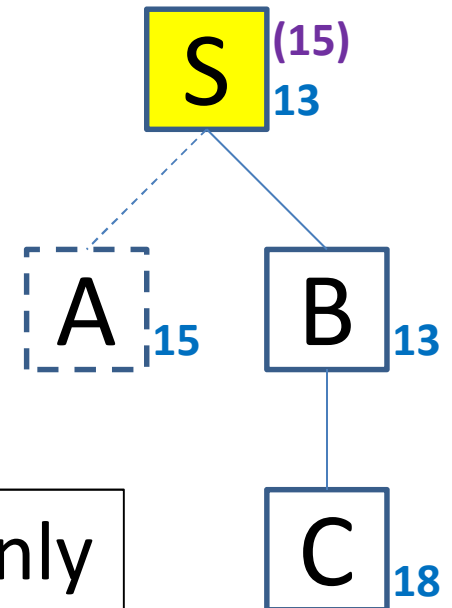
Simplified Memory-bounded A*

Simplified Memory-bounded A*

SMA* ALGORITHM

SMA* Algorithm

- Optimizes A* to work within reduced memory
- **Key Idea:**
 - IF memory full for extra node (C)
 - Remove highest f-value leaf (A)
 - Remember best-forgotten child in each parent node (15 in S)



E.g. Memory of 3 nodes only

SMA* Algorithm

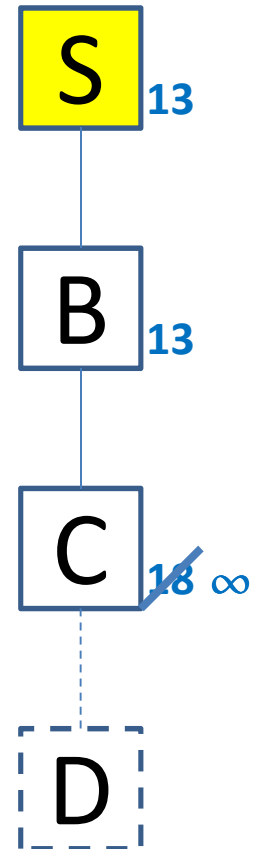
- **Generate Children 1 by 1**
 - **Expanding:** add 1 child at the time to QUEUE
 - Avoids **memory overflow**
 - **Allows monitoring** if nodes need deletion



SMA* Algorithm

- **Too long paths: Give up**
 - **Extending path cannot fit in memory**
 - give up (C)
 - **Set f-value node (C) to ∞**
 - **Remembers:**
path cannot be found here

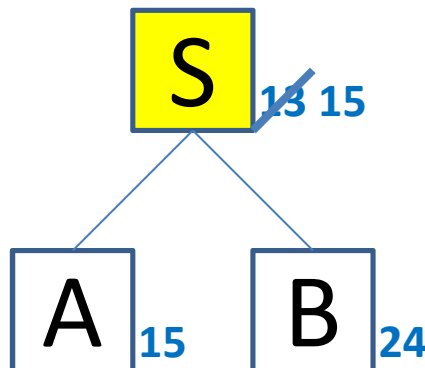
E.g. Memory of 3 nodes only



SMA* Algorithm

- **Adjust f-values**

- **IF** all children M_i of node N have been explored
- **AND** $\forall i: f(S...M_i) > f(S...N)$
- **THEN reset** (through $N \implies$ through children)
 - $f(S...N) = \min\{f(S...M_i) \mid M_i \text{ child of } N\}$



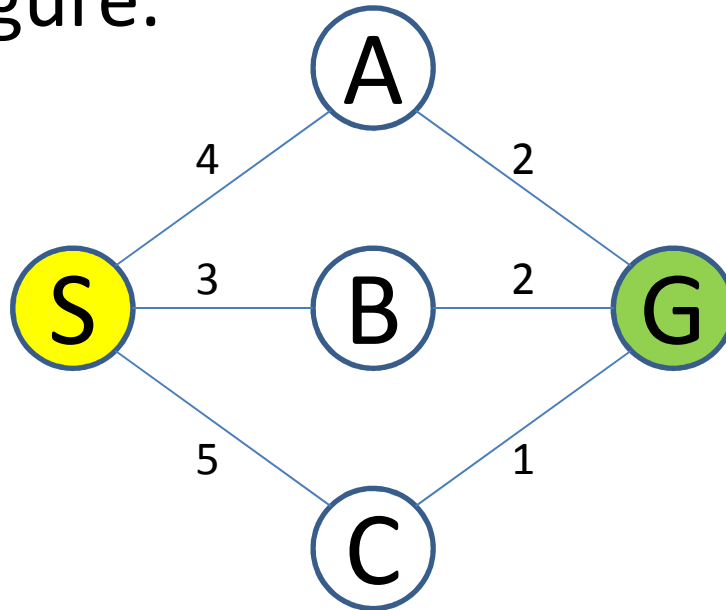
Better estimate for $f(S)$

Simplified Memory-bounded A*

SMA* BY EXAMPLE

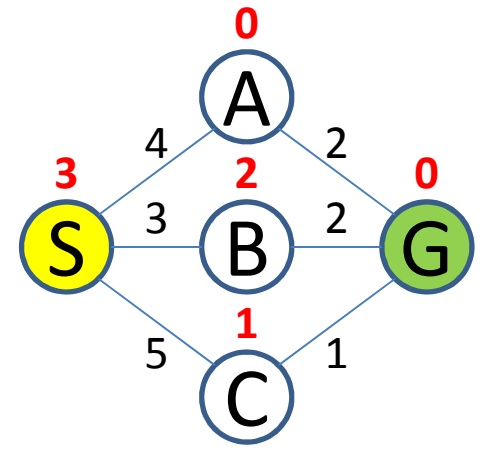
SMA* by Example

- Perform SMA* (memory: 3 nodes) on the following figure.

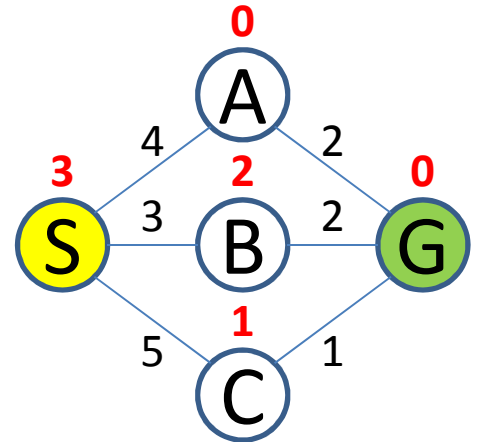
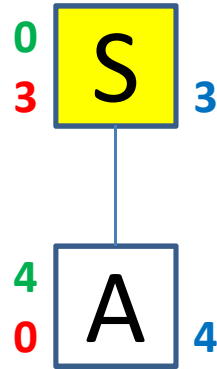


	S	A	B	C	G
heuristic	3	0	2	1	0

SMA* by Example

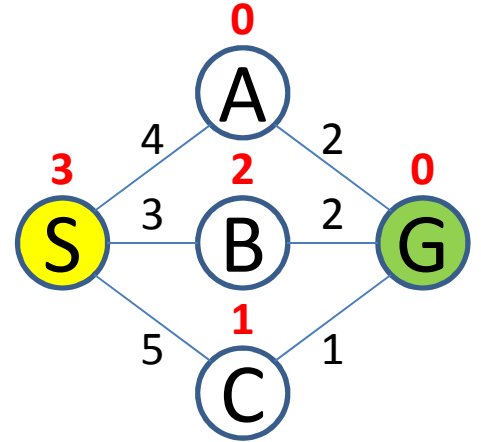
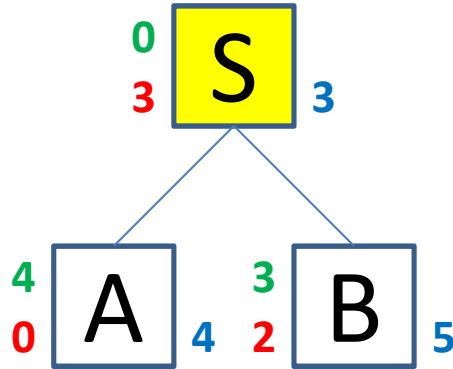


SMA* by Example



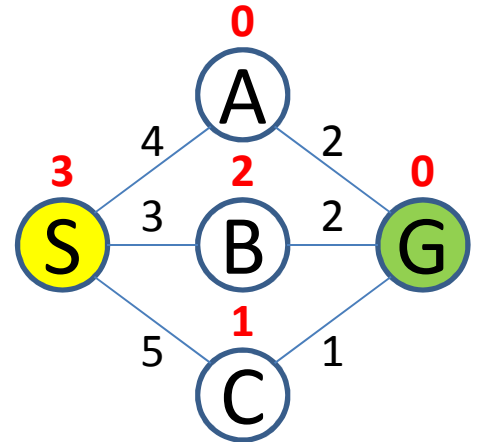
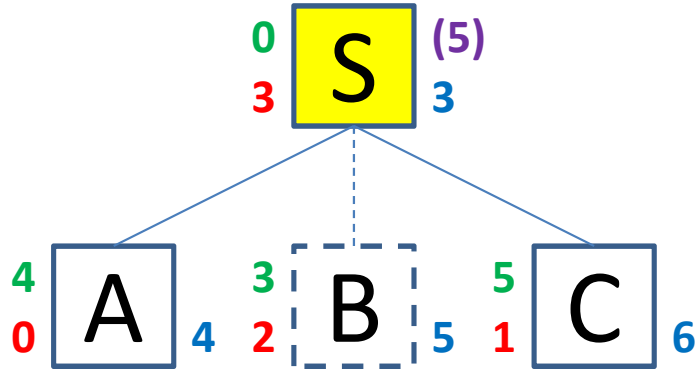
**Generate children
(One by one)**

SMA* by Example



**Generate children
(One by one)**

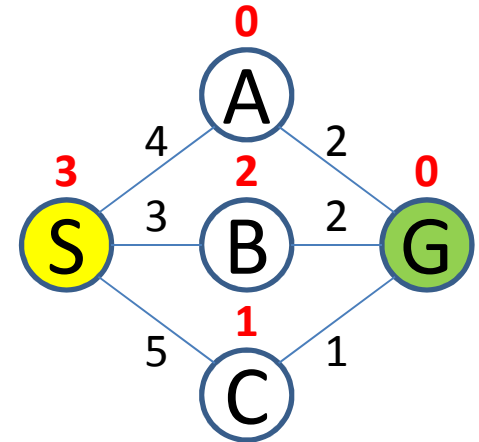
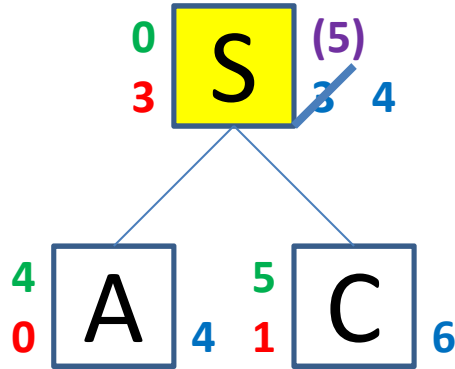
SMA* by Example



**Generate children
(One by one)**

Memory full

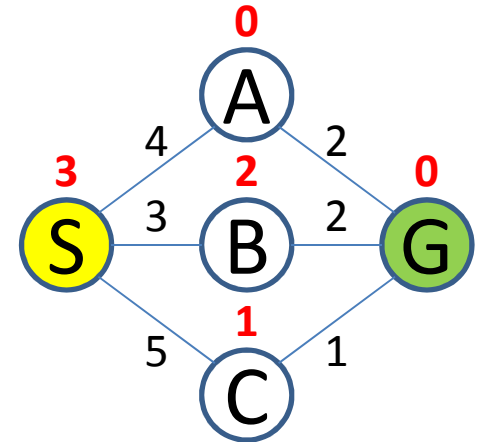
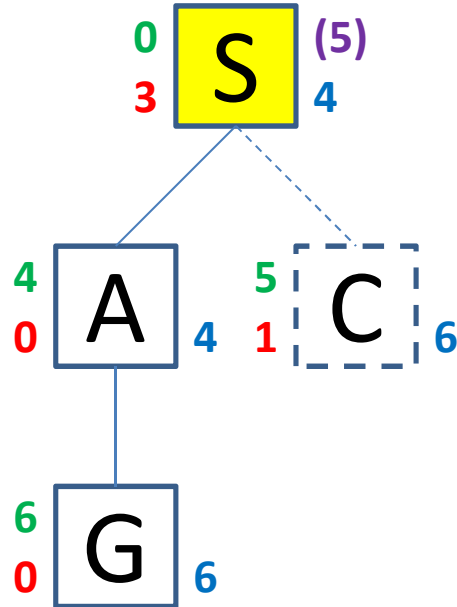
SMA* by Example



All children are explored

Adjust f-values

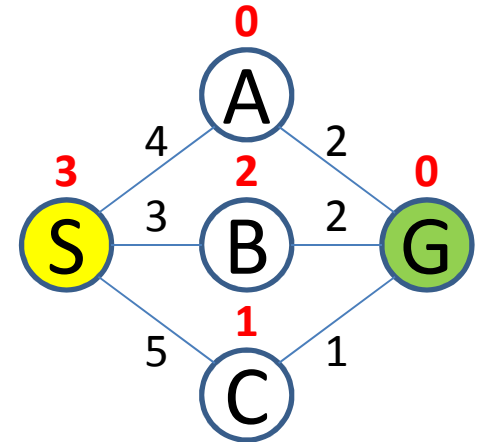
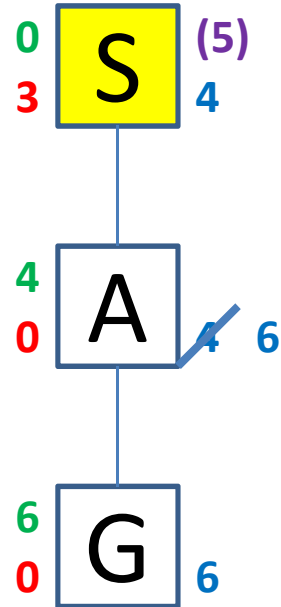
SMA* by Example



**Generate children
(One by one)**

Memory full

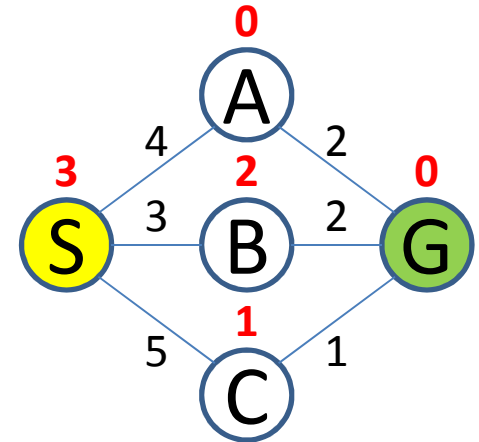
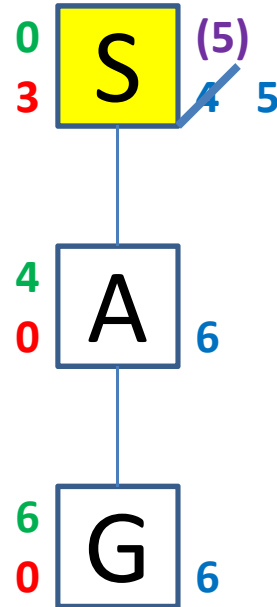
SMA* by Example



All children are explored

Adjust f-values

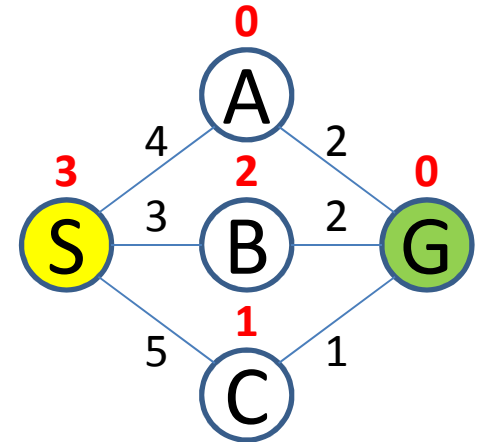
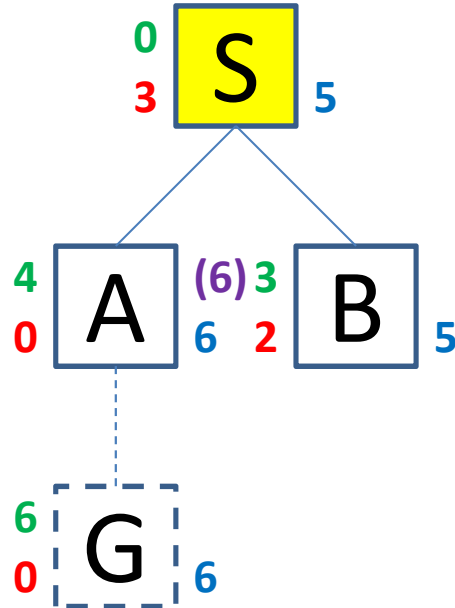
SMA* by Example



All children are explored (update)

Adjust f-values

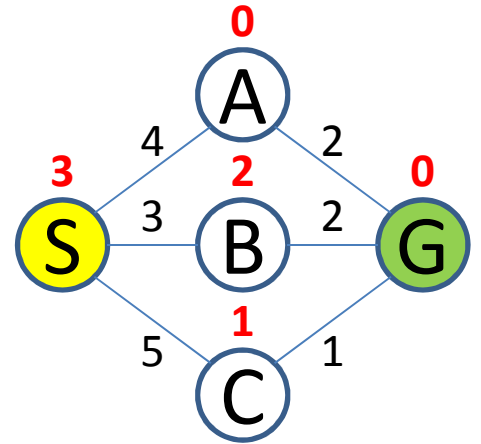
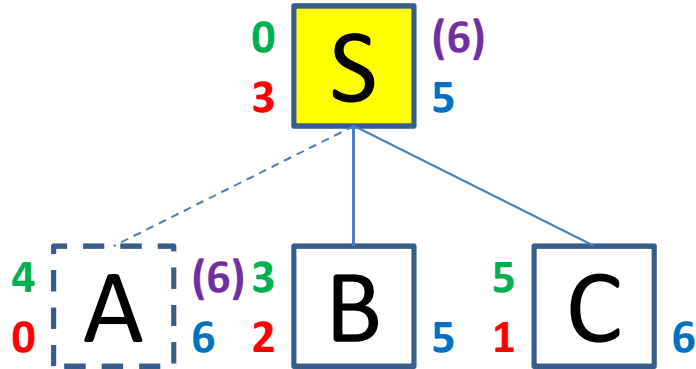
SMA* by Example



**Generate children
(One by one)**

Memory full

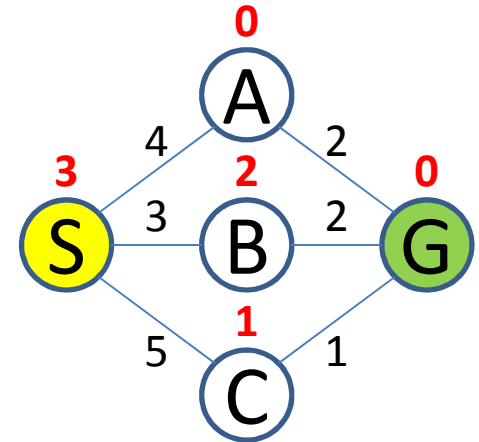
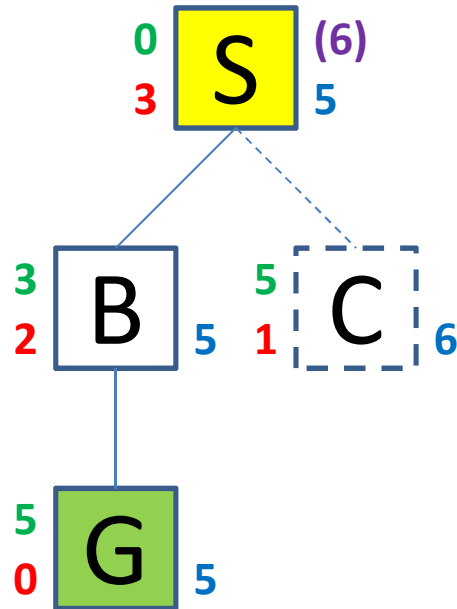
SMA* by Example



**Generate children
(One by one)**

Memory full

SMA* by Example



**Generate children
(One by one)**

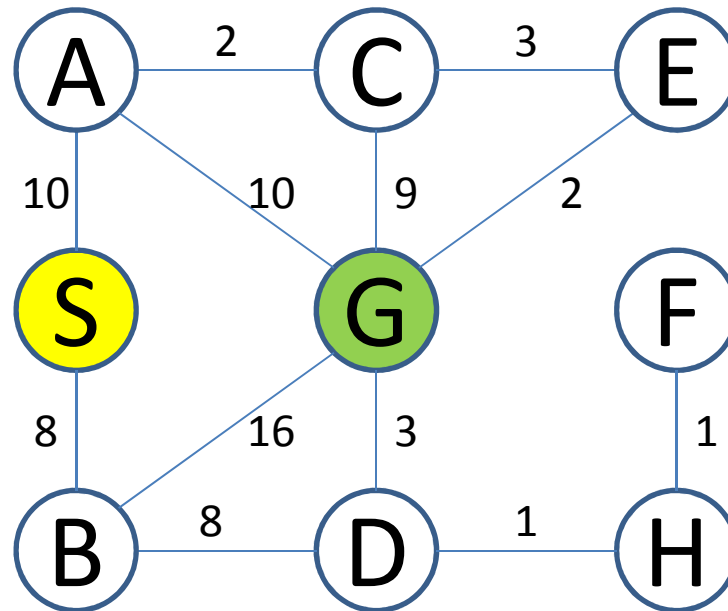
Memory full

Simplified Memory-bounded A*

PROBLEM

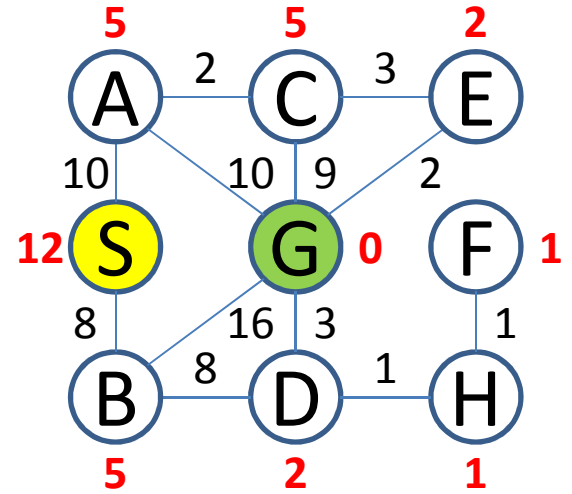
Problem

- Perform SMA* (memory: 4 nodes) on the following figure.

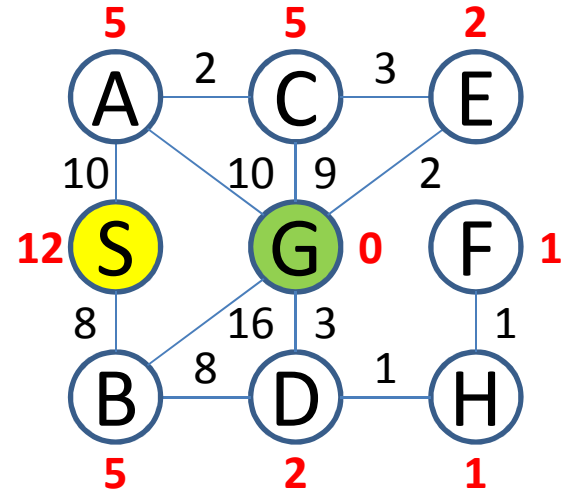
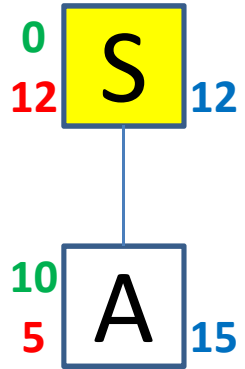


	S	A	B	C	D	E	F	H	G
heuristic	12	5	5	5	2	2	1	1	0

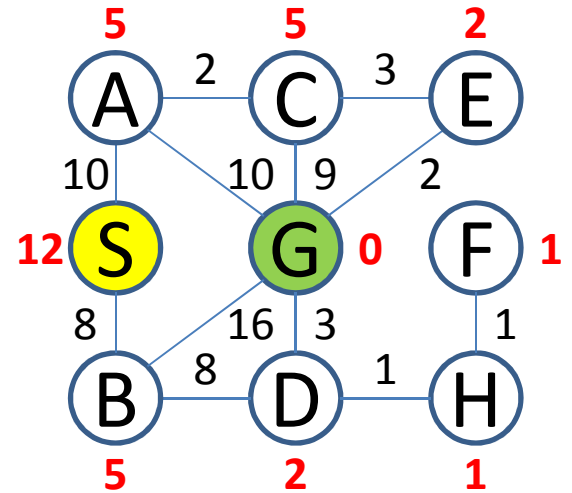
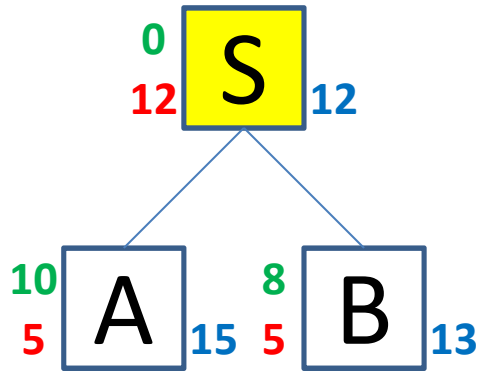
Problem



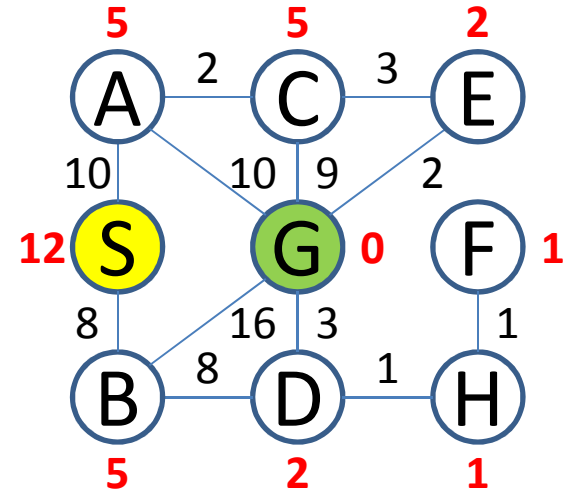
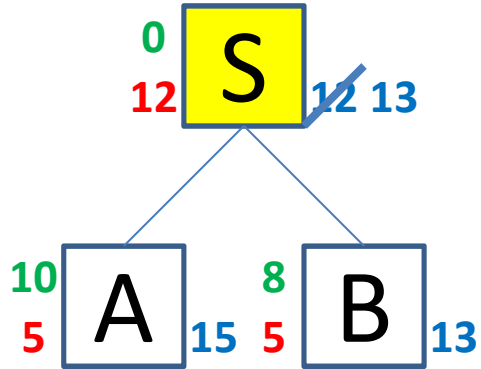
Problem



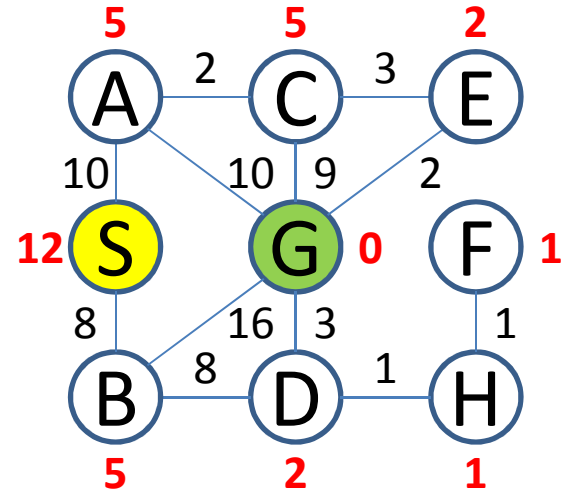
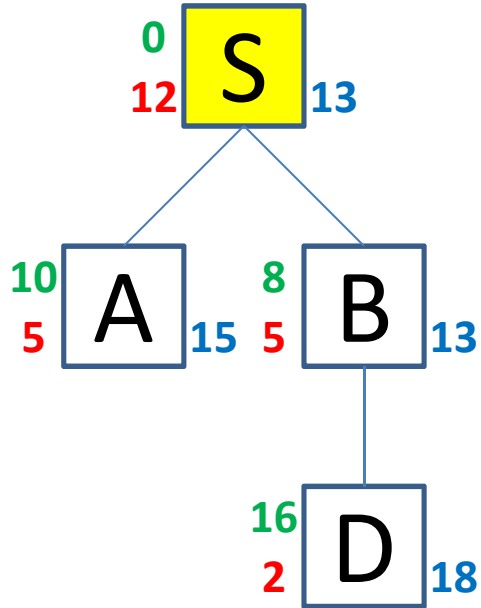
Problem



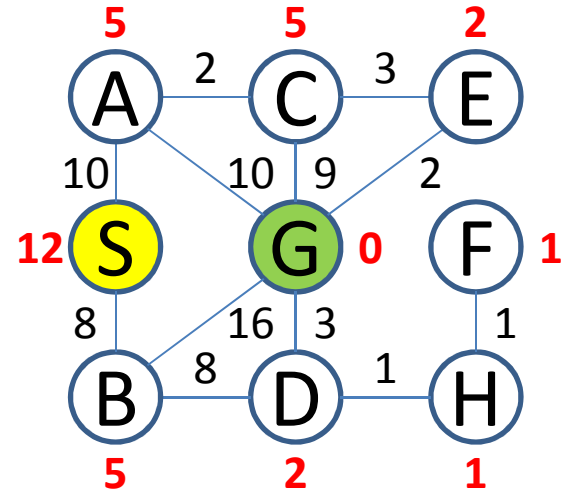
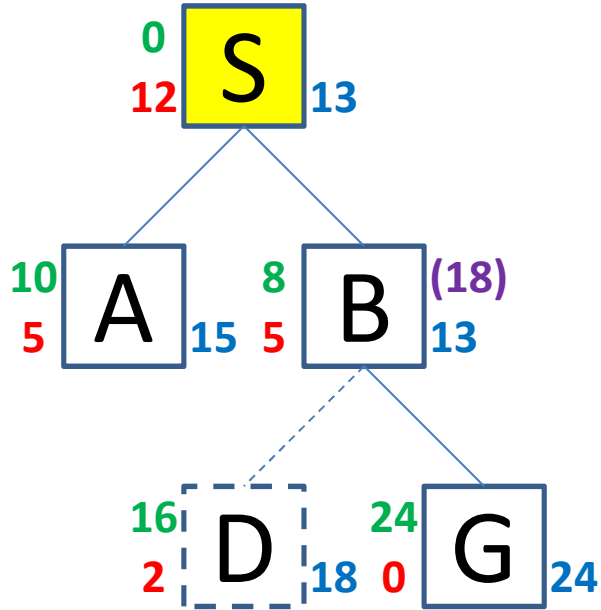
Problem



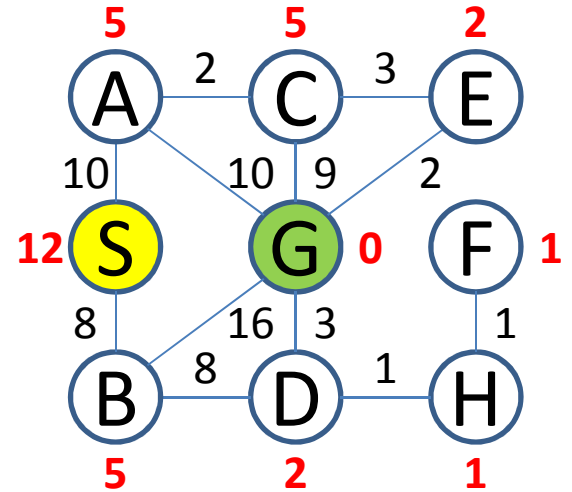
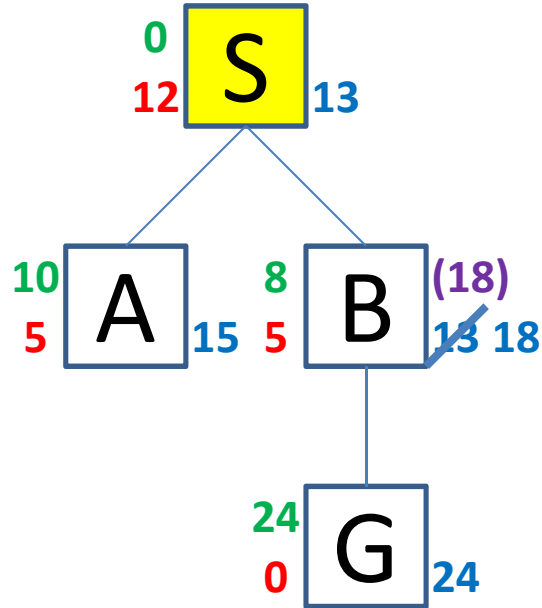
Problem



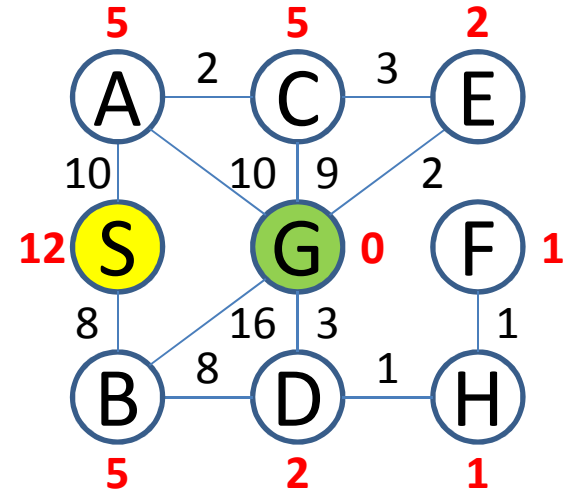
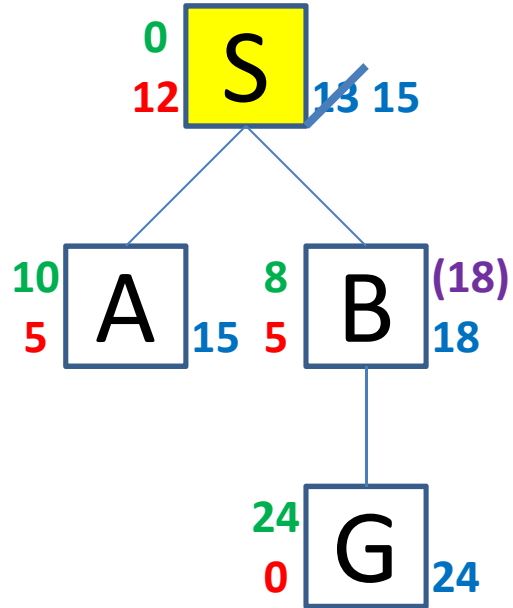
Problem



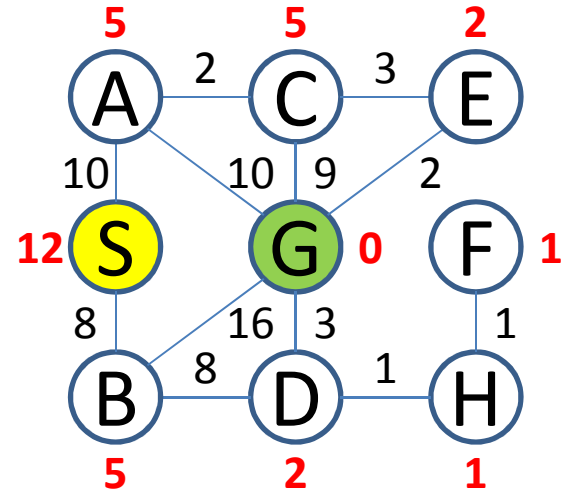
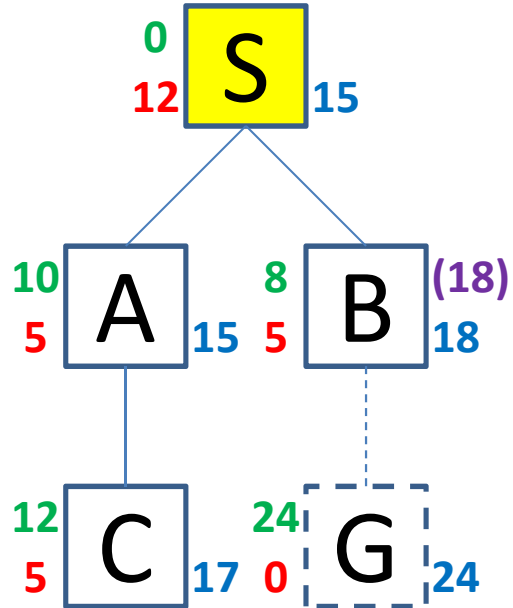
Problem



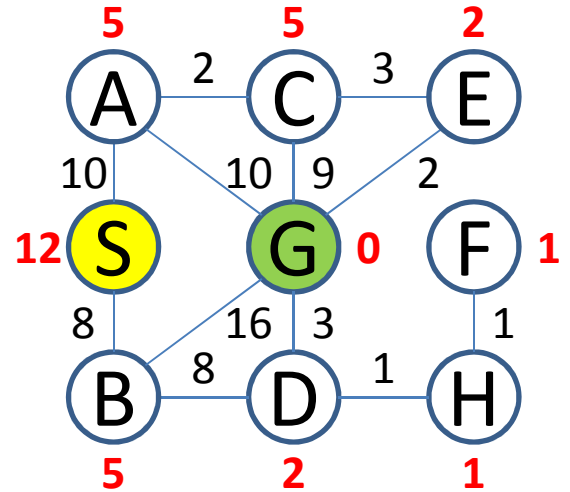
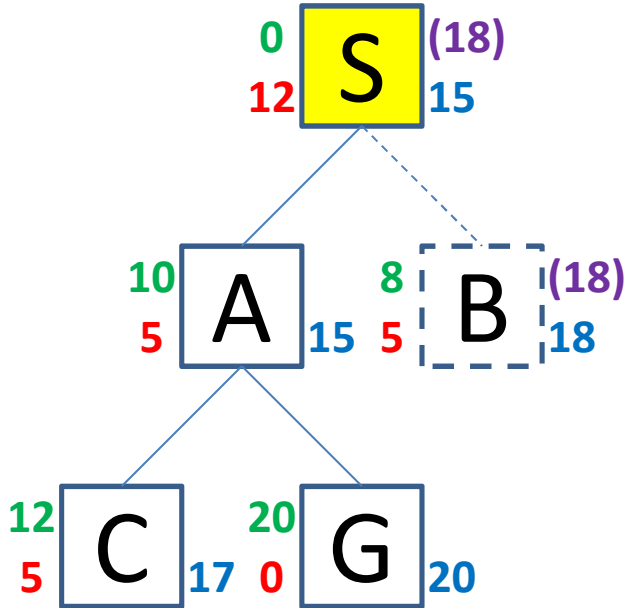
Problem



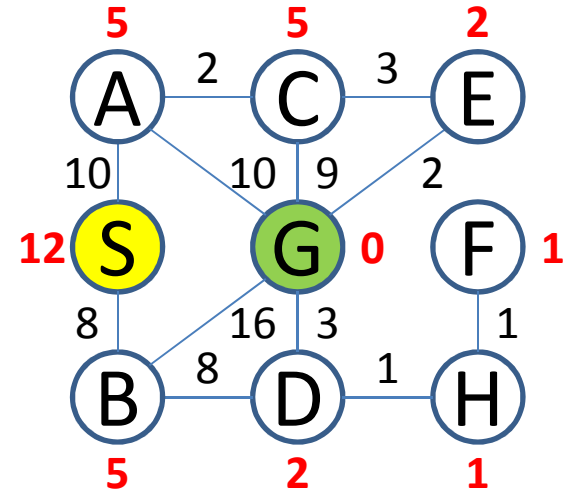
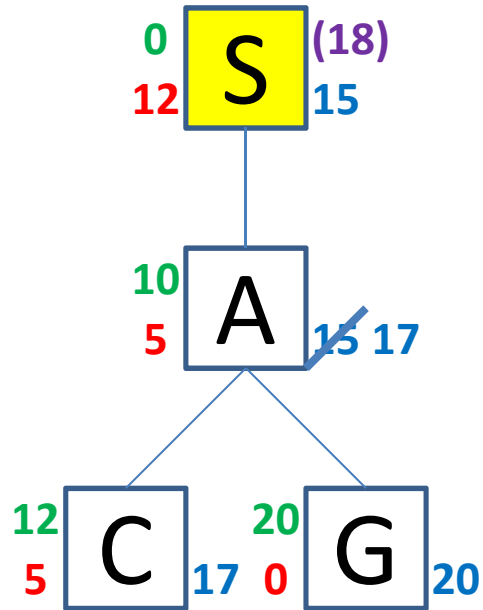
Problem



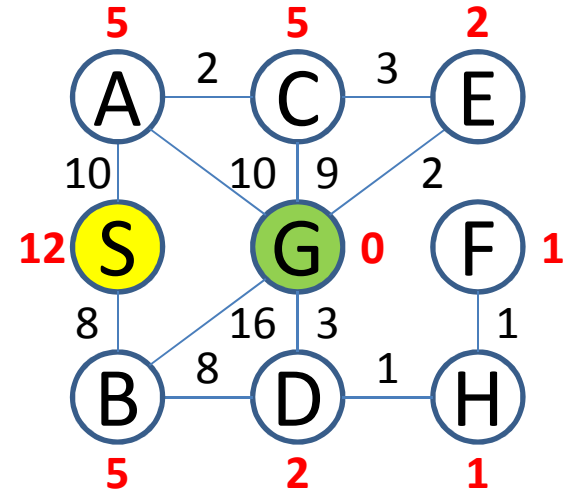
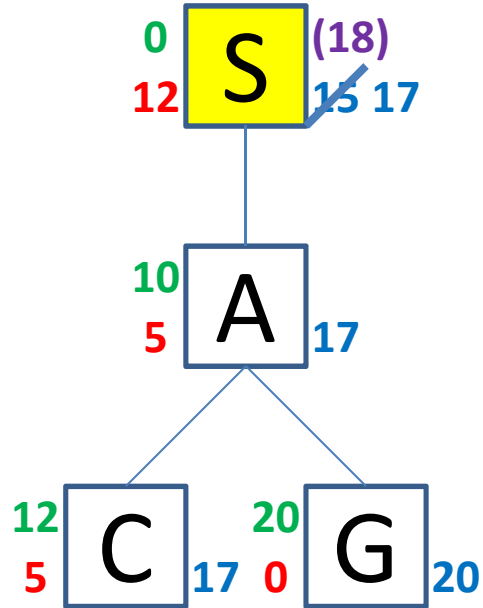
Problem



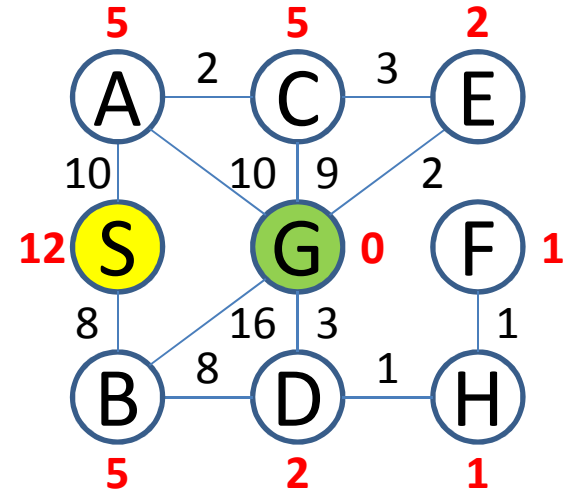
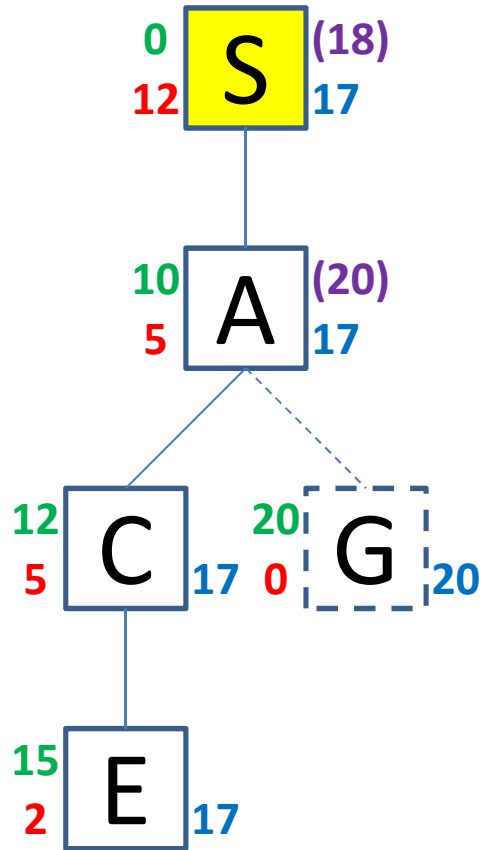
Problem



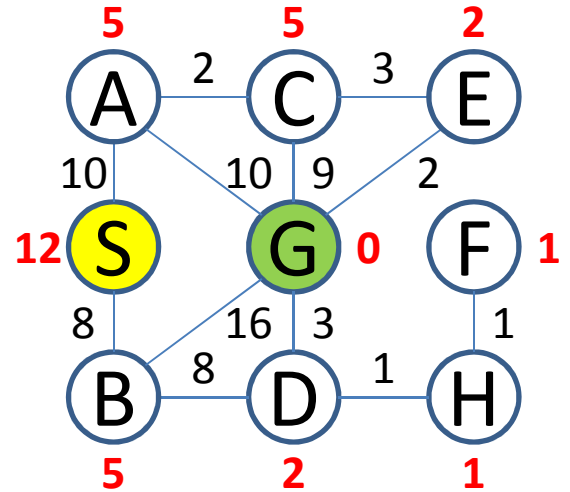
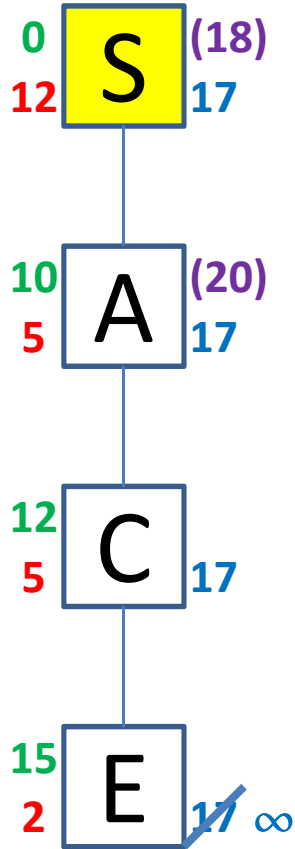
Problem



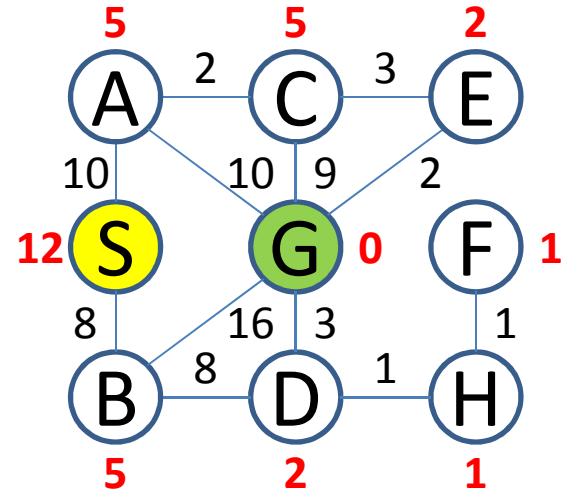
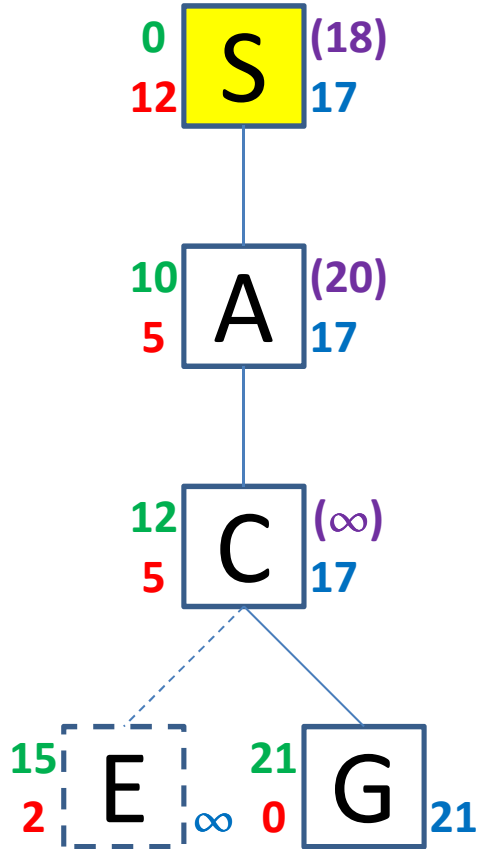
Problem



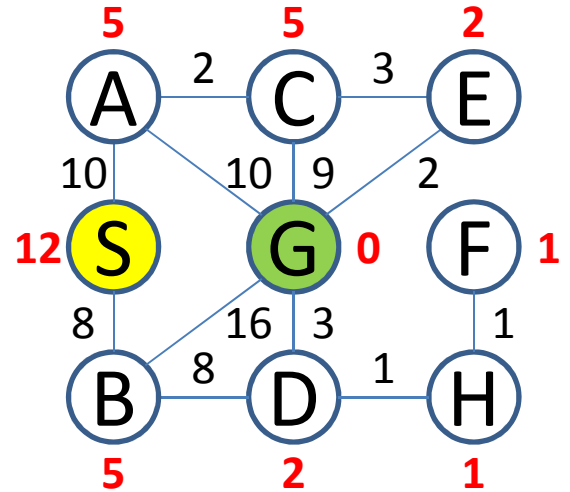
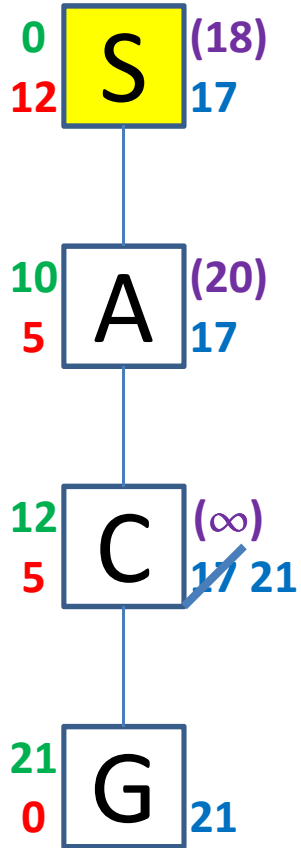
Problem



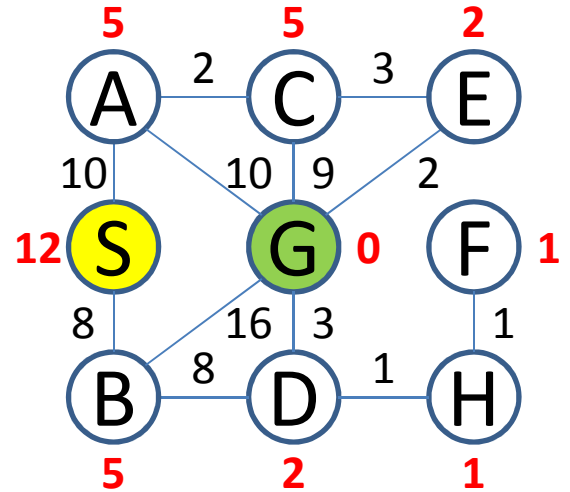
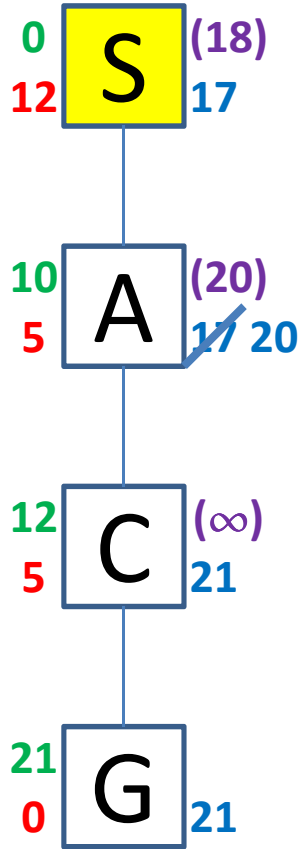
Problem



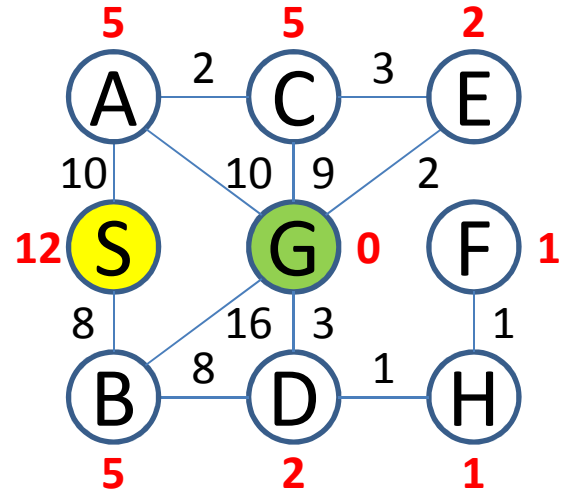
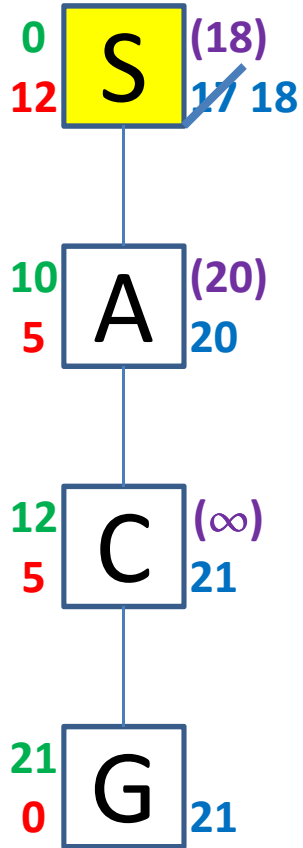
Problem



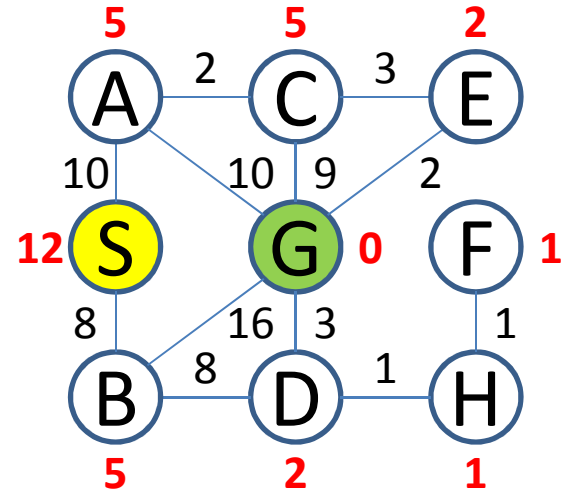
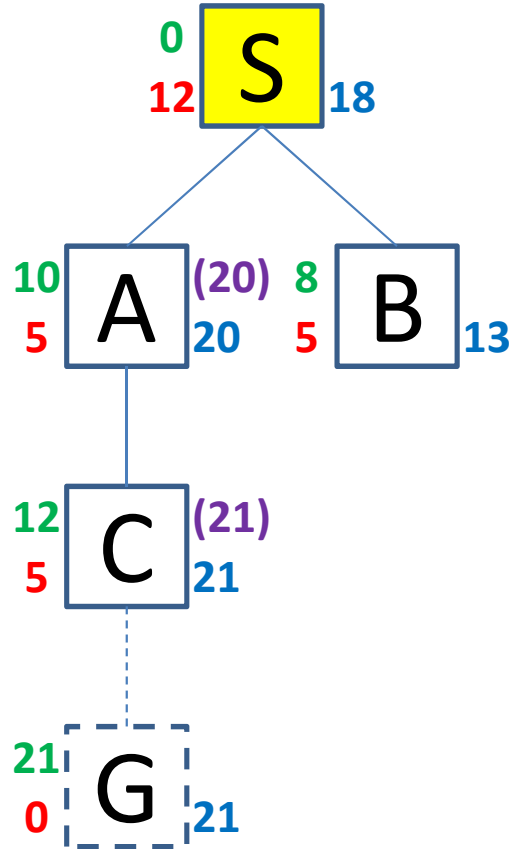
Problem



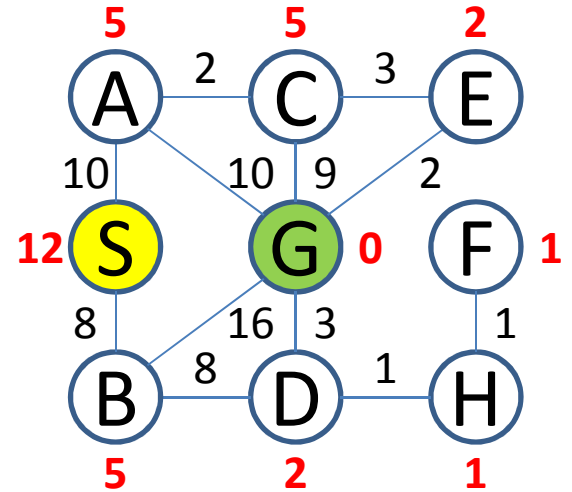
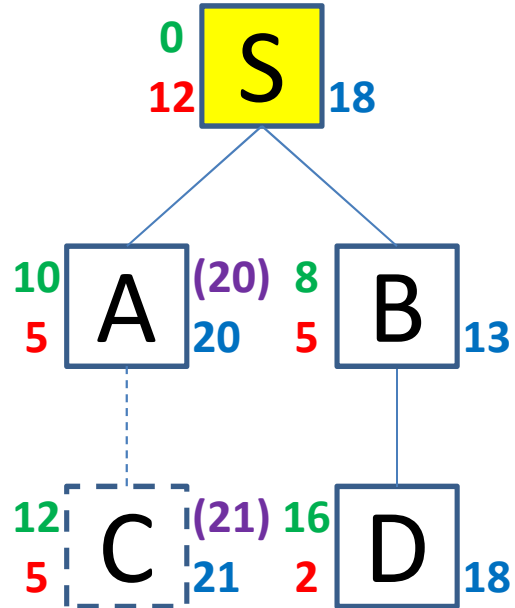
Problem



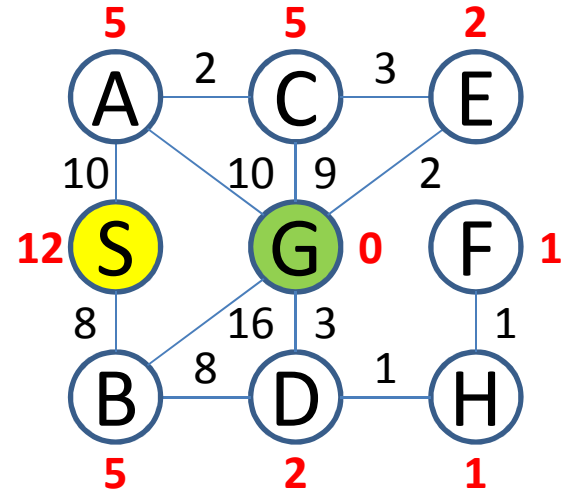
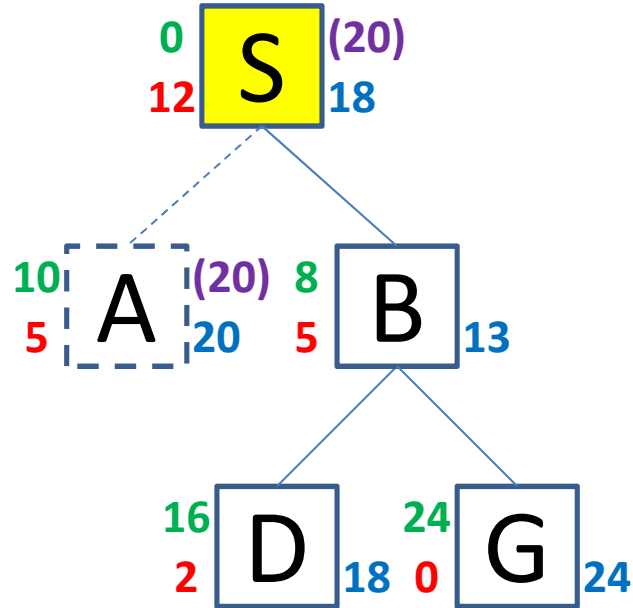
Problem



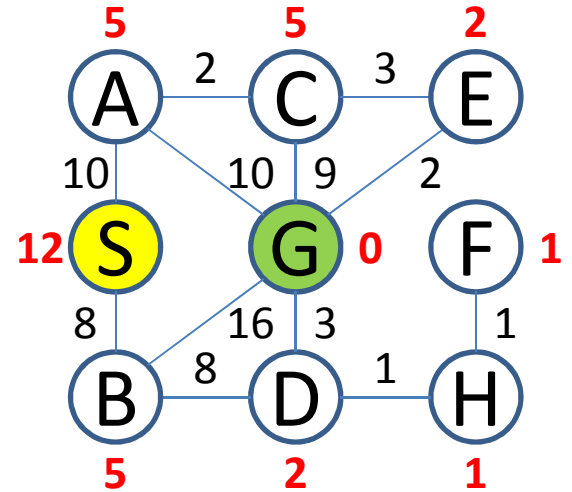
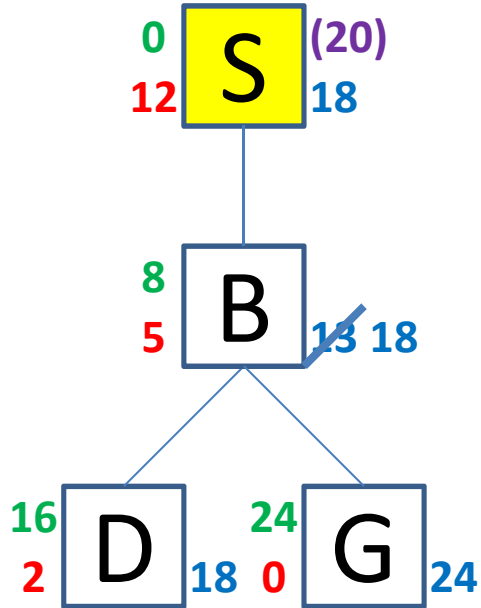
Problem



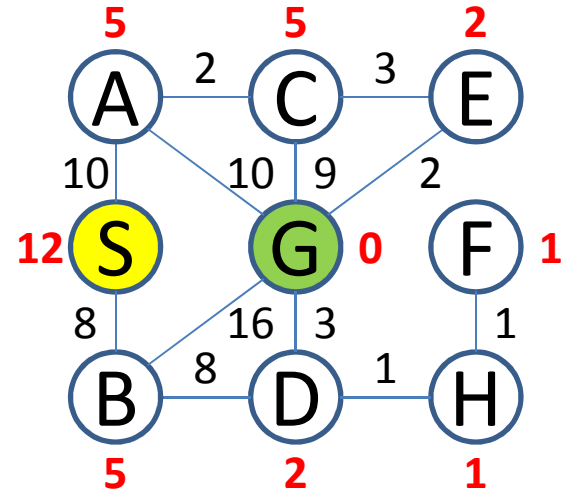
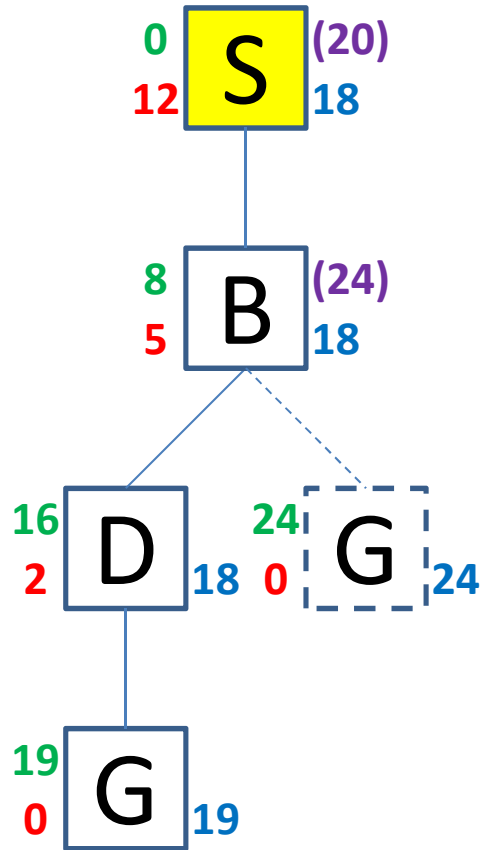
Problem



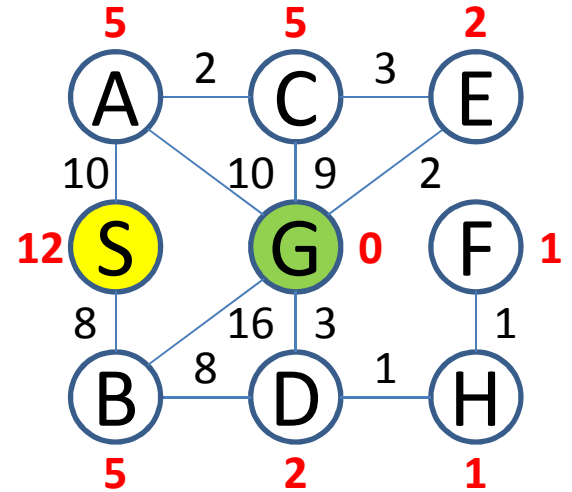
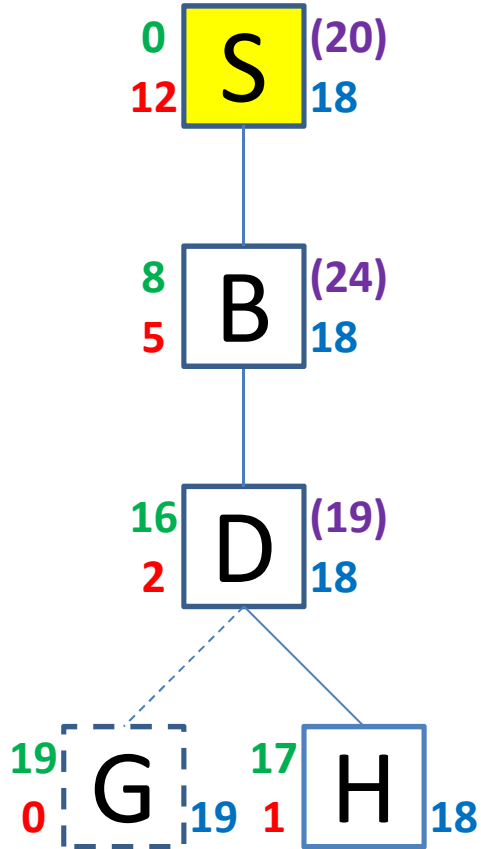
Problem



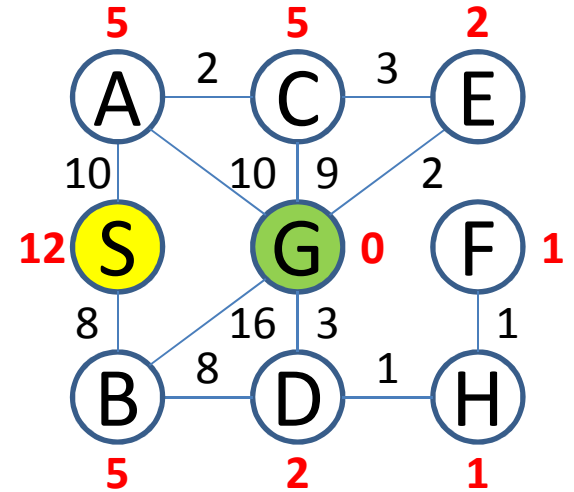
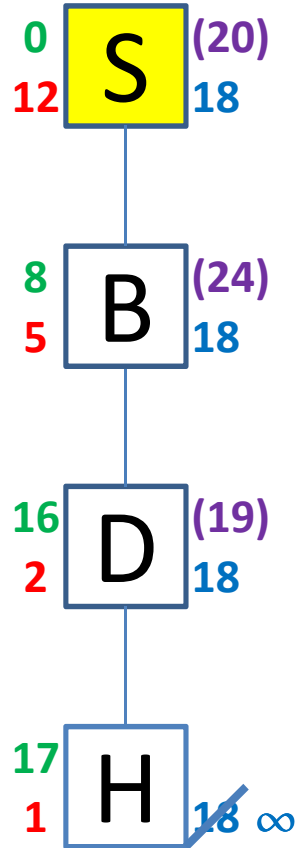
Problem



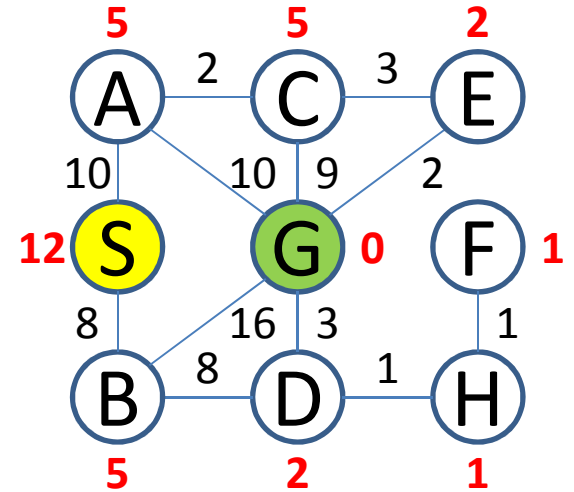
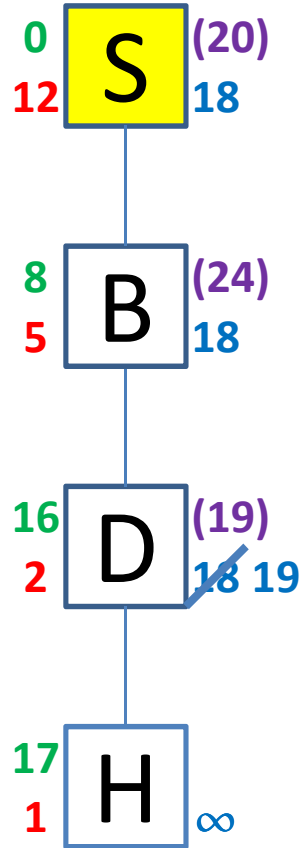
Problem



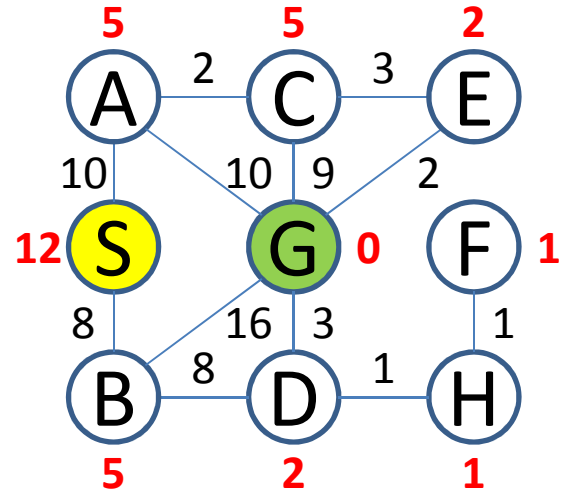
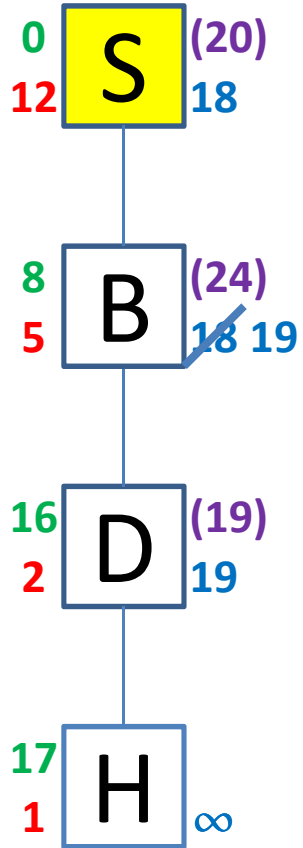
Problem



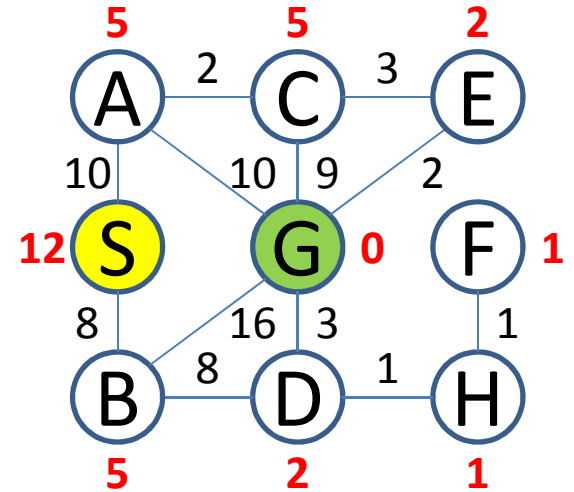
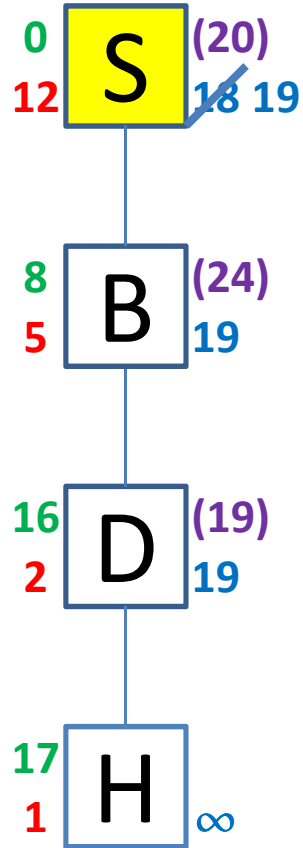
Problem



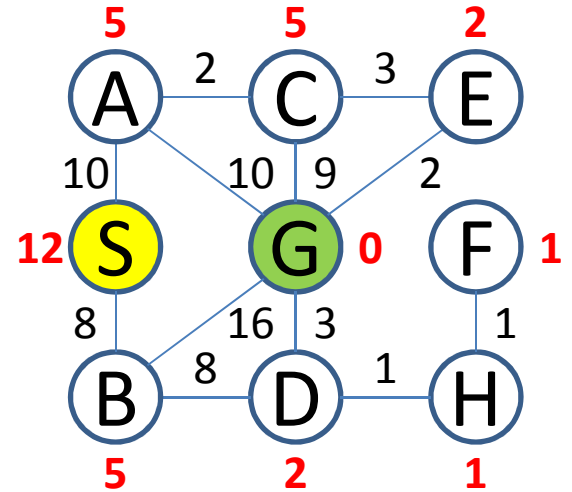
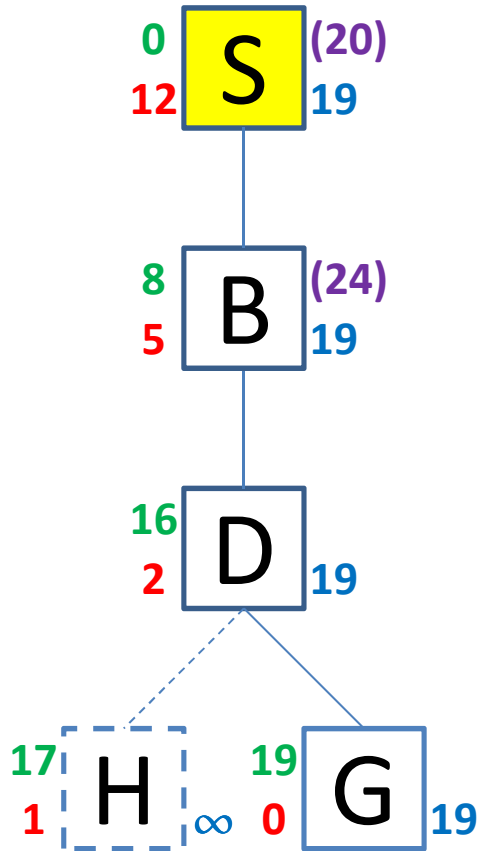
Problem



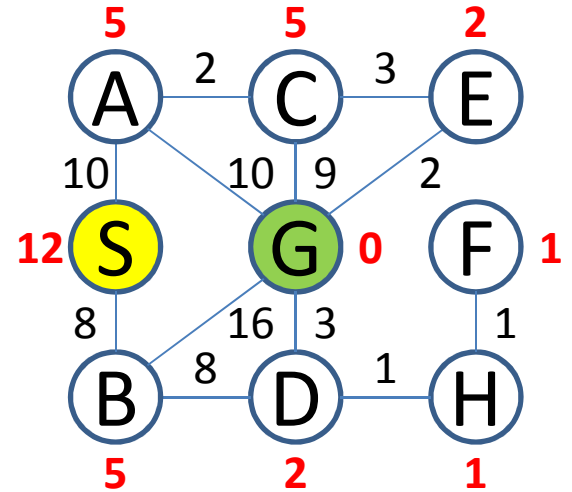
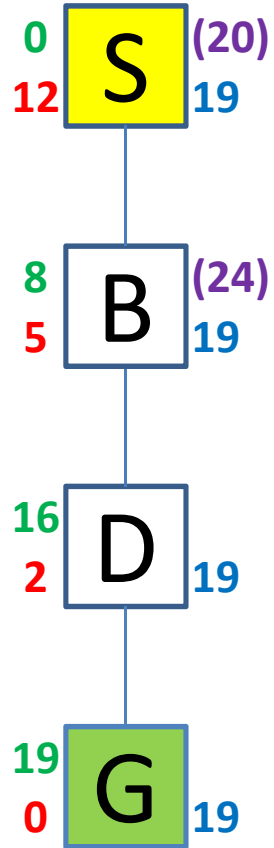
Problem



Problem



Problem



Exercises: Artificial Intelligence

Monotonicity 1

Monotonicity 1

PROBLEM

Problem

- Prove that:
 - **IF** a heuristic function h satisfies the *monotonicity restriction*
 - $h(x) \leq \text{cost}(x...y) + h(y)$
 - **THEN** f is *monotonously non-decreasing*
 - $f(s...x) \leq f(s...x...y)$

Monotonicity 1

- *Given:*
 - *h* satisfies the *monotonicity restriction*
- *Proof:*
$$f(S...A) = \text{cost}(S...A) + \mathbf{h(A)}$$

Monotonicity 1

- *Given:*
 - *h* satisfies the *monotonicity restriction*
- *Proof:*
$$\begin{aligned} f(S...A) &= \text{cost}(S...A) + h(A) \\ &\leq \text{cost}(S...A) + \text{cost}(A...B) + h(B) \end{aligned}$$

Monotonicity 1

- *Given:*
 - *h* satisfies the *monotonicity restriction*
- *Proof:*
$$\begin{aligned}f(S...A) &= \text{cost}(S...A) + \mathbf{h(A)} \\ &\leq \text{cost}(S...A) + \mathbf{\text{cost}(A...B) + h(B)} \\ &\leq \text{cost}(S...A...B) + h(B)\end{aligned}$$

Monotonicity 1

- *Given:*
 - *h* satisfies the *monotonicity restriction*

- *Proof:*

$$\begin{aligned} f(S...A) &= \text{cost}(S...A) + h(A) \\ &\leq \text{cost}(S...A) + \mathbf{\text{cost}(A...B)} + h(B) \\ &\leq \text{cost}(S...A...B) + h(B) \\ &\leq \mathbf{f(S...A...B)} \end{aligned}$$

Exercises: Artificial Intelligence

Monotonicity 2

Monotonicity 2

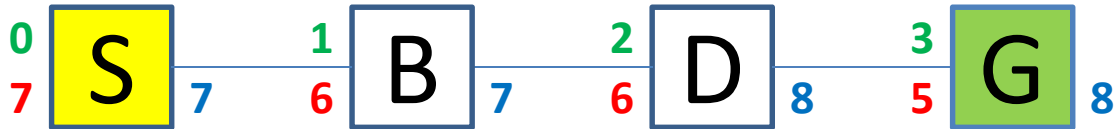
PROBLEM

Problem

- Prove or refute:
 - **IF** f is *monotonously non-decreasing*
 - $f(s...x) \leq f(s...xy)$
 - **THEN** h is an *admissible heuristic*
 - h is an underestimate of the remaining path to the goal with the smallest cost
- Can an extra constraint on h change this?

Monotonicity 2

- *Given:*
 - *f* is monotonously non-decreasing
- *Proof (Counter-example):*



f is monotonously non-decreasing,
yet *h* is not an admissible heuristic.

Monotonicity 2

- *Given:*
 - f is monotonously non-decreasing
 - Extra constraint: $h(G) = 0$
- *Proof:*
 - $f(S...A) \leq f(S...AB) \leq \dots \leq f(S...AB...G)$

Monotonicity 2

- *Given:*
 - f is monotonously non-decreasing
 - Extra constraint: $h(G) = 0$
- *Proof:*
 - $f(S...A) \leq f(S...AB) \leq \dots \leq f(S...AB...G)$ \Leftrightarrow
 $f(S...A) \leq f(S...G)$

Monotonicity 2

- *Given:*
 - f is monotonously non-decreasing
 - Extra constraint: $h(G) = 0$

- *Proof:*

$$\underline{f(S...A) \leq f(S...AB) \leq \dots \leq f(S...AB...G)} \Leftrightarrow$$

$$f(S...A) \leq f(S...G) \Leftrightarrow$$

$$\text{cost}(S...A) + h(A) \leq \text{cost}(S...G) + h(G)$$

Monotonicity 2

- *Given:*

- f is monotonously non-decreasing

- Extra constraint: $h(G) = 0$

- *Proof:*

$$\underline{f(S...A) \leq f(S...AB) \leq \dots \leq f(S...AB...G)} \Leftrightarrow$$

$$f(S...A) \leq f(S...G) \Leftrightarrow$$

$$\text{cost}(S...A) + h(A) \leq \text{cost}(S...G) + h(G) \Leftrightarrow$$

$$\underline{\text{cost}(S...A)} + h(A) \leq \underline{\text{cost}(S...A)} + \text{cost}(A...G) + h(G)$$

Monotonicity 2

- *Given:*

- f is monotonously non-decreasing

- Extra constraint: $h(G) = 0$

- *Proof:*

$$\underline{f(S...A) \leq f(S...AB) \leq \dots \leq f(S...AB...G)} \Leftrightarrow$$

$$f(S...A) \leq f(S...G) \Leftrightarrow$$

$$\text{cost}(S...A) + h(A) \leq \text{cost}(S...G) + h(G) \Leftrightarrow$$

$$\underline{\text{cost}(S...A)} + h(A) \leq \underline{\text{cost}(S...A)} + \text{cost}(A...G) + h(G) \Leftrightarrow$$

$$h(A) \leq \text{cost}(A...G) + \mathbf{h(G)}$$

Monotonicity 2

- *Given:*

- f is monotonously non-decreasing

- Extra constraint: $h(G) = 0$

- *Proof:*

$$\underline{f(S...A) \leq f(S...AB) \leq \dots \leq f(S...AB...G)} \Leftrightarrow$$

$$f(S...A) \leq f(S...G) \Leftrightarrow$$

$$\text{cost}(S...A) + h(A) \leq \text{cost}(S...G) + h(G) \Leftrightarrow$$

$$\underline{\text{cost}(S...A)} + h(A) \leq \underline{\text{cost}(S...A)} + \text{cost}(A...G) + h(G) \Leftrightarrow$$

$$h(A) \leq \text{cost}(A...G) + \underline{h(G)} \Leftrightarrow$$

$$h(A) \leq \text{cost}(A...G)$$