Exercises: Artificial Intelligence

A*

A*

A* ALGORITHM

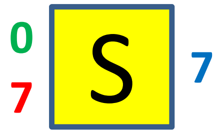
A* Algorithm

- ***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 = cost + heuristic)
    - **IF** QUEUE contains paths: P, Q
      AND P ends in node N_i && Q contains node N_i
      AND cost_P ≥ cost_Q
      THEN remove P
  - **IF** goal reached **THEN** success **ELSE** failure

A*

FIRST EXAMPLE ON A*

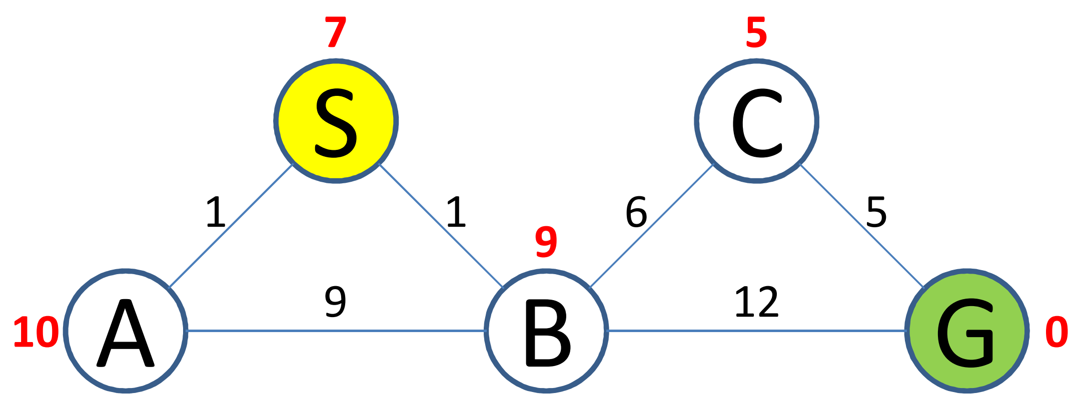
A* algorithm by Example



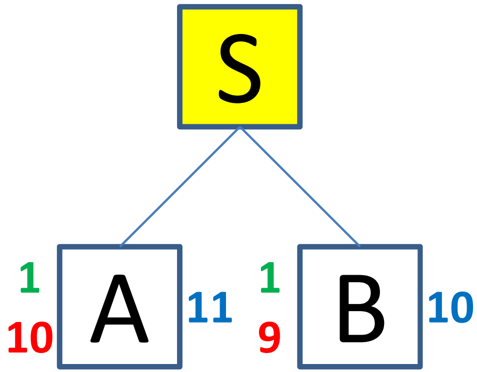
f = accumulated path cost + heuristic

<u>QUEUE</u> = <u>***path containing root***</u>

<u>QUEUE:</u> <S>



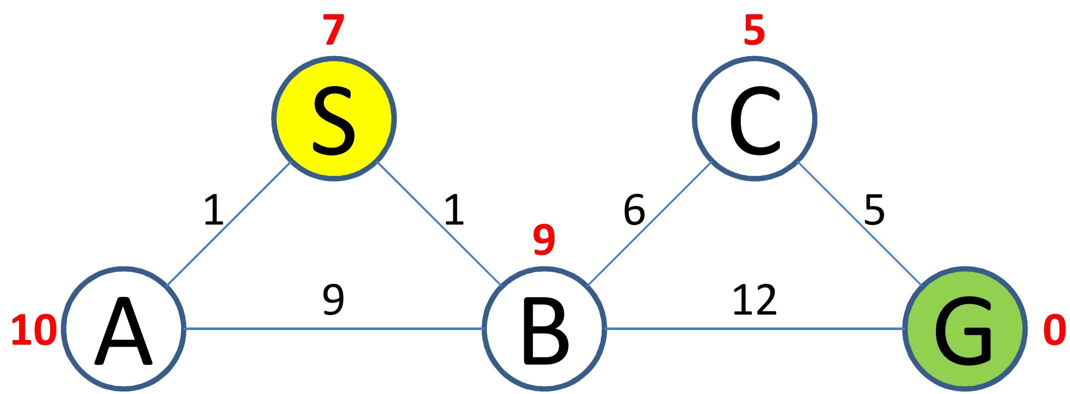
A* algorithm by Example



f = accumulated path cost + heuristic

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

QUEUE: <SB,SA>

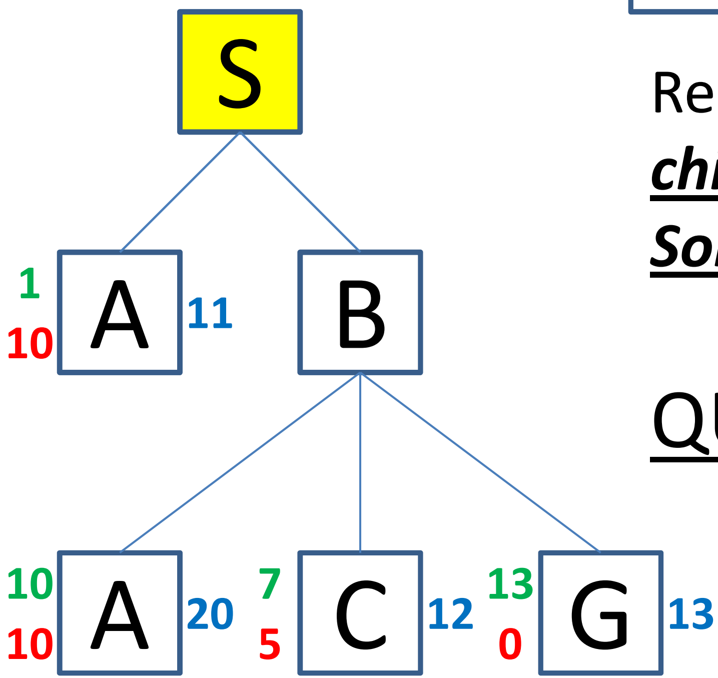


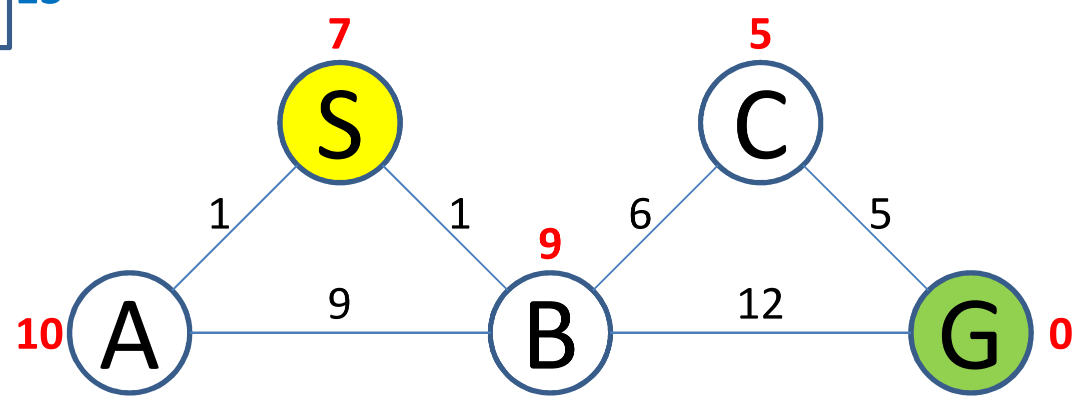
A* algorithm by Example

f = accumulated path cost + heuristic

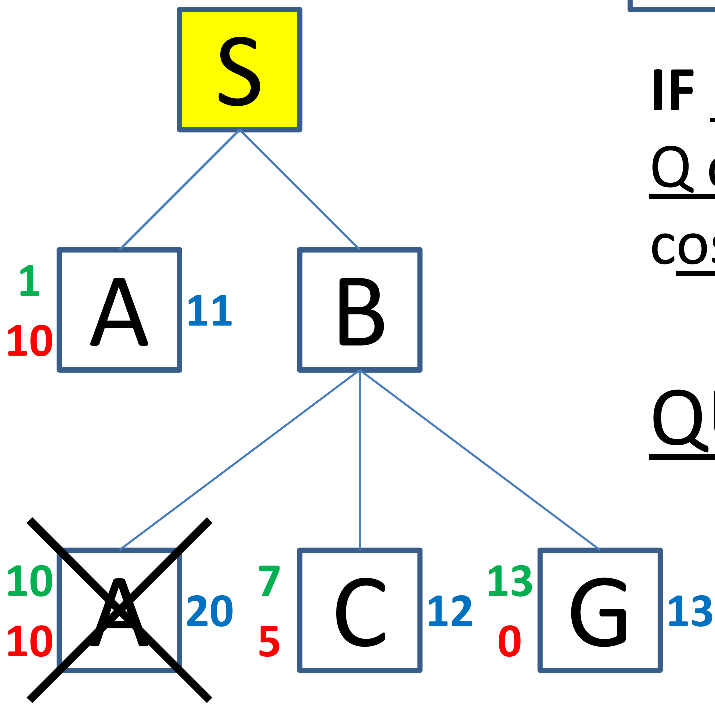
Remove ***first path***, Create ***paths to all children***, Reject ***loops*** and ***Add paths***. ***Sort*** QUEUE ***by f***

QUEUE: <SA,SBC,SBG,SBA>





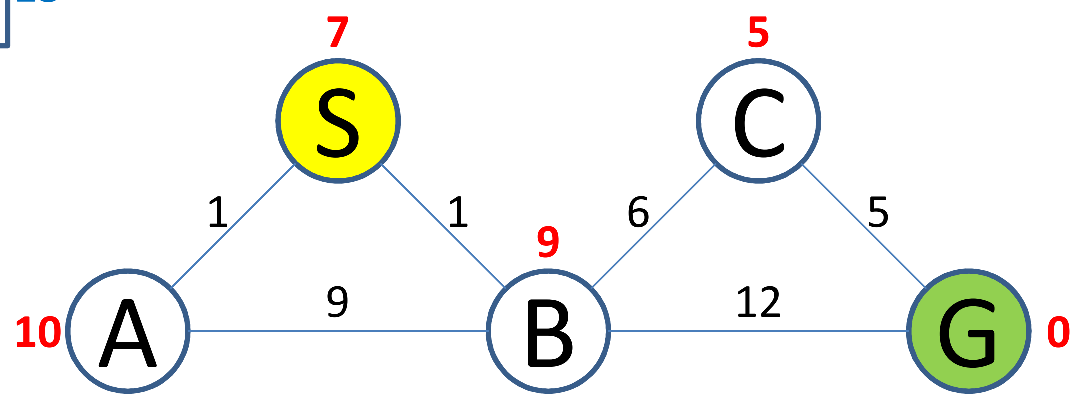
A* algorithm by Example



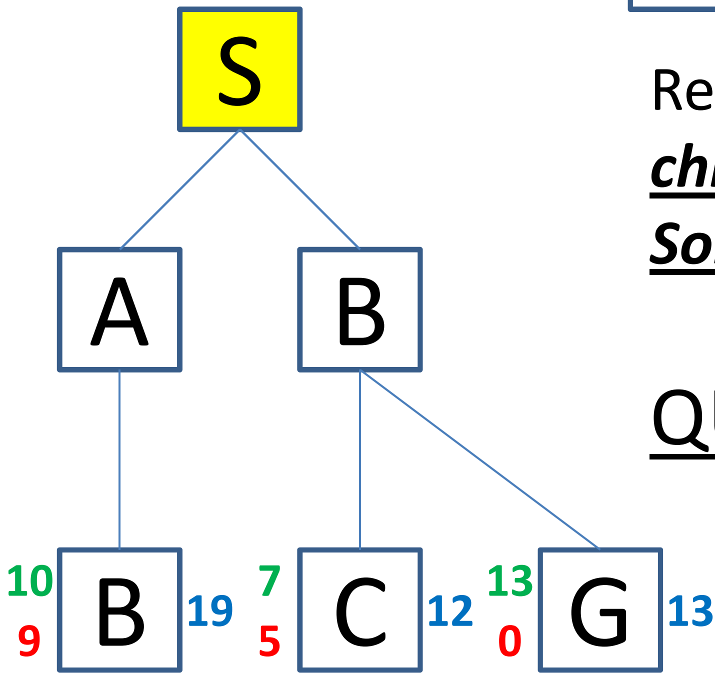
f = accumulated path cost + heuristic

IF P terminating in I with cost_P && Q containing I with cost_Q **AND** cost_P ≥ cost_Q **THEN** remove P

QUEUE: <SA,SBC,SBG,**SBA**>



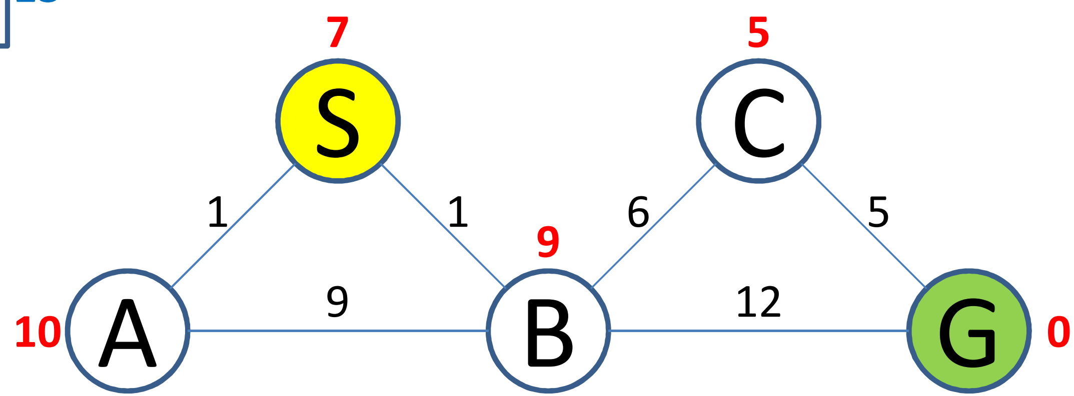
A* algorithm by Example



f = accumulated path cost + heuristic

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

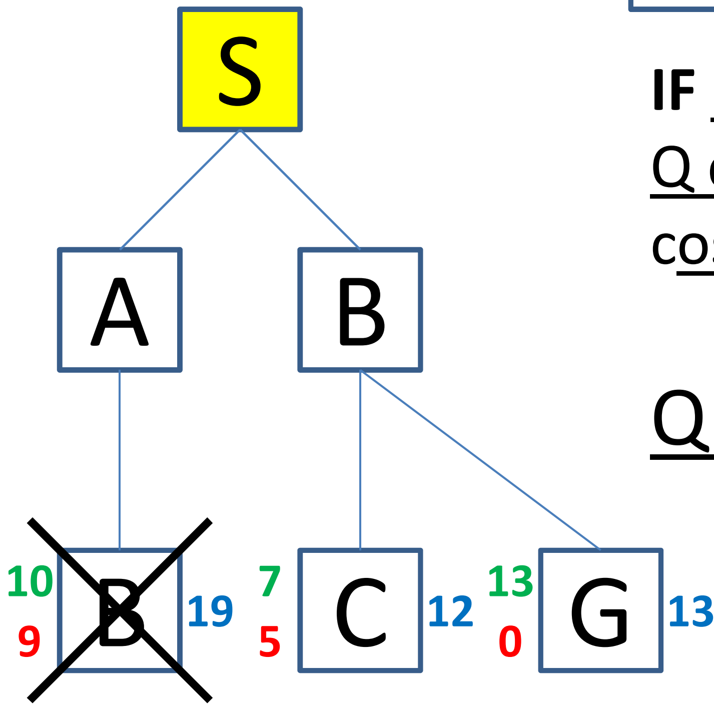
QUEUE: <SBC,SBG,SAB>



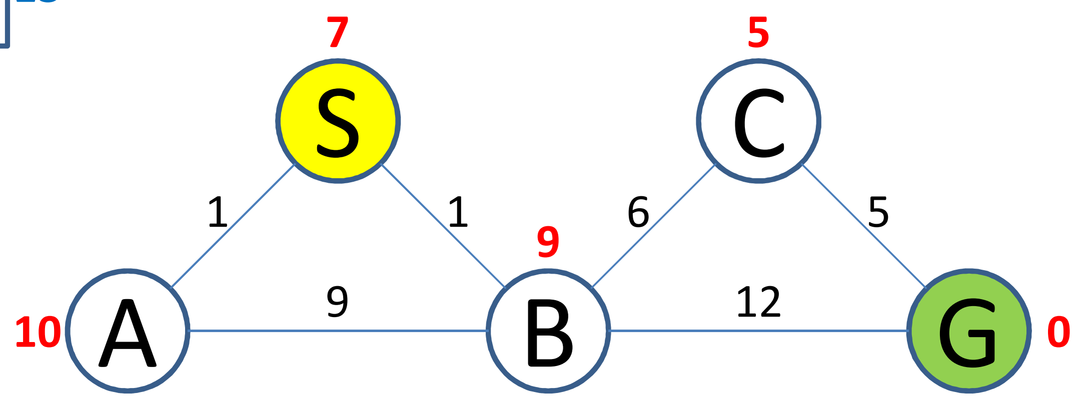
A* algorithm by Example

f = accumulated path cost + heuristic

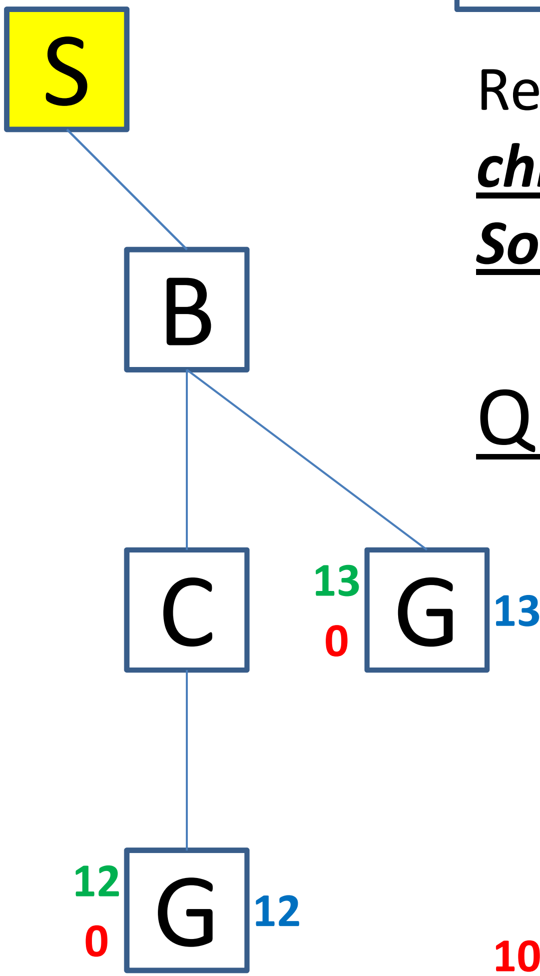
IF P terminating in I with cost_P && Q containing I with cost_Q **AND** cost_P ≥ cost_Q **THEN** remove P



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



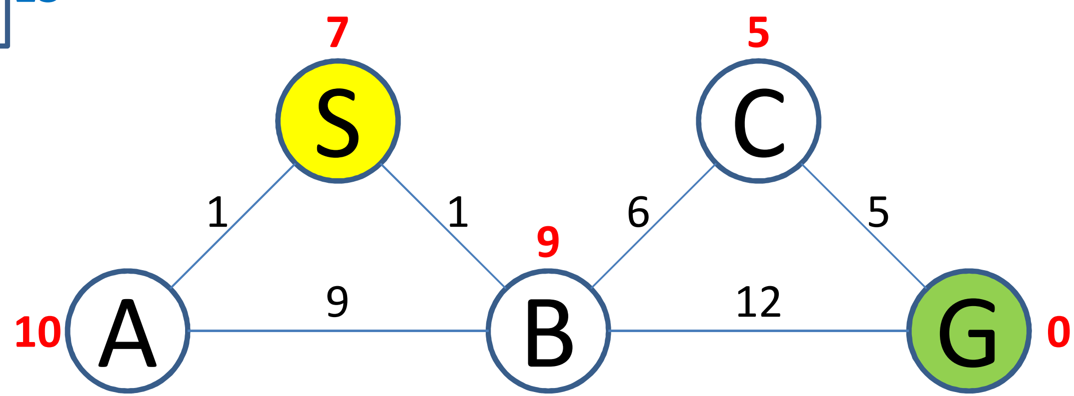
A* algorithm by Example



f = accumulated path cost + heuristic

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

QUEUE: <SBCG,SBG>

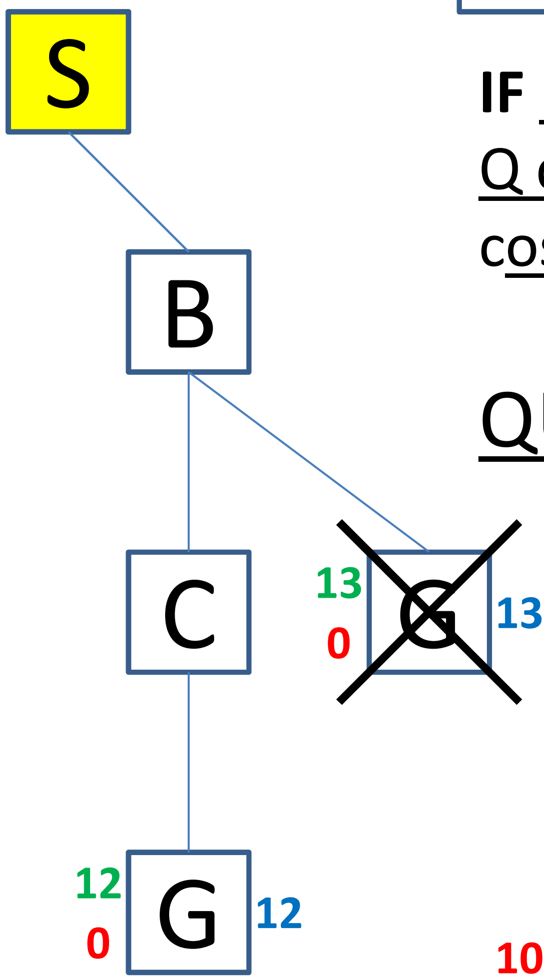


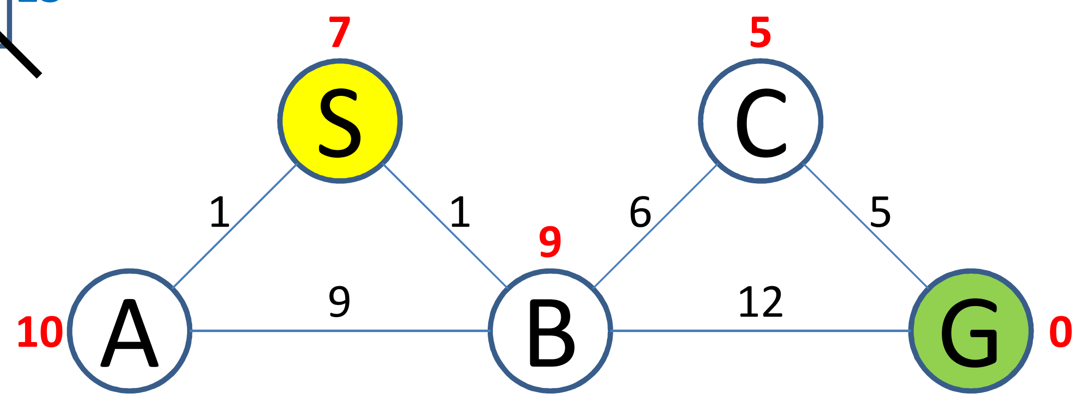
A* algorithm by Example

f = accumulated path cost + heuristic

IF P terminating in I with cost_P && Q containing I with cost_Q **AND** cost_P ≥ cost_Q **THEN** remove P

QUEUE: <SBCG,**SBG**>

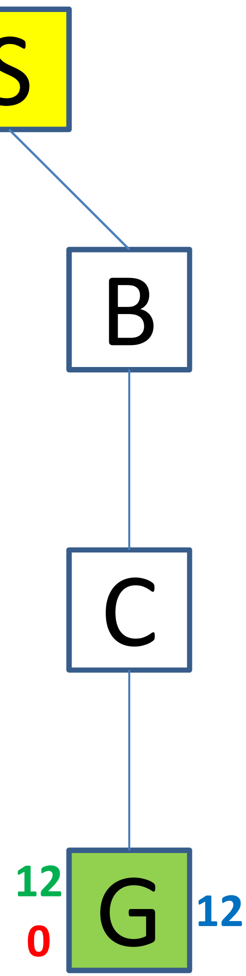




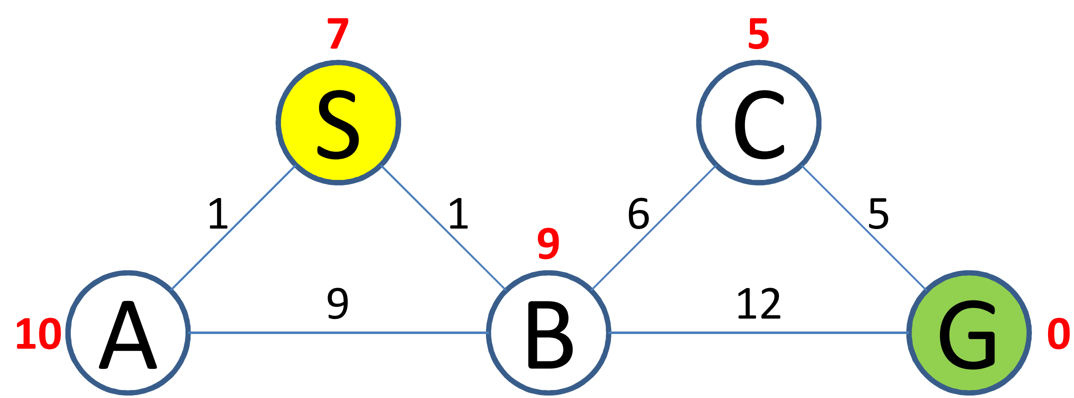
A* algorithm by Example

f = accumulated path cost + heuristic

SUCCESS



QUEUE: <SBCG>

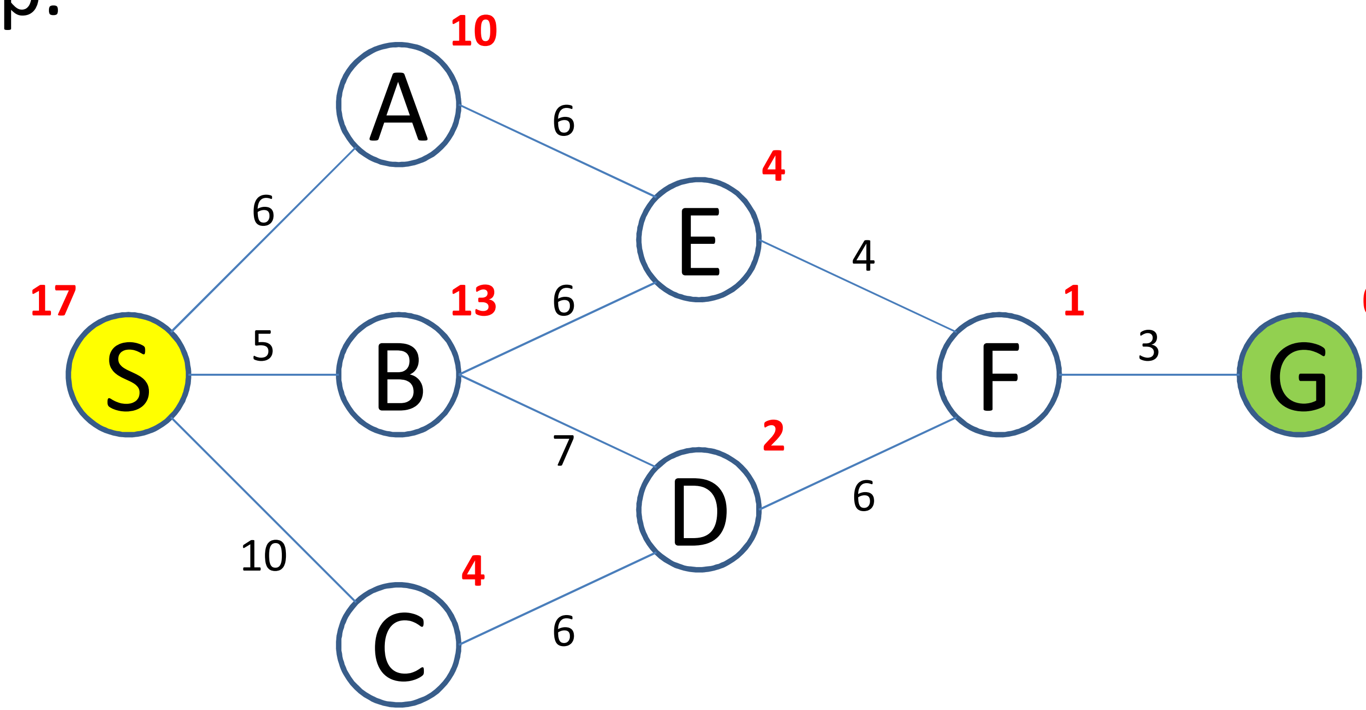


A*

PROBLEM

Problem

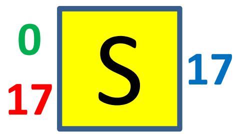
- Perform the A* Algorithm on the following figure. Explicitly write down the queue at each step.

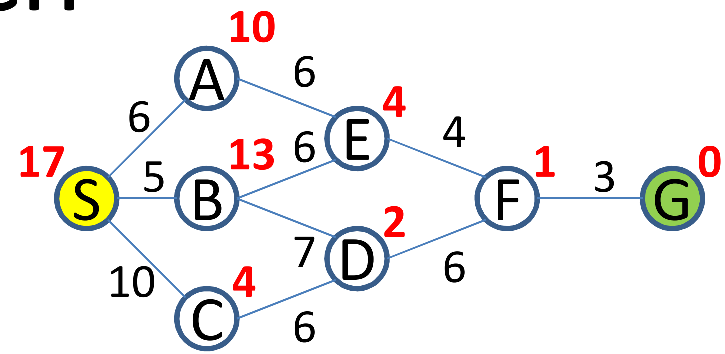


A*

A* SEARCH

A* Search

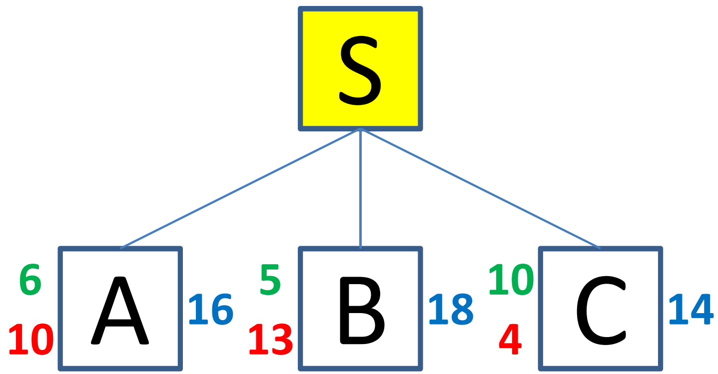


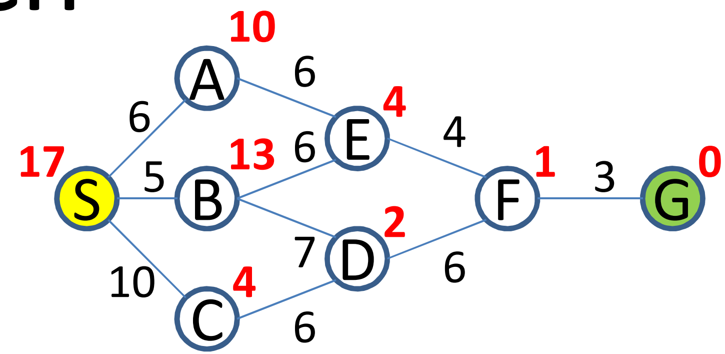


QUEUE:

S

A* Search





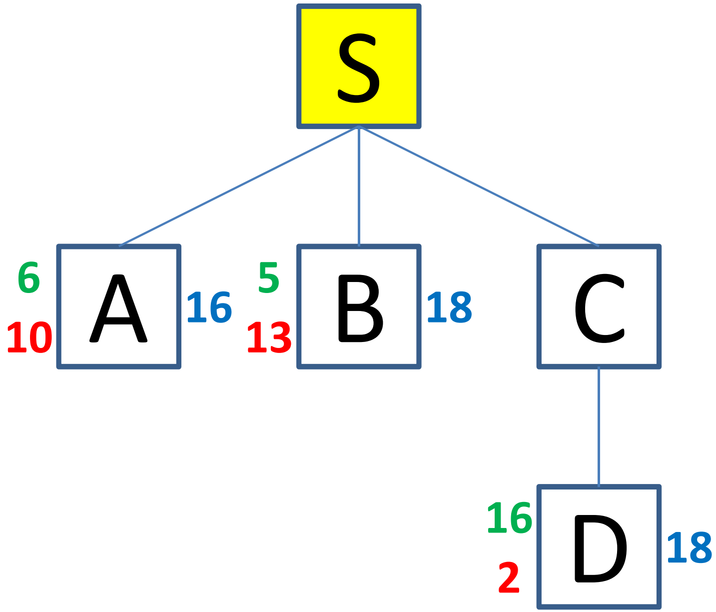
QUEUE:

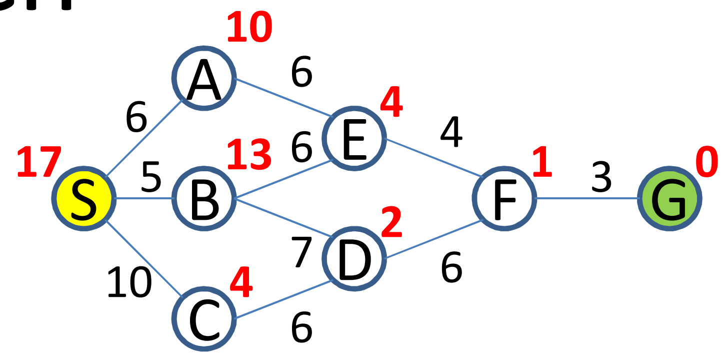
SC

SA

SB

A* Search





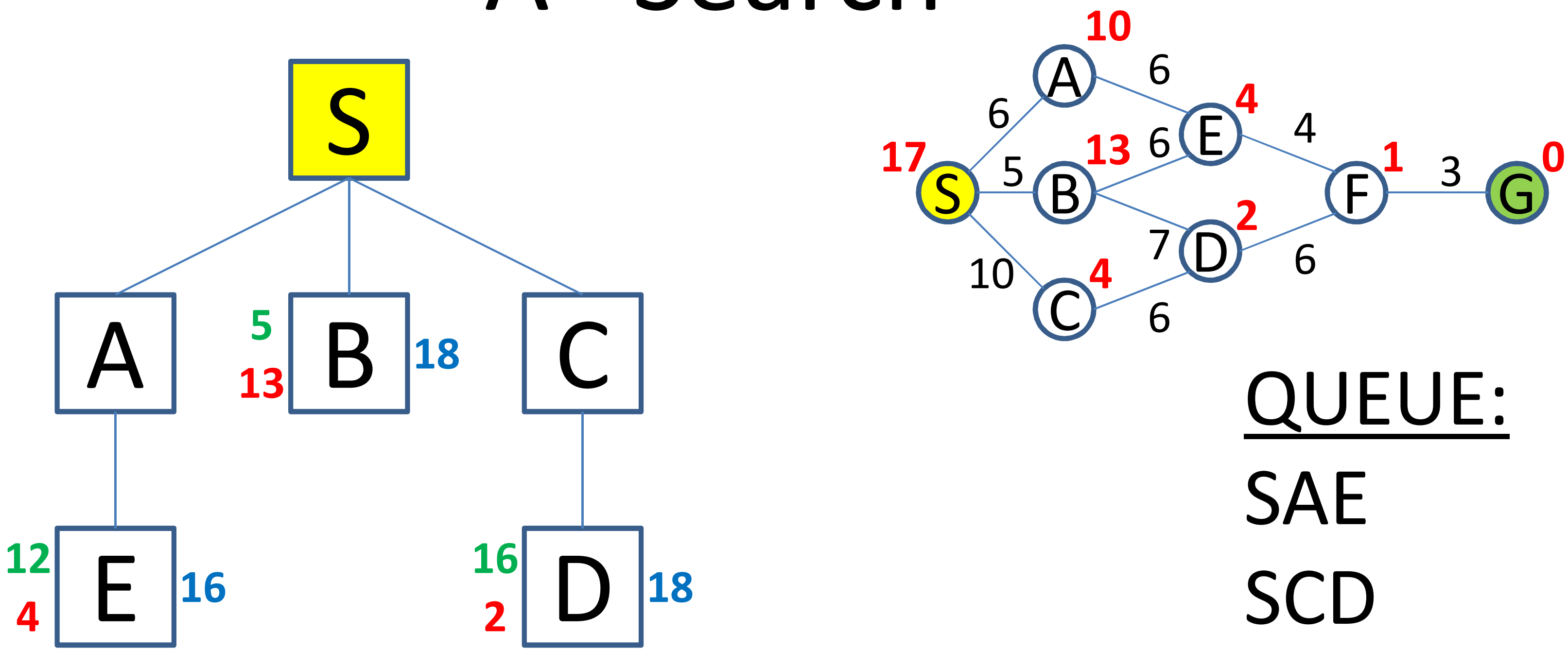
QUEUE:

SA

SCD

SB

A* Search



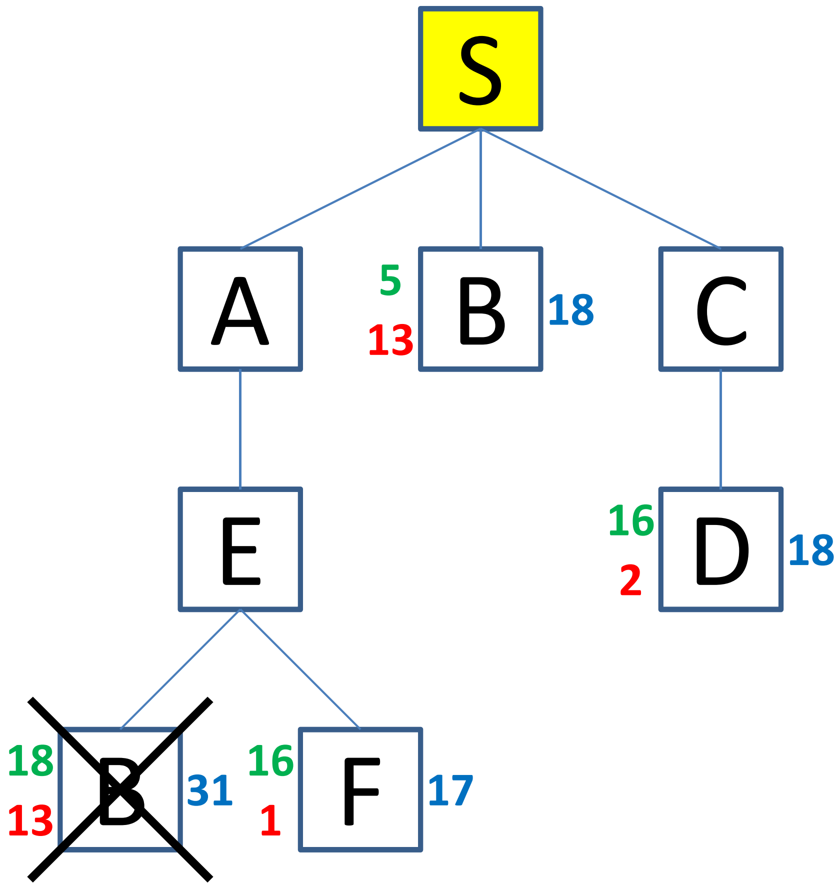
QUEUE:

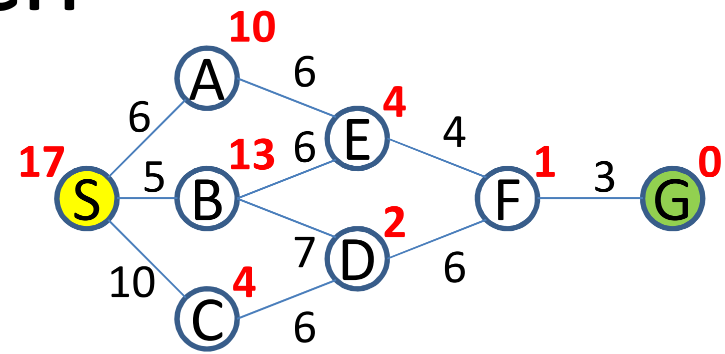
SAE

SCD

SB

A* Search





QUEUE:

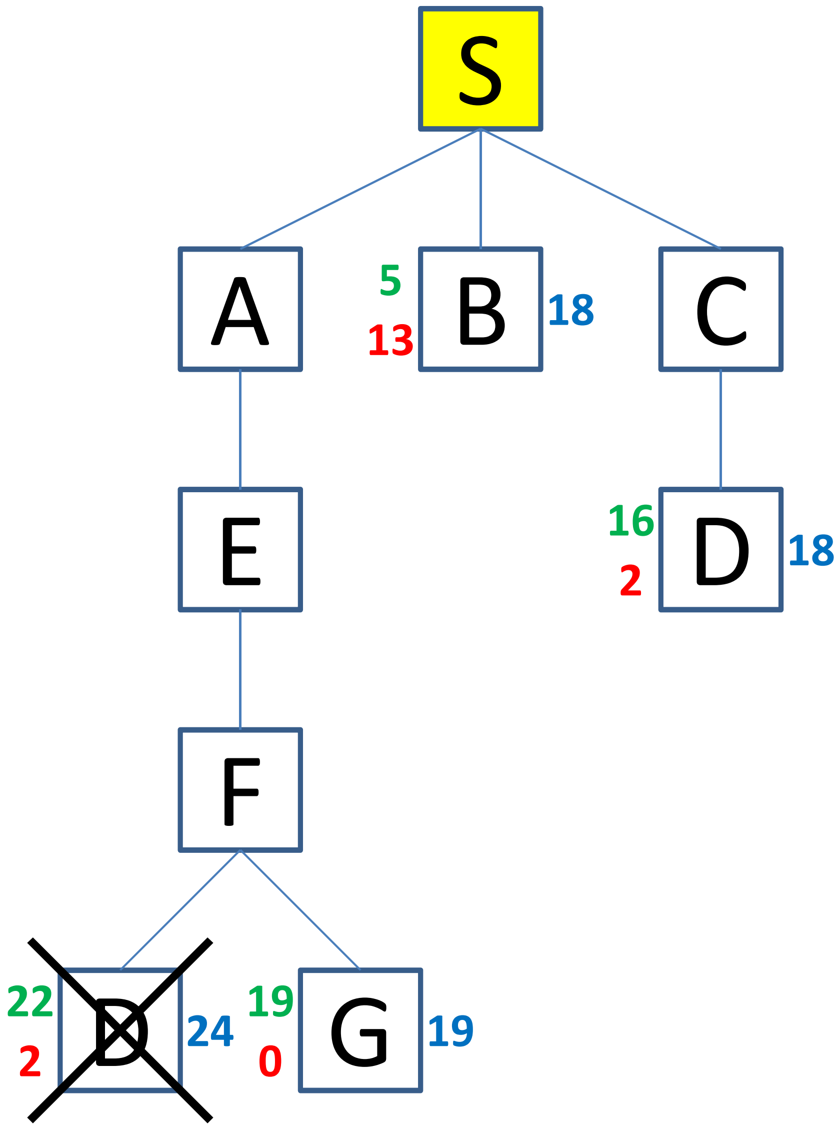
SAEF

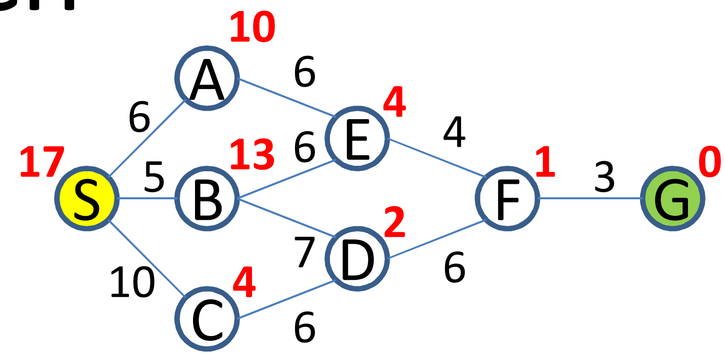
SCD

SB

SAEB

A* Search





<u>QUEUE:</u>

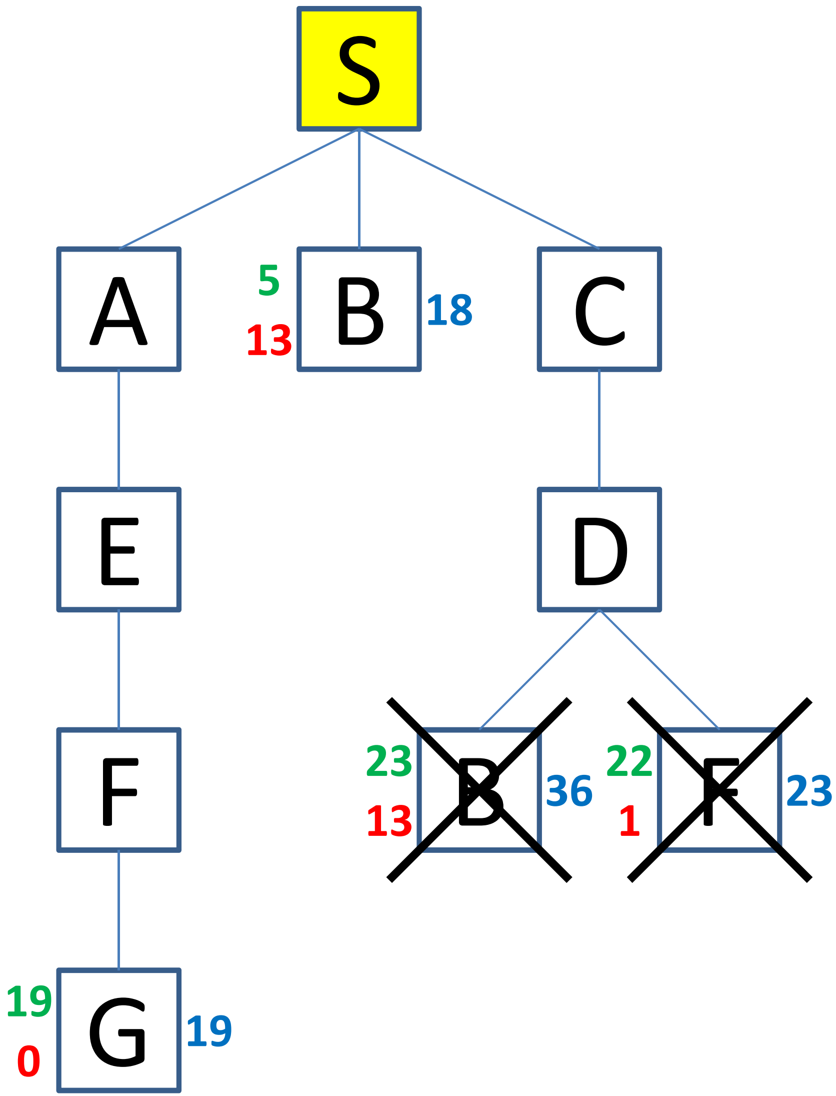
SCD

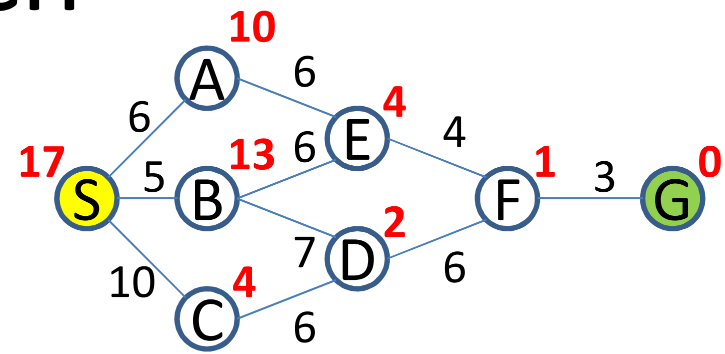
SB

SAEFG

<u>**SAEFD**</u>

A* Search





QUEUE:

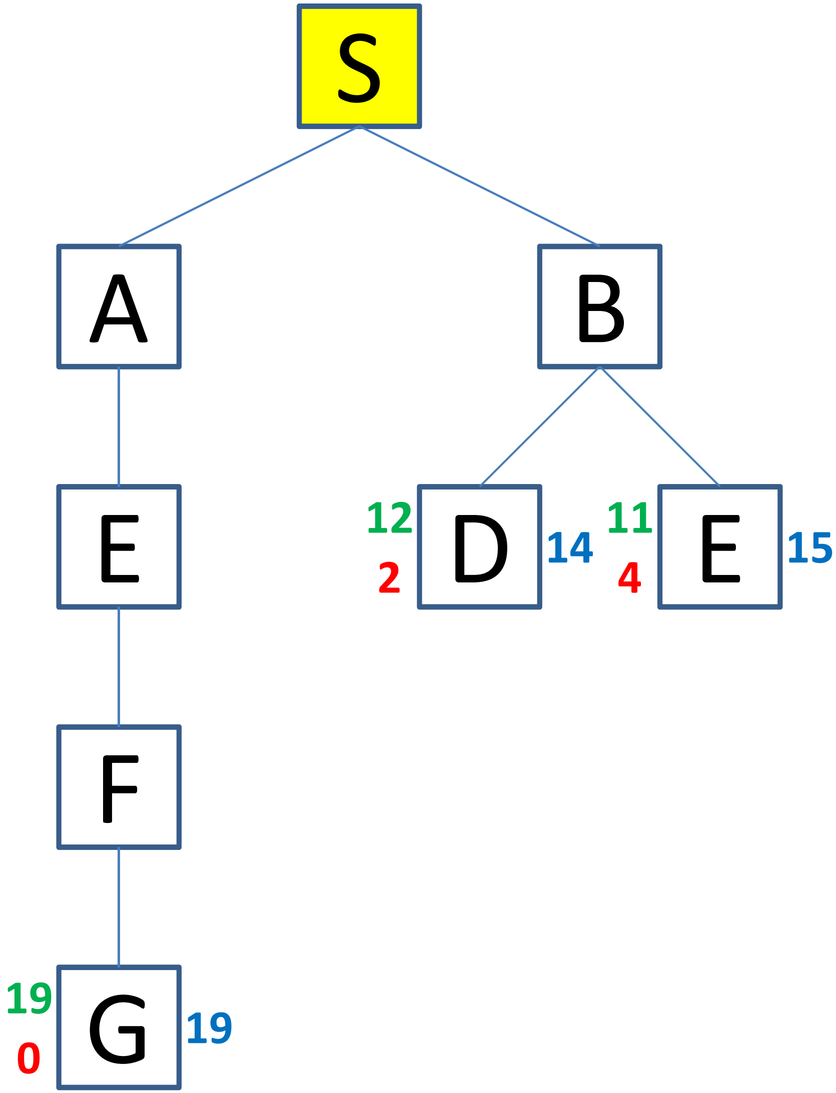
SB

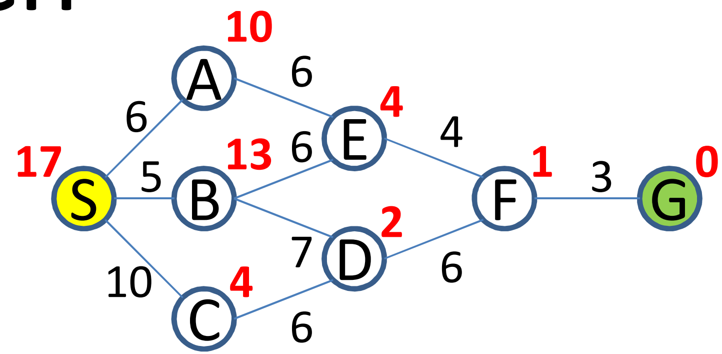
SAEFG

SCDF

SCDB

A* Search





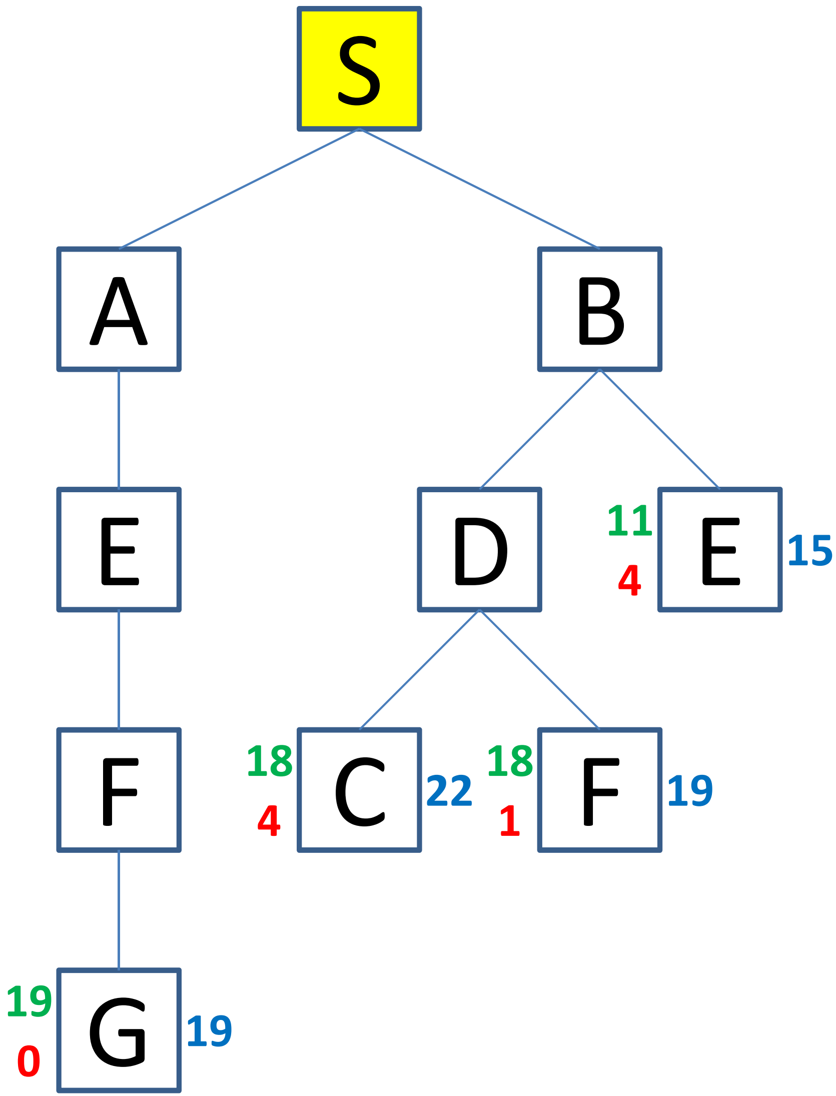
QUEUE:

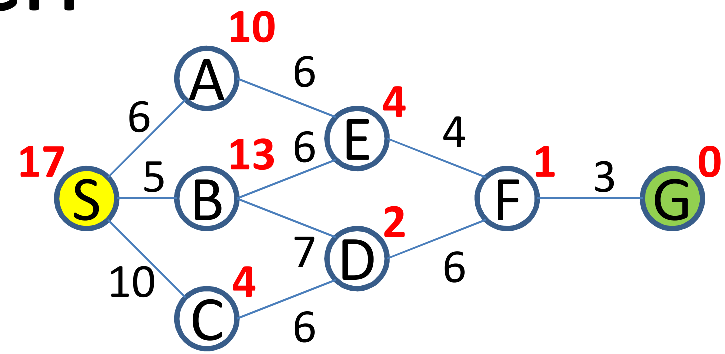
SBD

SBE

SAEFG

A* Search





QUEUE:

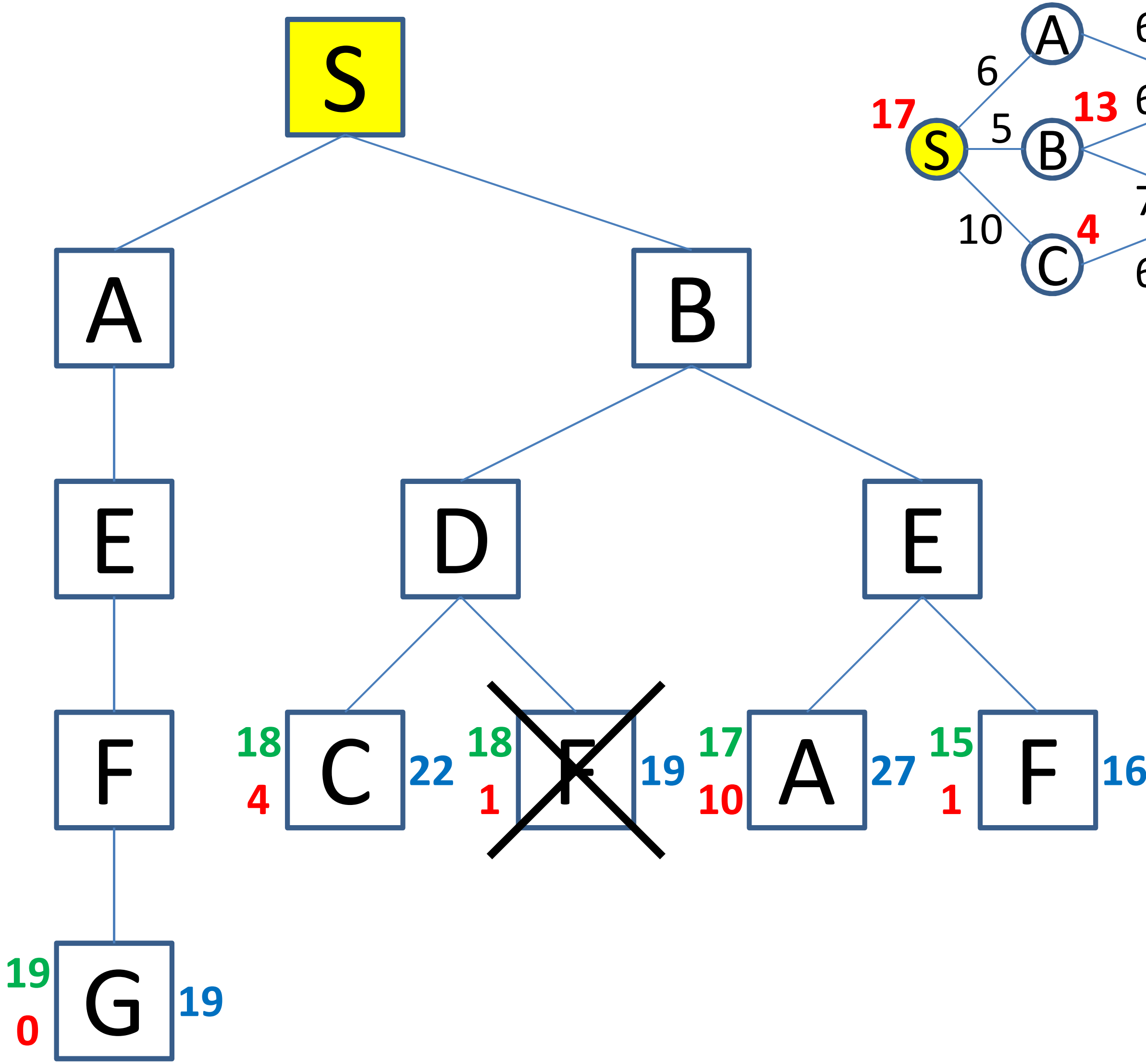
SBE

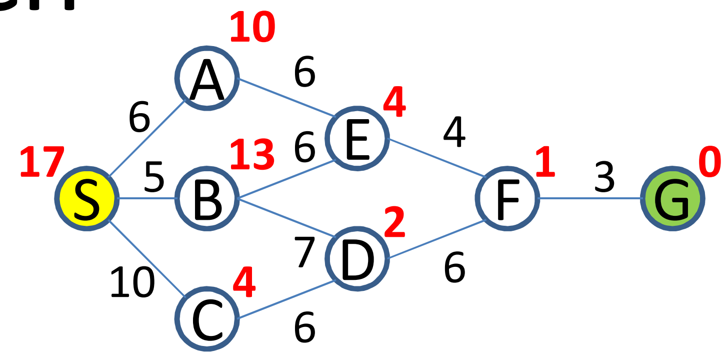
SBDF

SAEFG

SBDC

A* Search





QUEUE:

SBEF

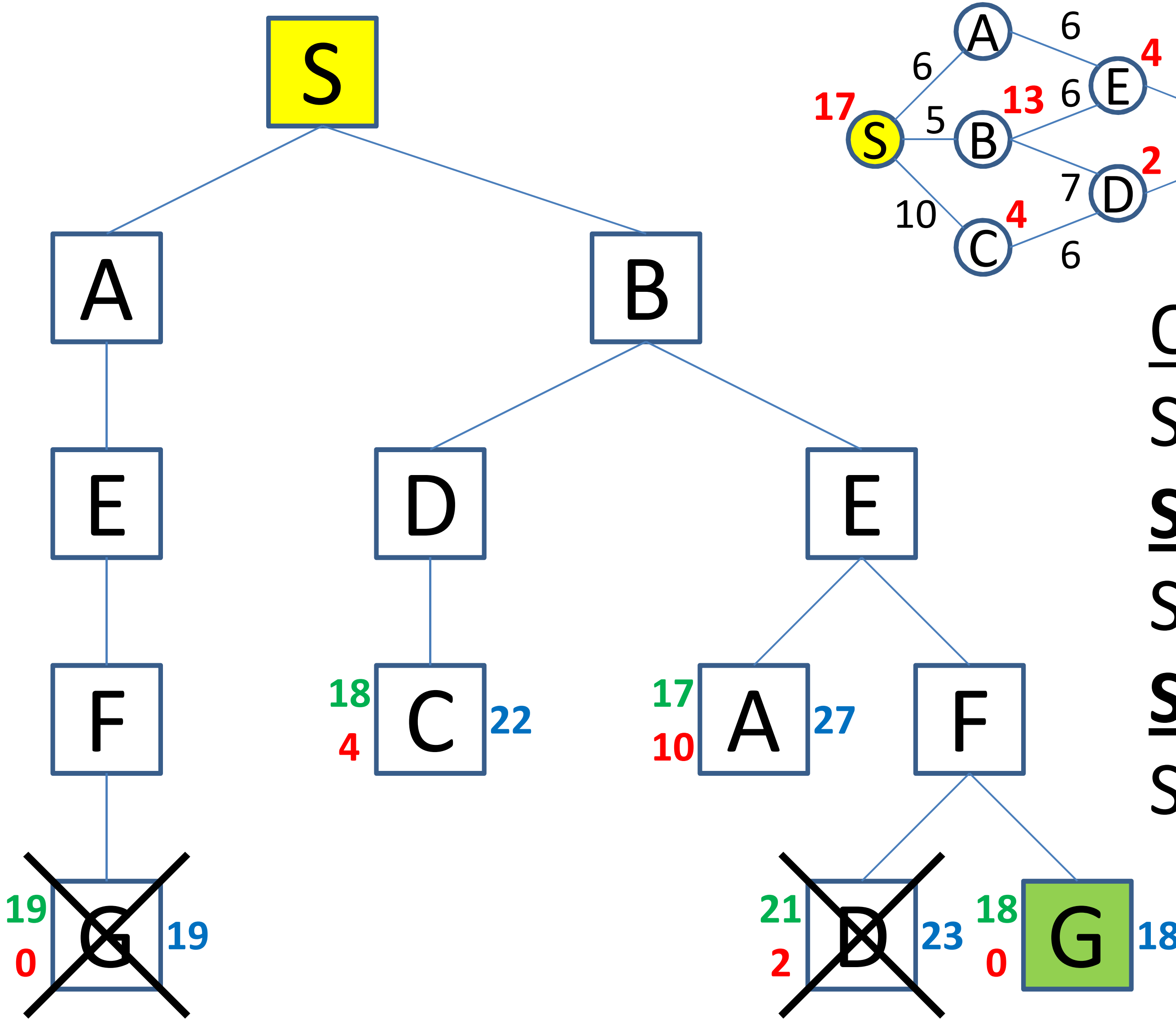
SAEFG

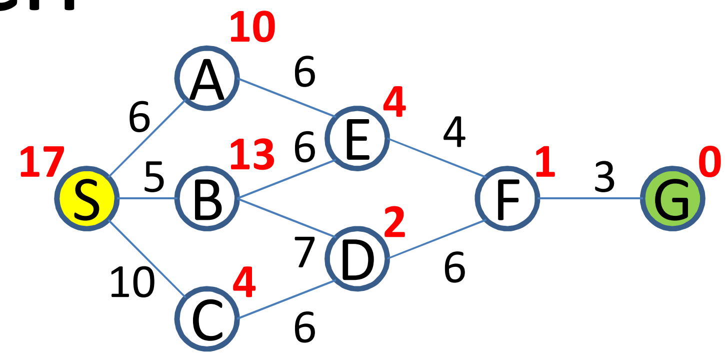
SBDF

SBDC

SBEA

A* Search





<u>QUEUE:</u>

SBEFG

<u>**SAEFG**</u>

SBDC

<u>**SBEFD**</u>

SBEA

Exercises: Artificial Intelligence

Iterated Deepening A*

Iterated Deepening A*

IDA* ALGORITHM

IDA* Algorithm

- ***f-bound*** $\longleftarrow$ ***f(S)***
- ***Algorithm:***
  - **WHILE** (goal is not reached) **DO**
    - ***f-bound*** $\longleftarrow$ ***f-limitted_search(f-bound)***
      - Perform **f-limited search** with ***f-bound***
        (See next slide)

f-limitted Search Algorithm

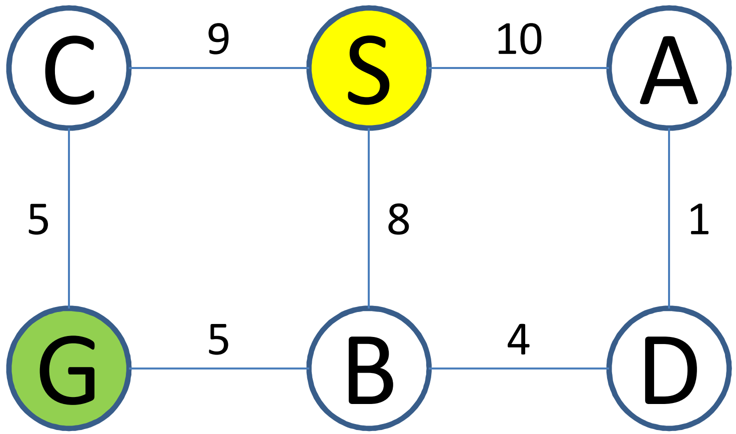
- ***Input:***
  - QUEUE ⟵ Path only containing root
  - f-bound ⟵ Natural number
  - f-new ⟵ ∞
- ***Algorithm:***
  - **WHILE** (QUEUE not empty && goal not reached) **DO**
    - Remove **first path** from QUEUE
    - Create paths to children
    - Reject paths with loops
    - Add paths with **f(path) ≤ f-bound** to **front** of QUEUE *(depth-first)*
    - f-new ⟵ minimum({f-new} ∪ {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.

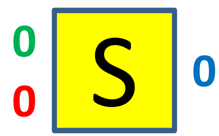


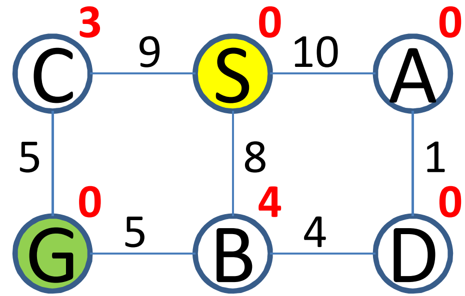
	S	A	B	C	D	G
heuristic	0	0	4	3	0	0

Iterated Deepening A*

IDA* SEARCH

IDA* Search

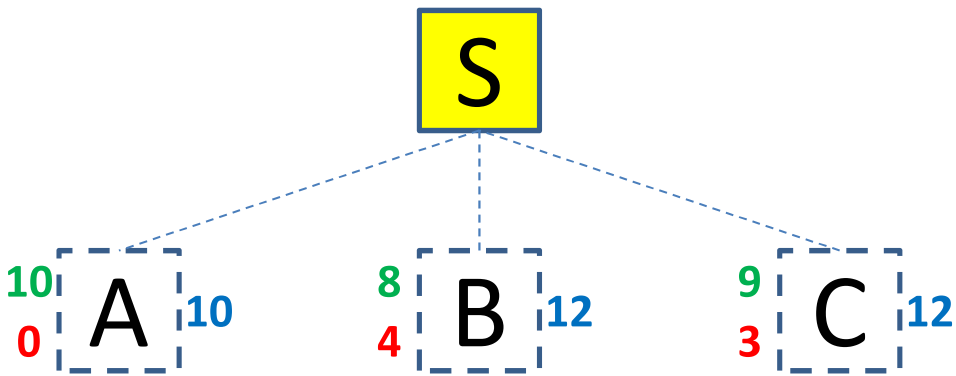


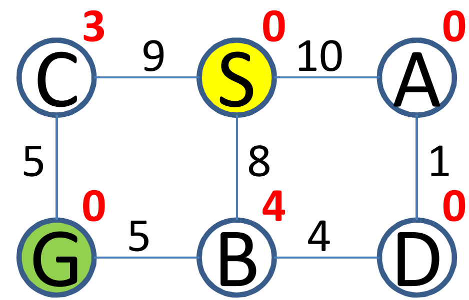


f-bound = 0

f-new = ∞

IDA* Search



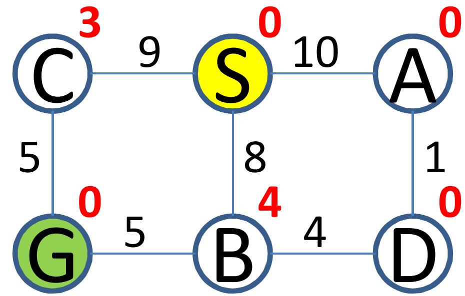


f-bound = 0

f-new = 10

Children are explored <u>depth-first</u>!

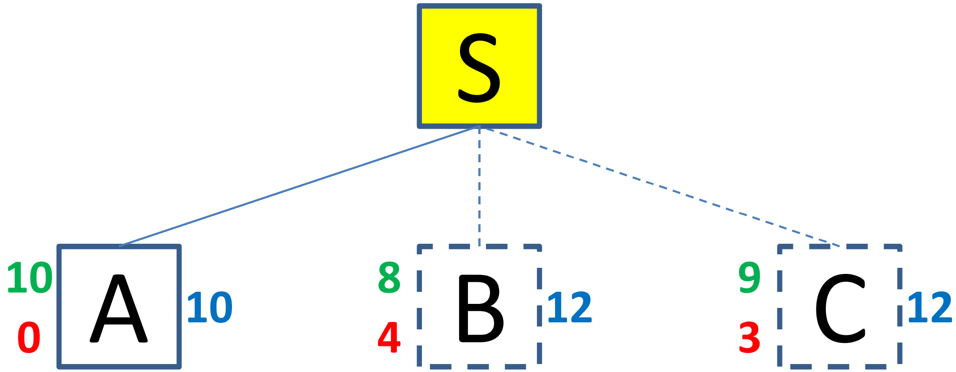
IDA* Search

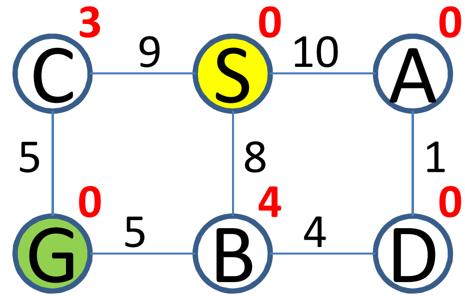


f-bound = 10

f-new = ∞

IDA* Search

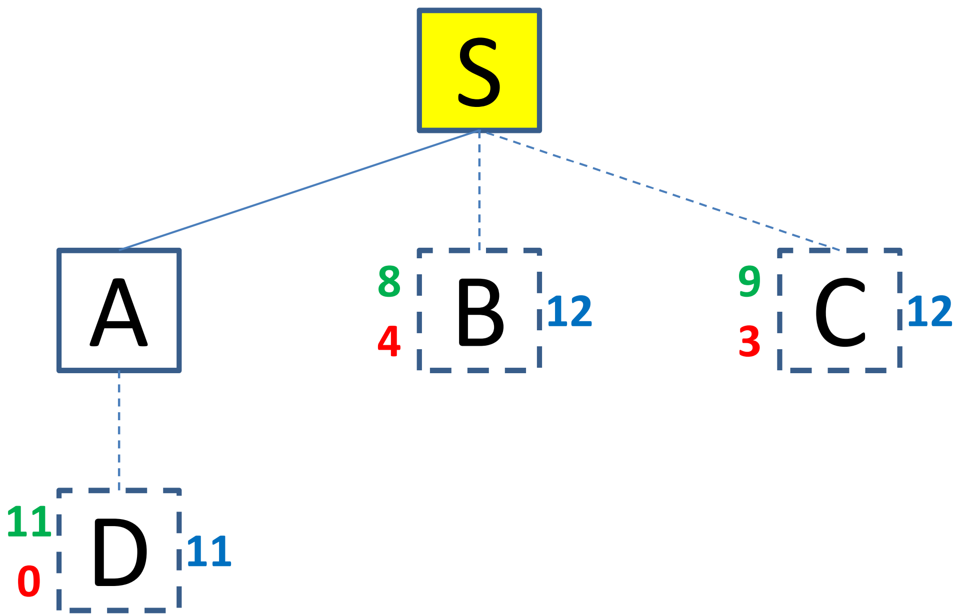


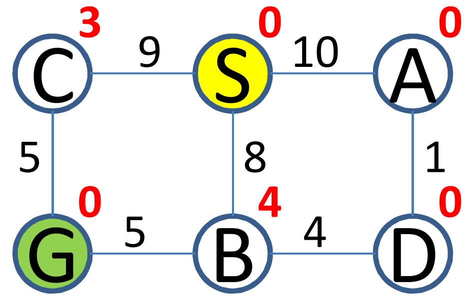


f-bound = 10

f-new = 12

IDA* Search

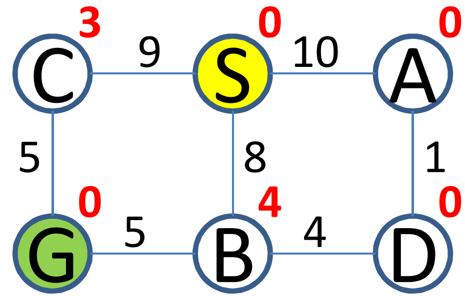




f-bound = 10

f-new = 11

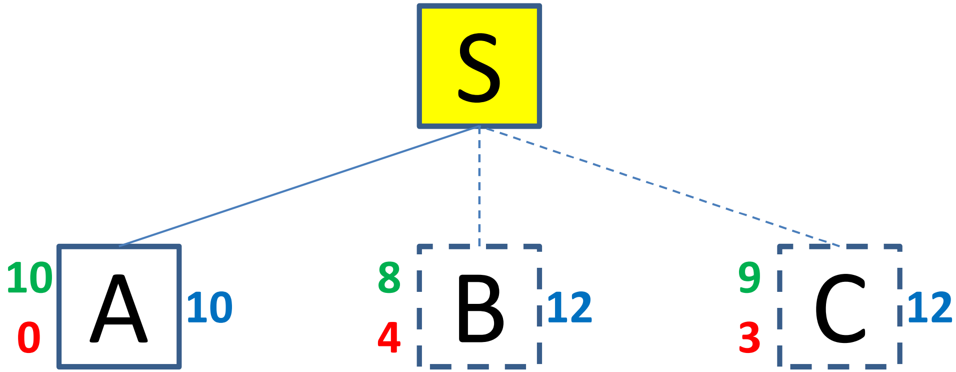
IDA* Search

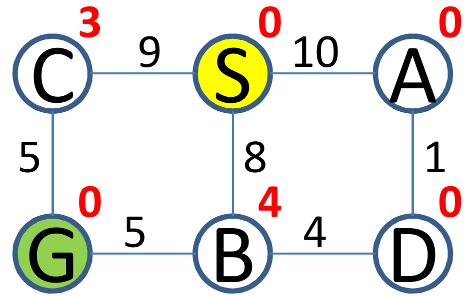


f-bound = 11

f-new = ∞

IDA* Search

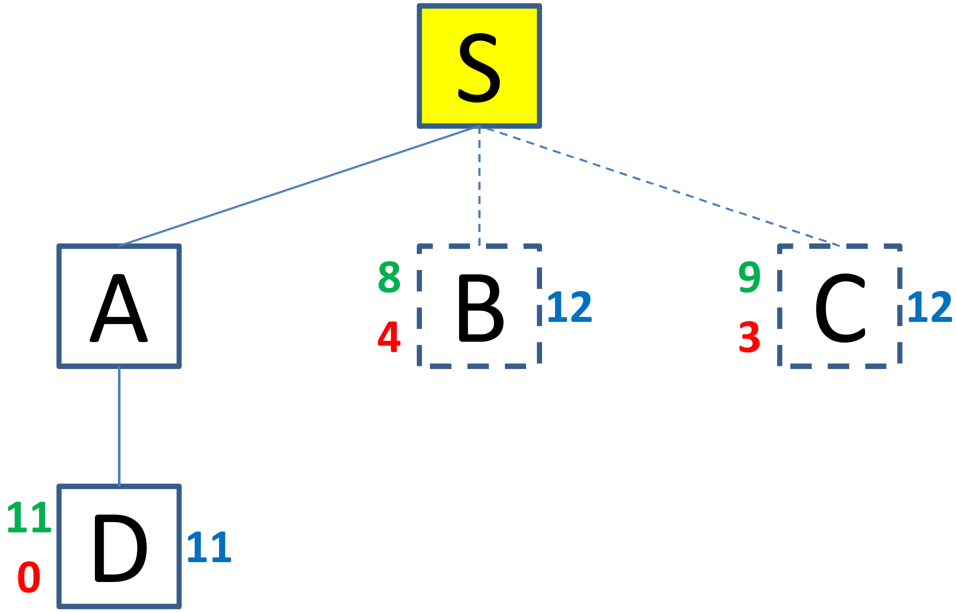


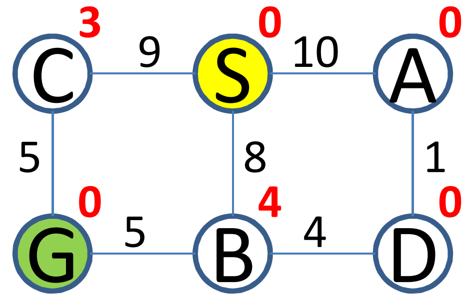


f-bound = 11

f-new = 12

IDA* Search

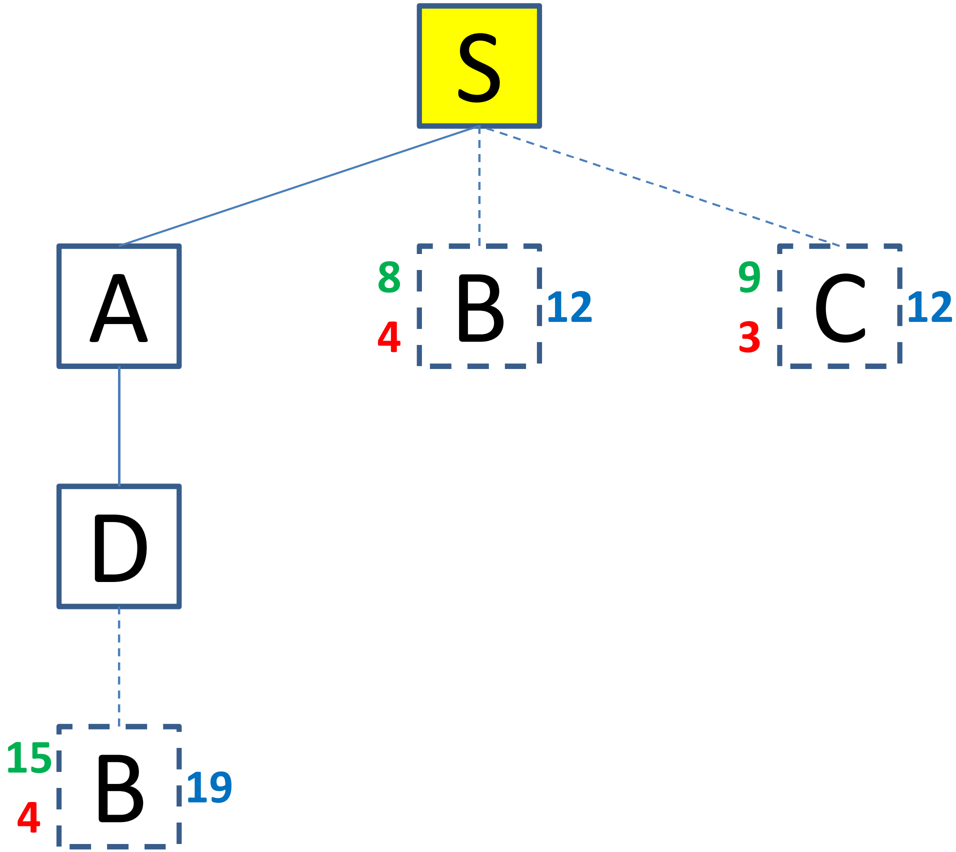


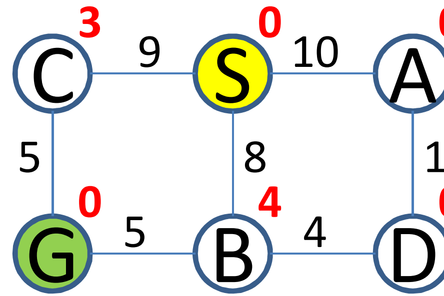


f-bound = 11

f-new = 12

IDA* Search

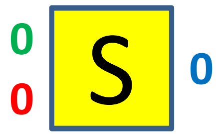


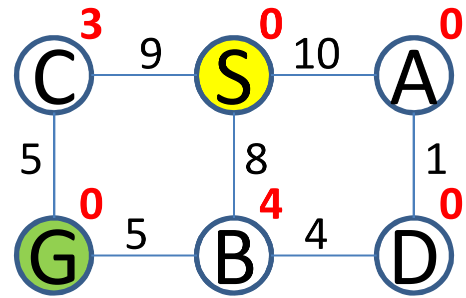


f-bound = 11

f-new = 12

IDA* Search

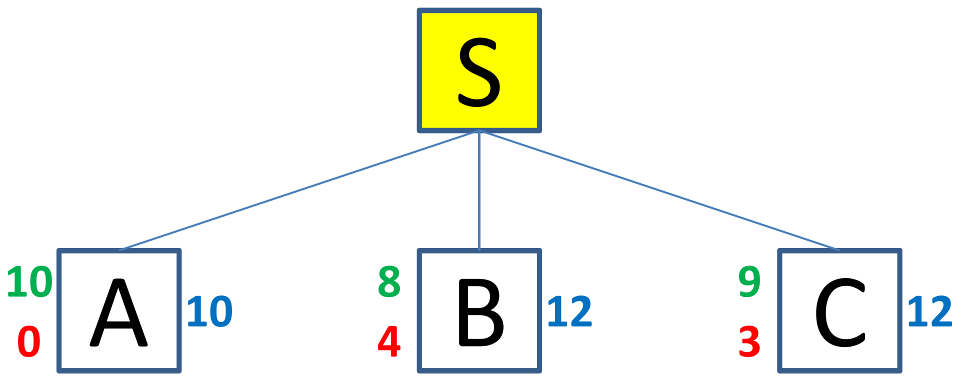


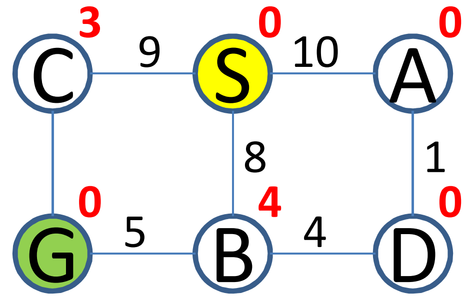


f-bound = 12

f-new = ∞

IDA* Search

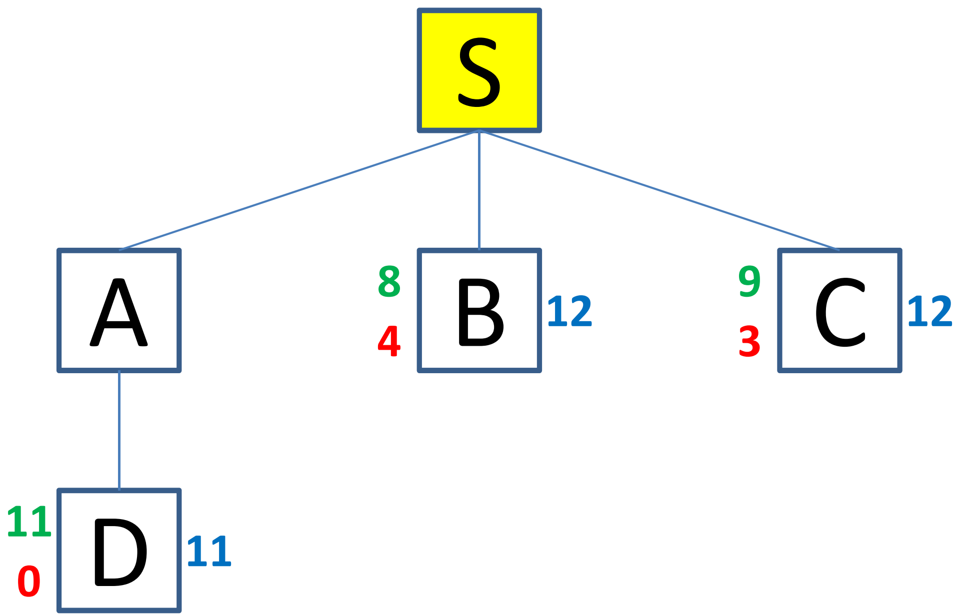


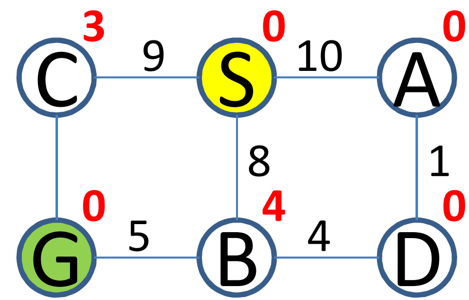


f-bound = 12

f-new = ∞

IDA* Search

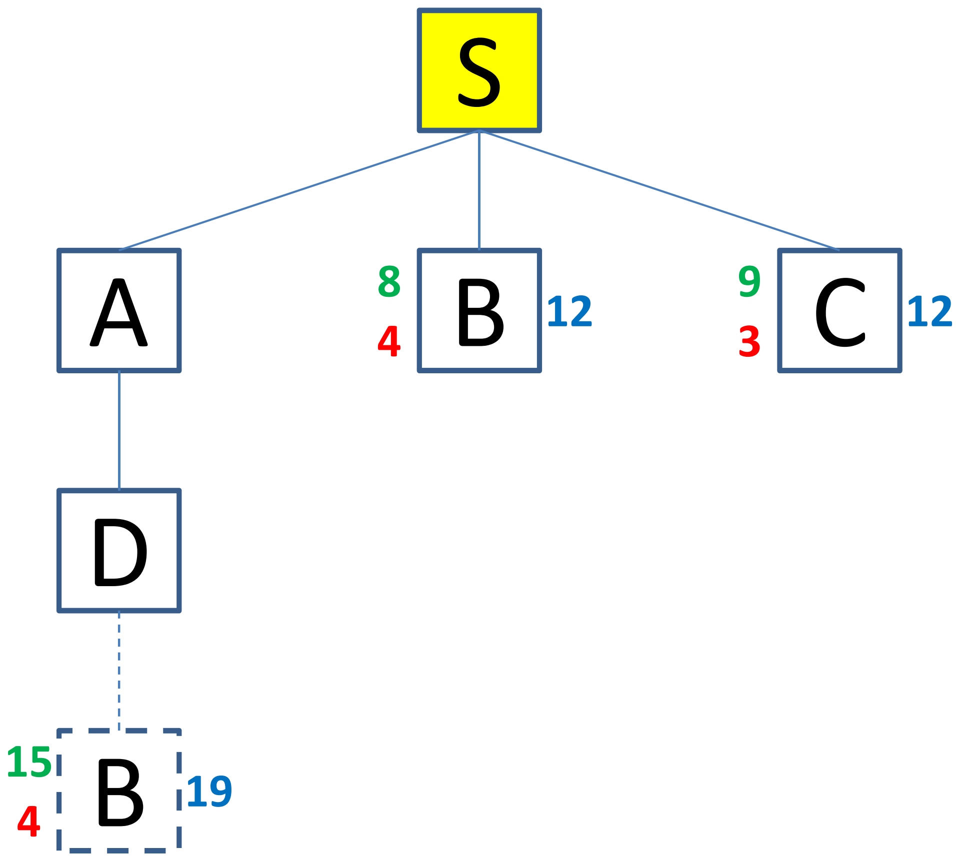


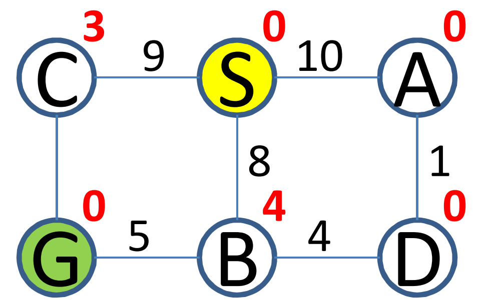


f-bound = 12

f-new = ∞

IDA* Search

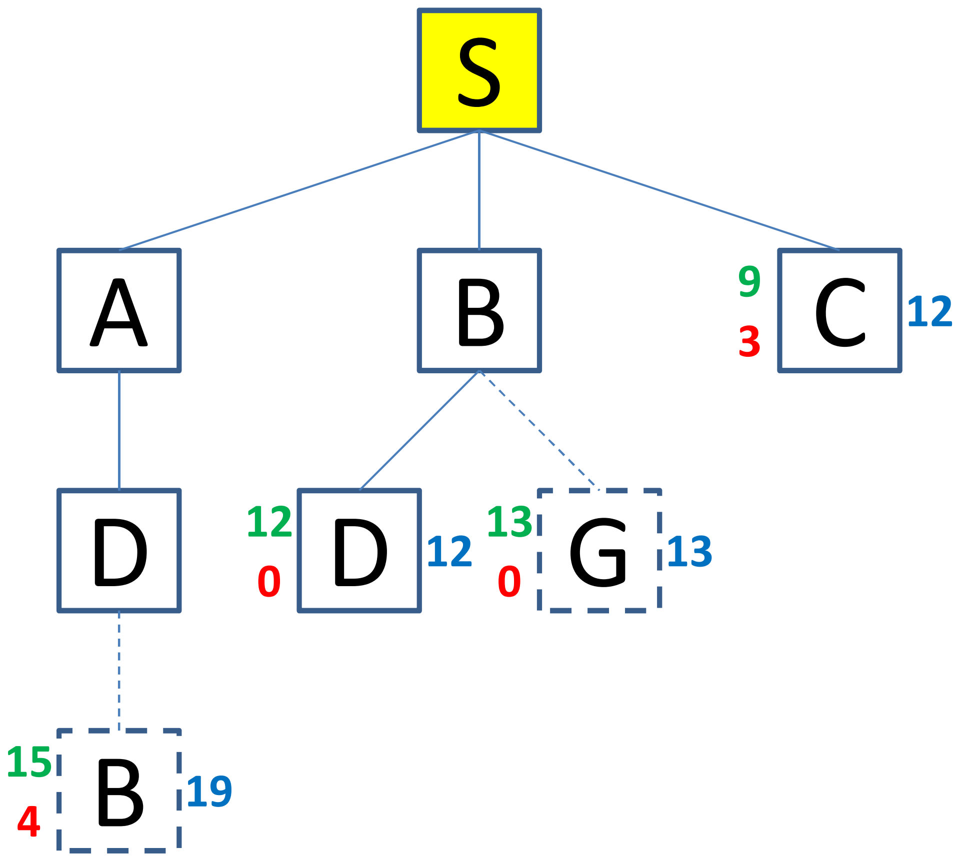


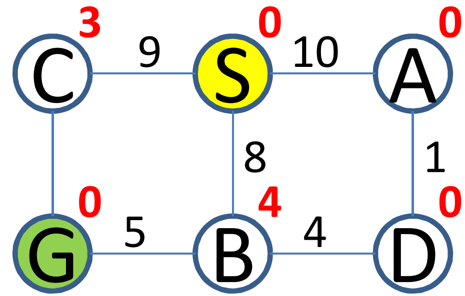


f-bound = 12

f-new = 19

IDA* Search

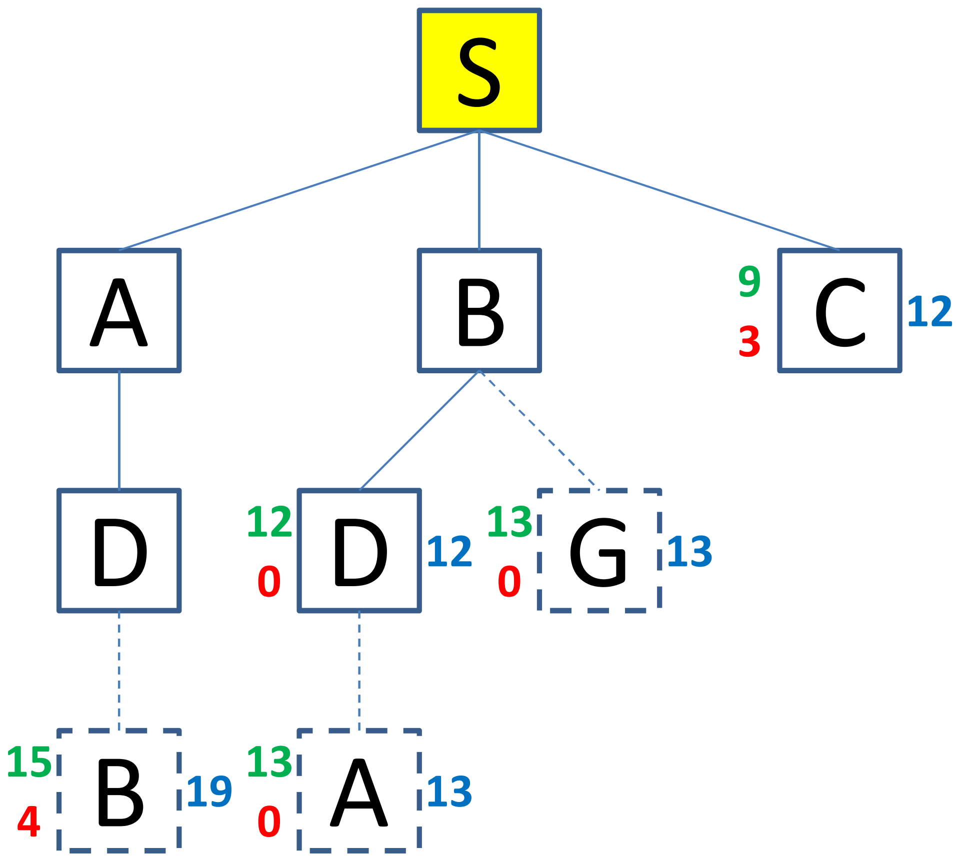


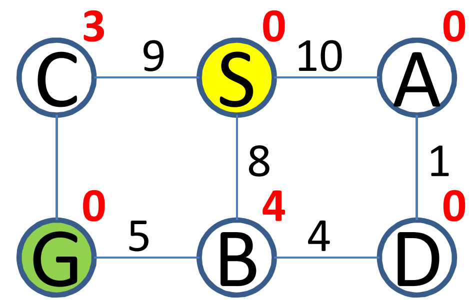


f-bound = 12

f-new = 13

IDA* Search

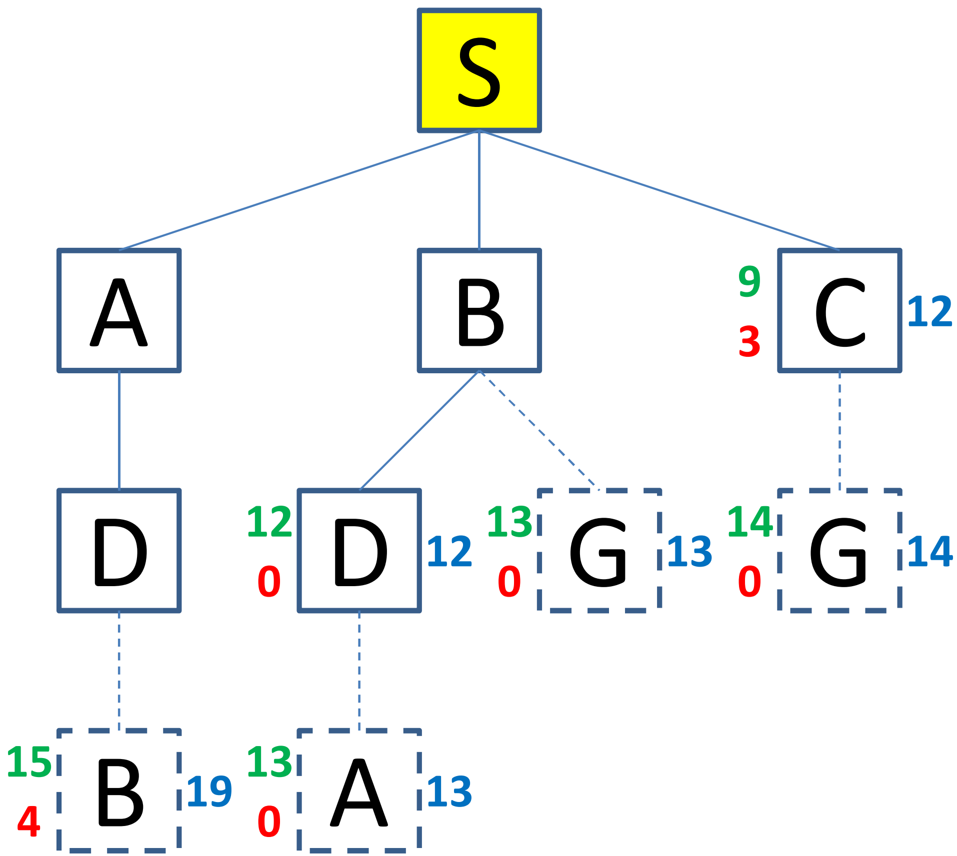


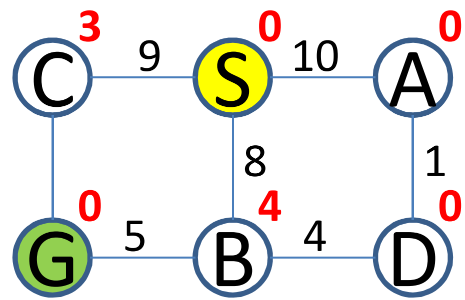


f-bound = 12

f-new = 13

IDA* Search

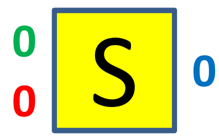


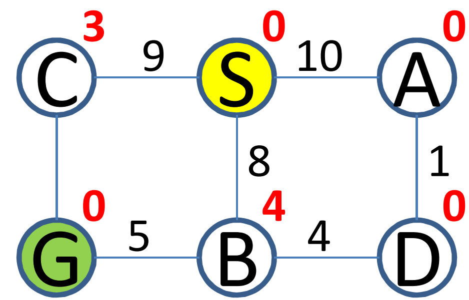


f-bound = 12

f-new = 13

IDA* Search

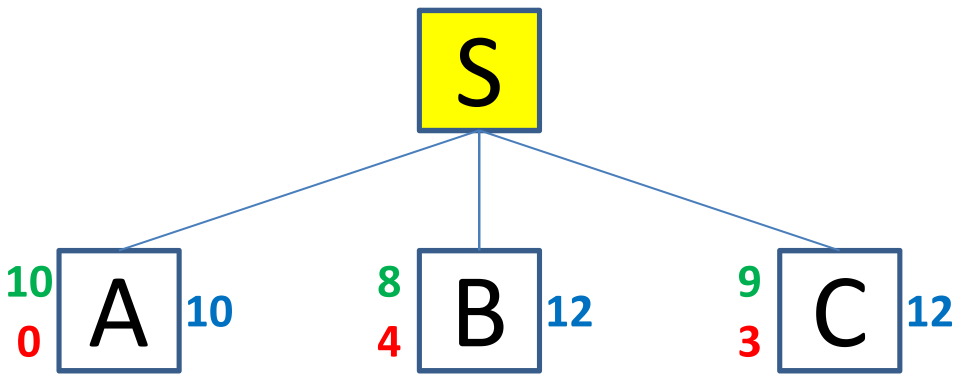


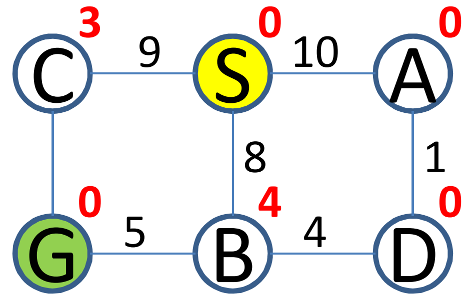


f-bound = 13

f-new = ∞

IDA* Search

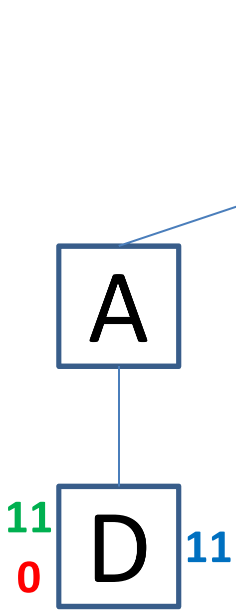


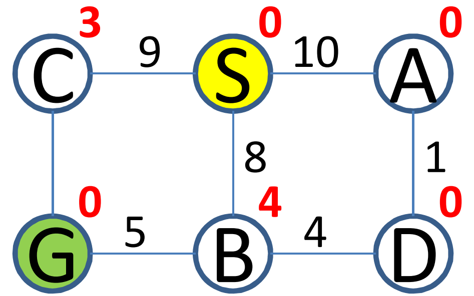


f-bound = 13

f-new = ∞

IDA* Search

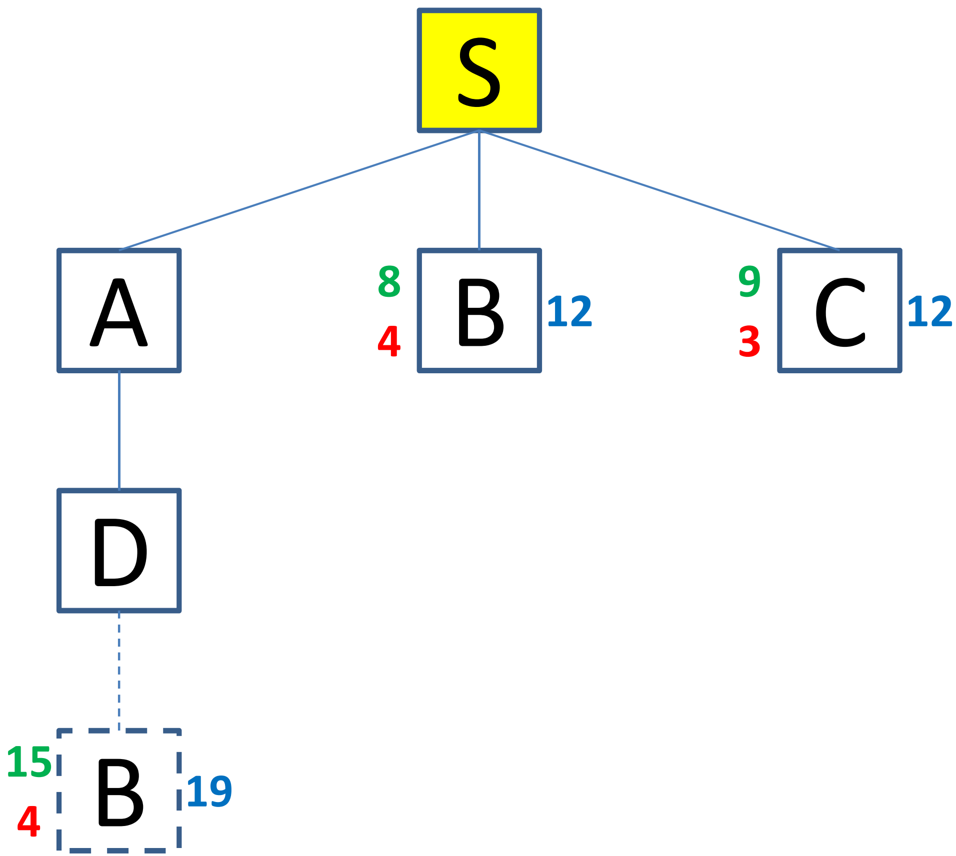


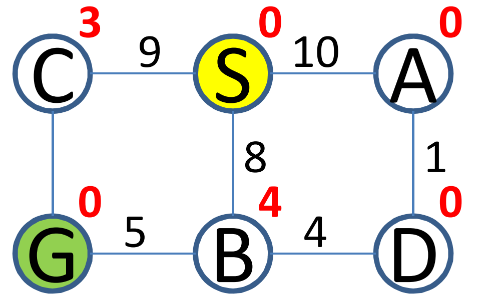


f-bound = 13

f-new = ∞

IDA* Search

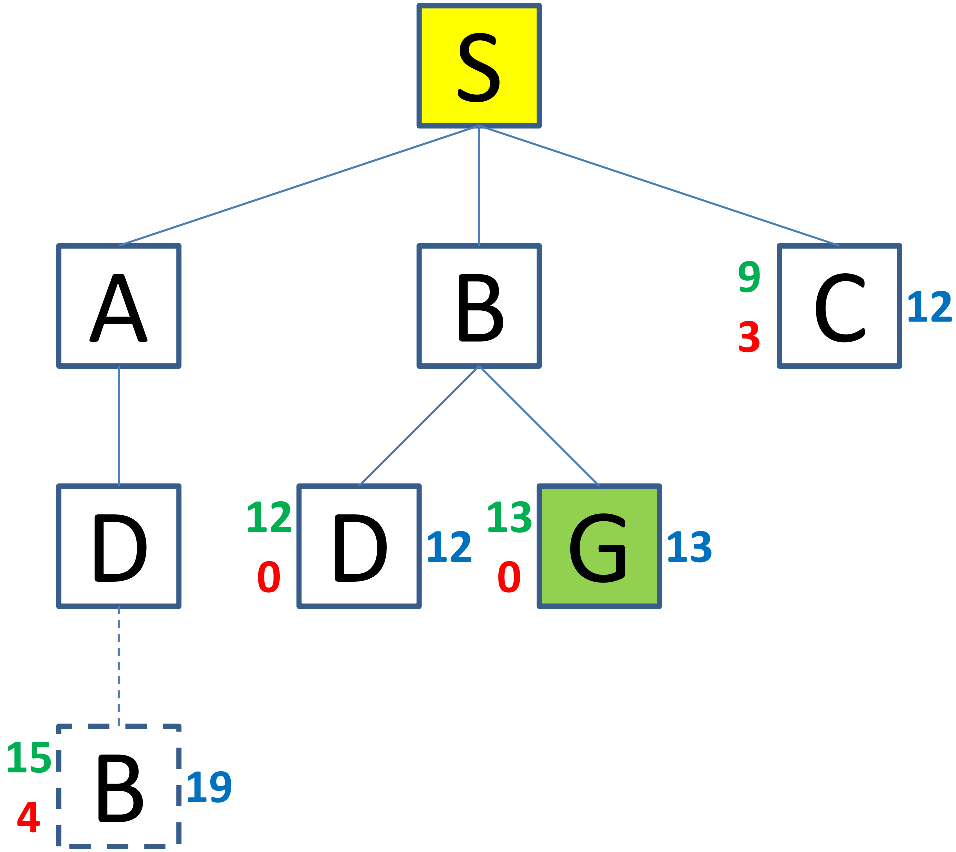


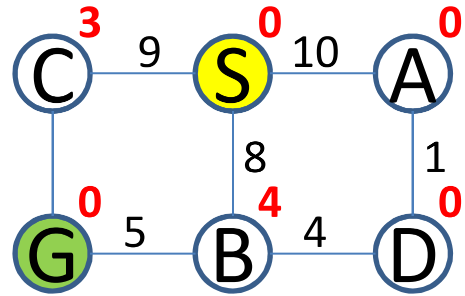


f-bound = 13

f-new = 19

IDA* Search





f-bound = 13

f-new = 19

Exercises: Artificial Intelligence

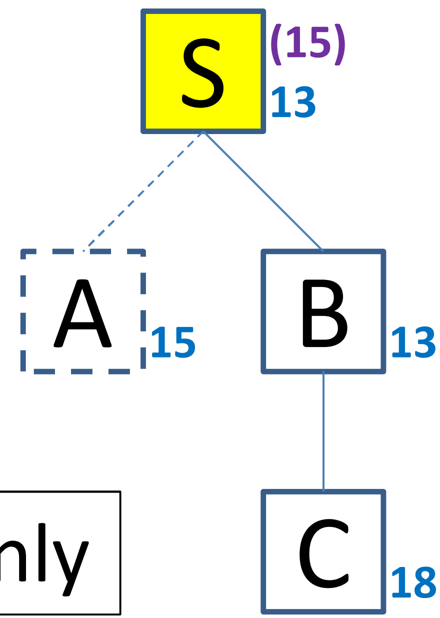
Simplified Memory-bounded A*

Simplified Memory-bounded A*

SMA* ALGORITHM

SMA* Algorithm

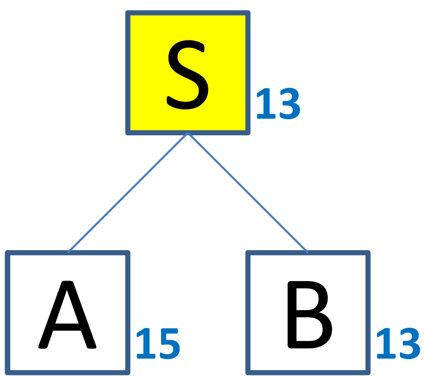
- Optimizes A* to work within reduced memory
- ***Key Idea:***
  - **IF** memory **full** for **extra node (C)**
  - **Remove highest f-value leaf (A)**
  - **Remember best-forgotten child** in <u>*each*</u> parent node **(15 in S)**



E.g. Memory of 3 nodes only

SMA* Algorithm

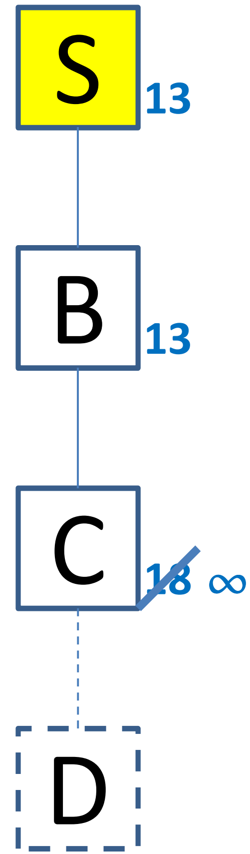
- **Generate Children 1 by 1**
  - **Expanding**: add 1 child at the time to QUEUE
  - Avoids **memory overflow**
  - **Allows monitoring** if nodes need deletion



First add A later B

SMA* Algorithm

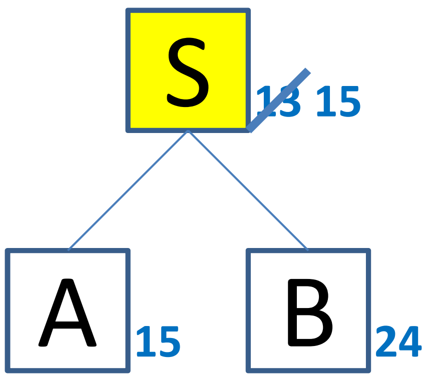
- **Too long paths: Give up**
  - **Extending** path **cannot fit** in **memory**
    - **give up (C)**
  - Set **f-value** node **(C)** to ∞
    - **Remembers:**
      path cannot be found here



E.g. Memory of 3 nodes only

SMA* Algorithm

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



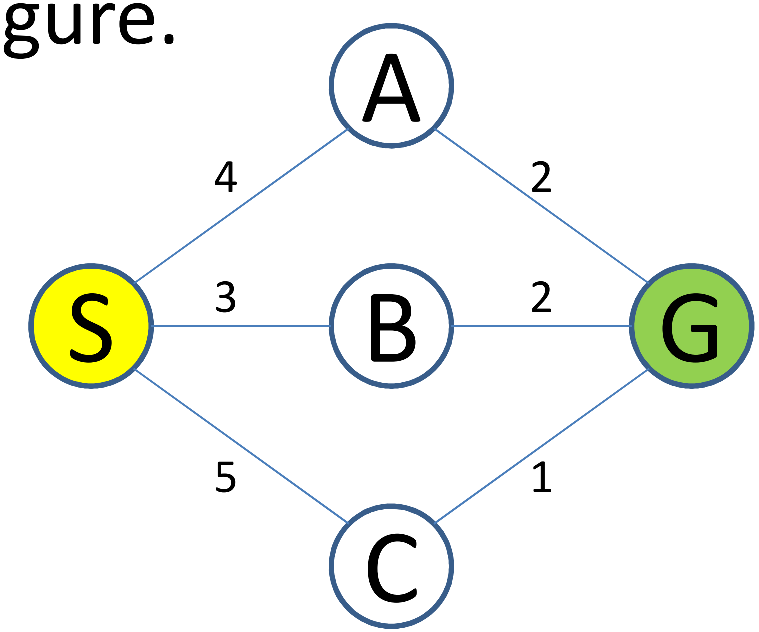
Better estimate for f(S)

Simplified Memory-bounded A*

SMA* BY EXAMPLE

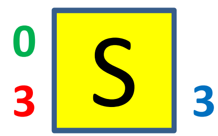
SMA* by Example

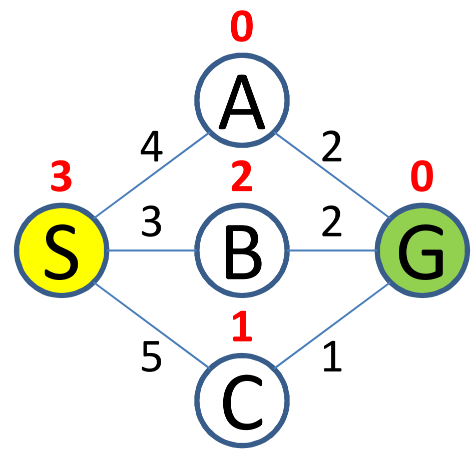
- Perform SMA* (memory: 3 nodes) on the following figure.



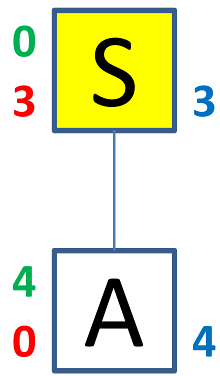
	S	A	B	C	G
heuristic	3	0	2	1	0

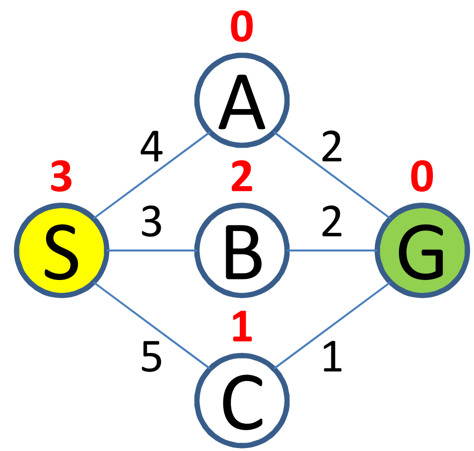
SMA* by Example





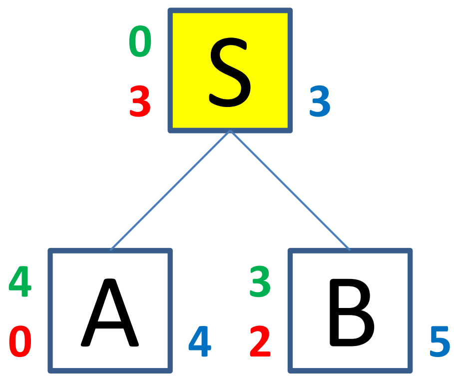
SMA* by Example

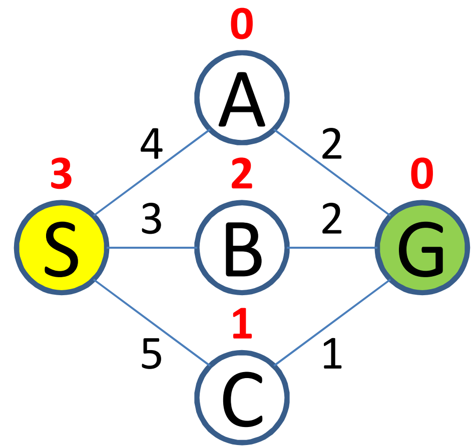




Generate children
(One by one)

SMA* by Example

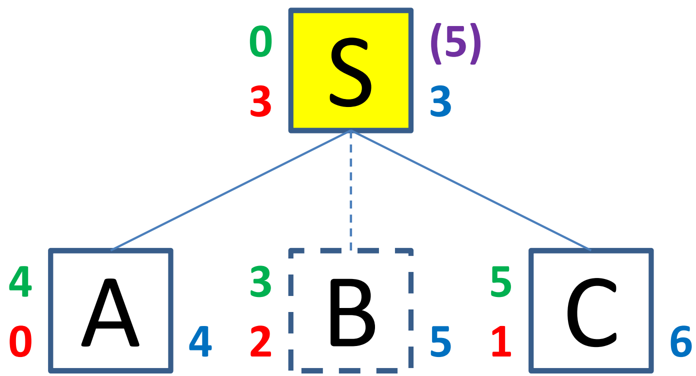


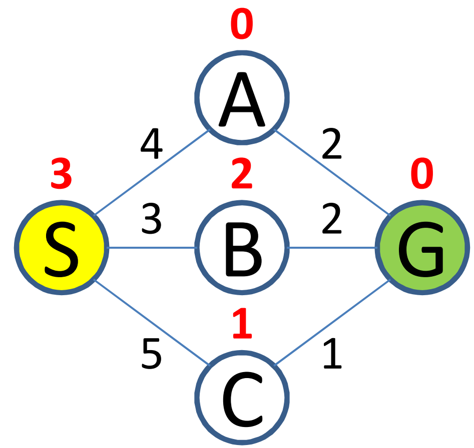


Generate children

(One by one)

SMA* by Example



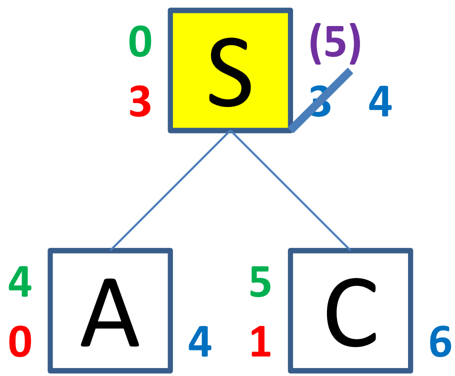


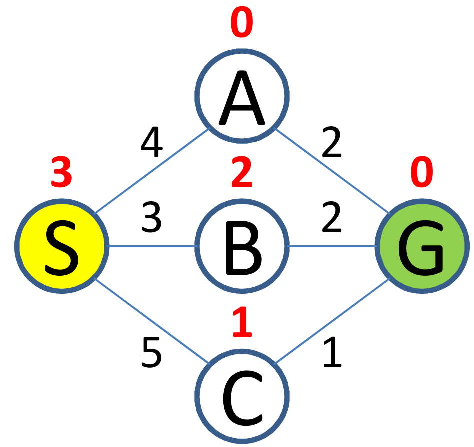
Generate children

(One by one)

Memory full

SMA* by Example

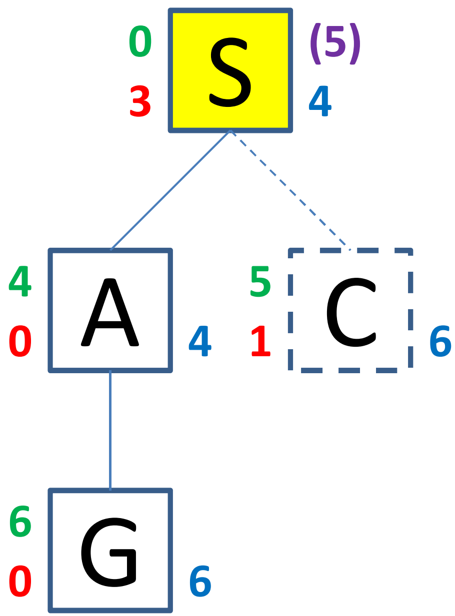


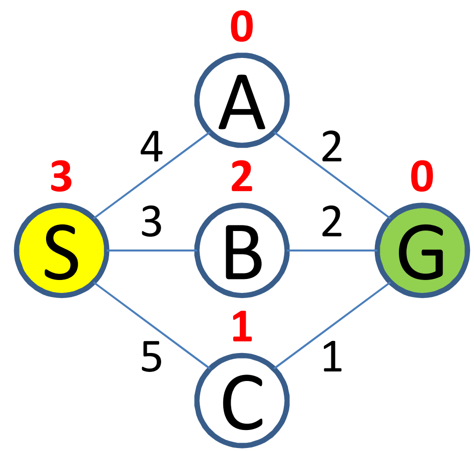


All children are explored

Adjust f-values

SMA* by Example



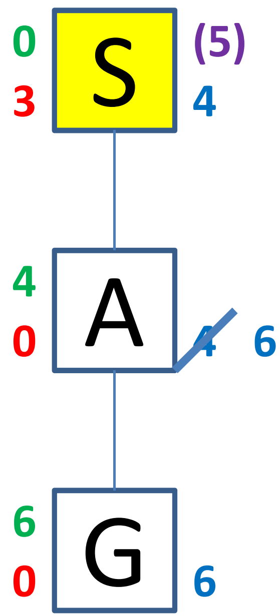


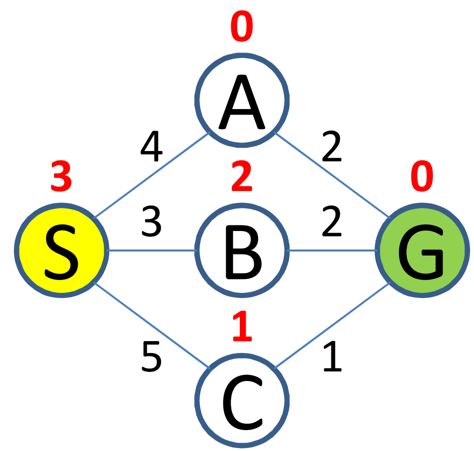
Generate children

(One by one)

Memory full

SMA* by Example

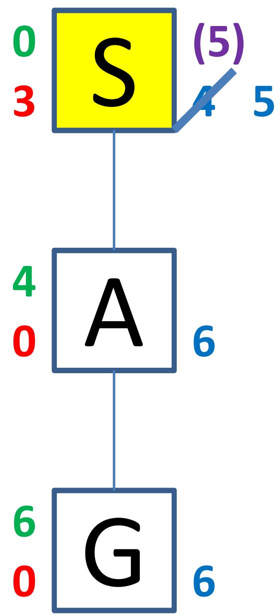


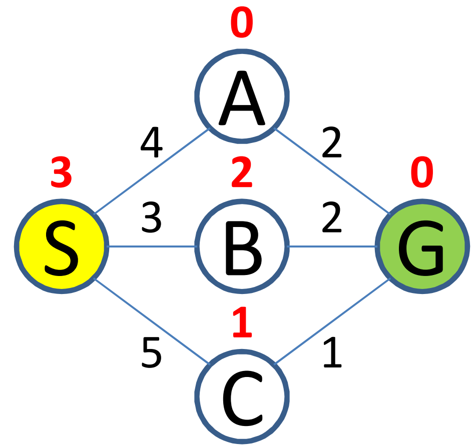


All children are explored

Adjust f-values

SMA* by Example

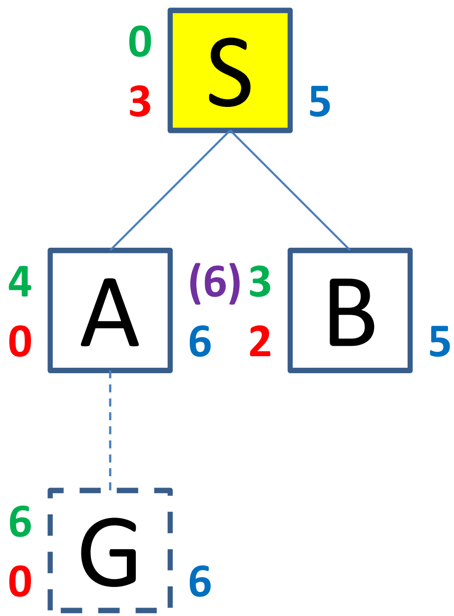


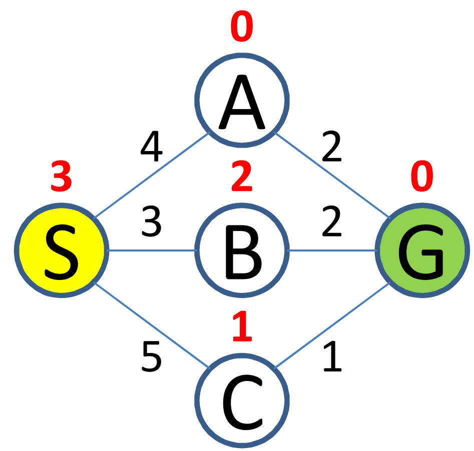


All children are explored (update)

Adjust f-values

SMA* by Example

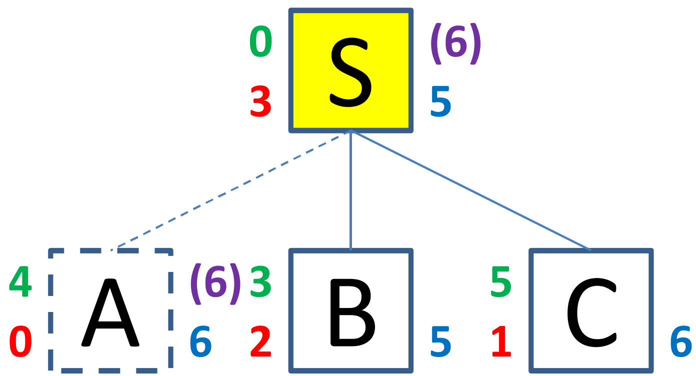


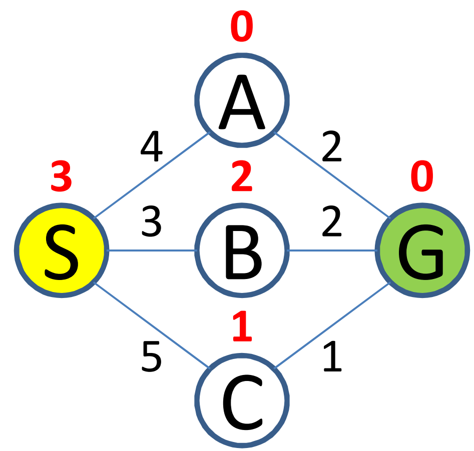


Generate children
(One by one)

Memory full

SMA* by Example

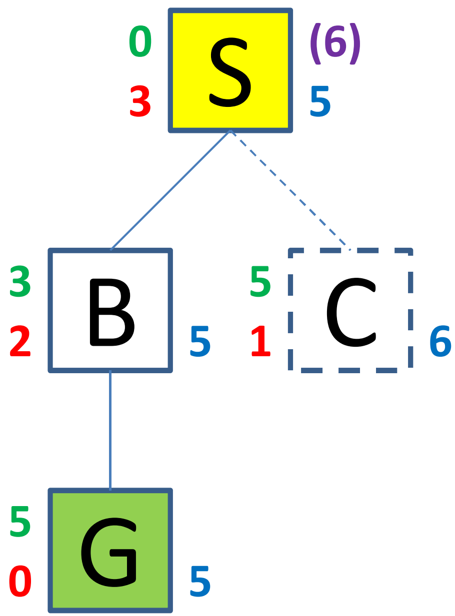


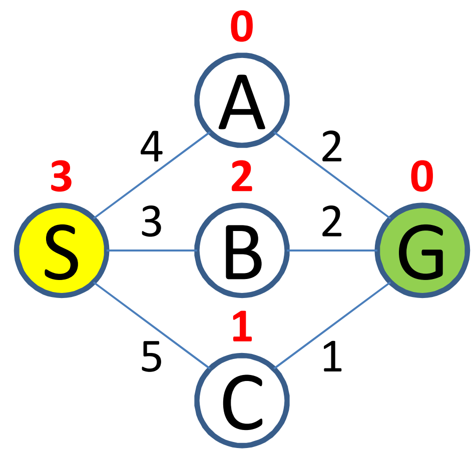


Generate children
(One by one)

Memory full

SMA* by Example





Generate children
(One by one)

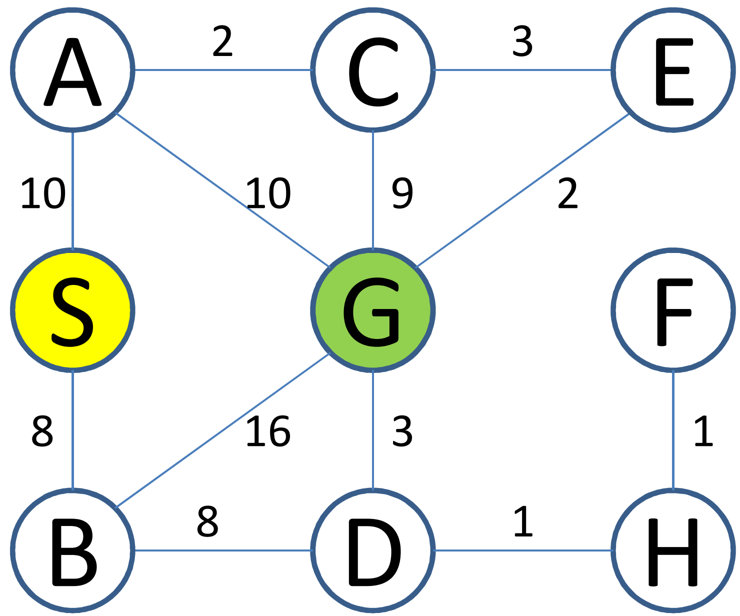
Memory full

Simplified Memory-bounded A*

PROBLEM

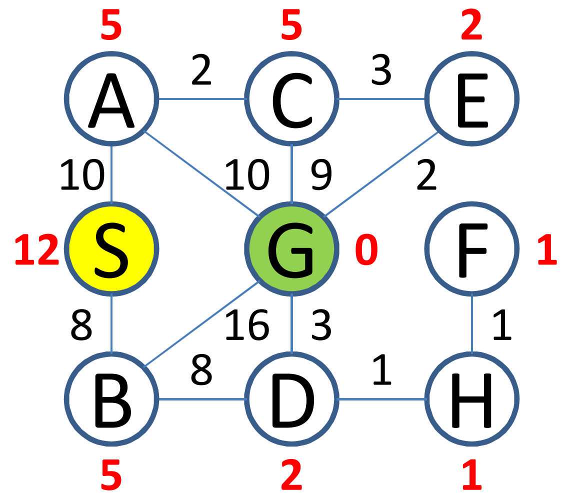
Problem

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

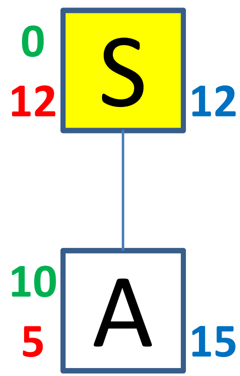


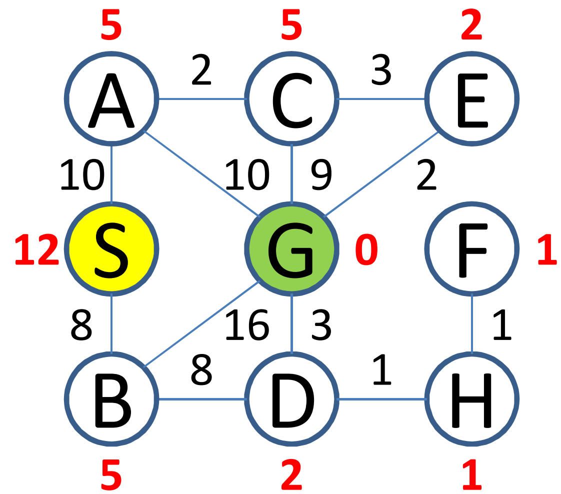
	S	A	B	C	D	E	F	H	G
heuristic	12	5	5	5	2	2	1	1	0

Problem

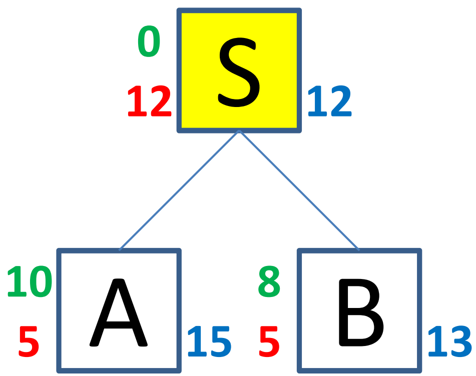


Problem

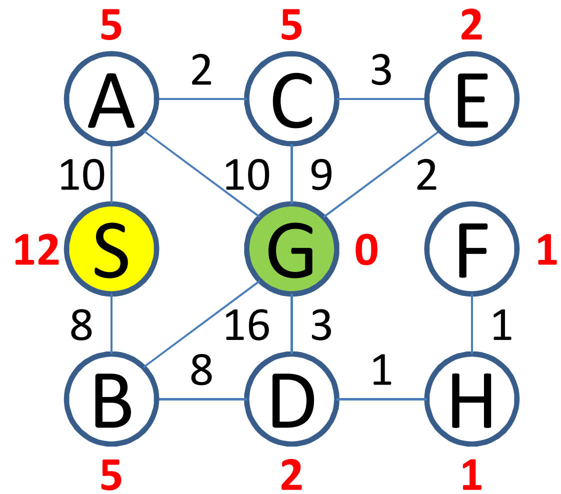




Problem

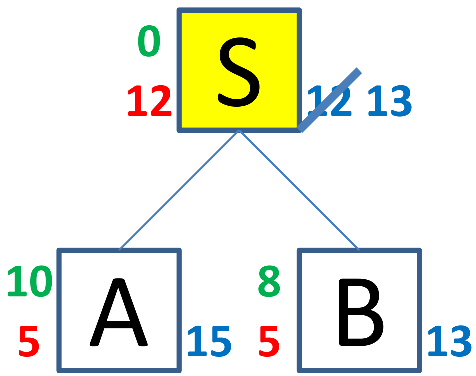


0
12
S
12
10
5
A
15
8
5
B
13

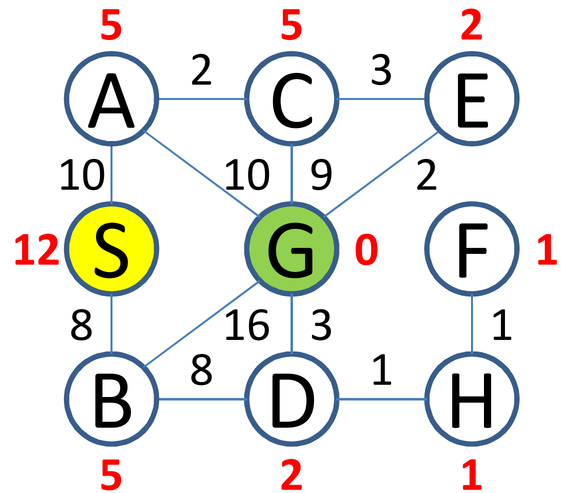


5
5
2
A
2
C
3
E
10
10
9
2
12
S
G
0
F
1
8
16
3
1
B
8
D
1
H
5
2
1

Problem

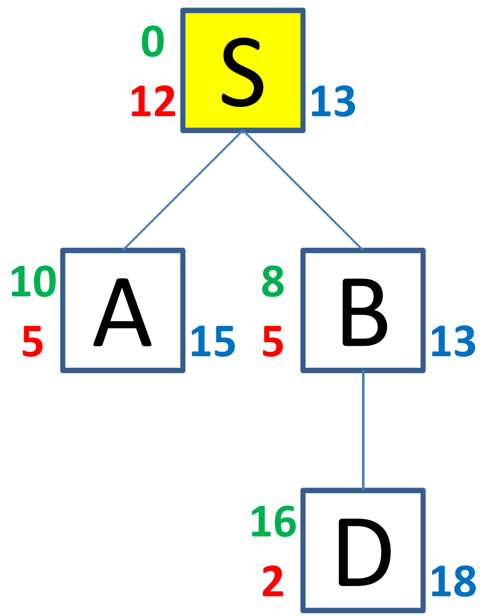


0
12
S
~~12~~ 13
10
5
A
15
8
5
B
13

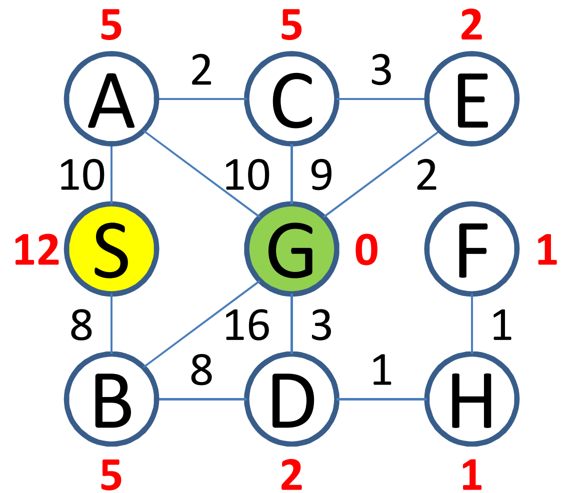


5 5 2
A C E
2 3
10 10 9 2
12 S G 0 F 1
8 16 3 1
8 1
B D H
5 2 1

Problem

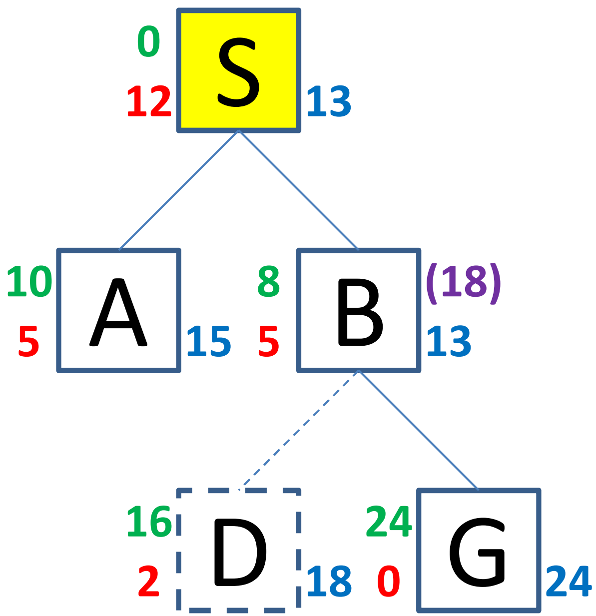


0
12
S
13
10
5
A
15
8
5
B
13
16
2
D
18

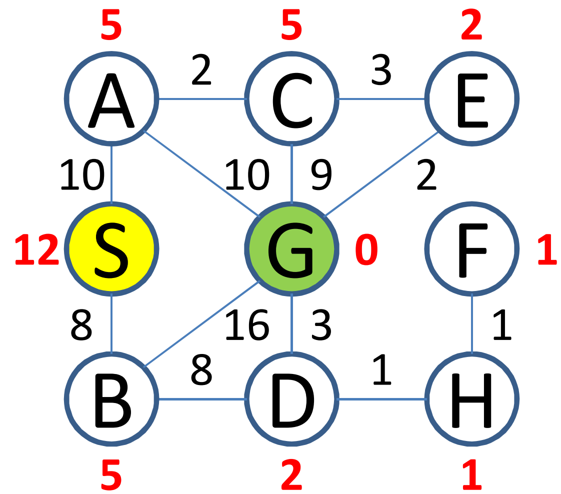


5
5
2
A
2
C
3
E
10
10
9
2
12
S
G
0
F
1
8
16
3
1
8
1
B
D
H
5
2
1

Problem

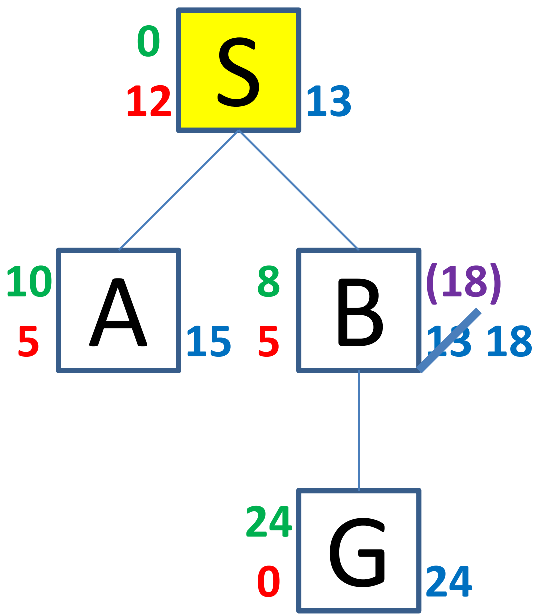


0
12
S
13
10
5
A
15
8
5
B
(18)
13
16
2
D
18
24
0
G
24

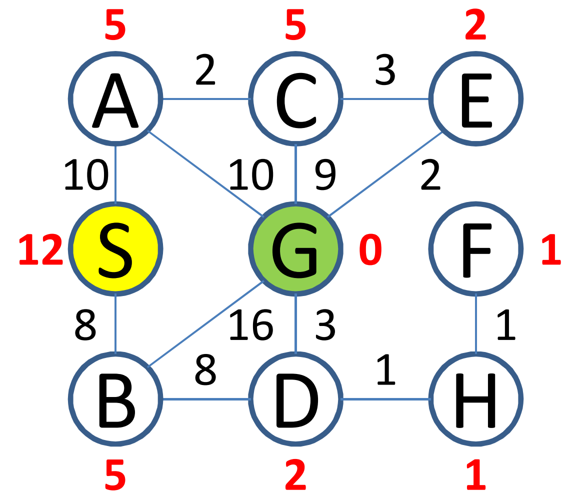


5
5
2
A
2
C
3
E
10
10
9
2
12
S
G
0
F
1
8
16
3
1
8
1
B
D
H
5
2
1

Problem

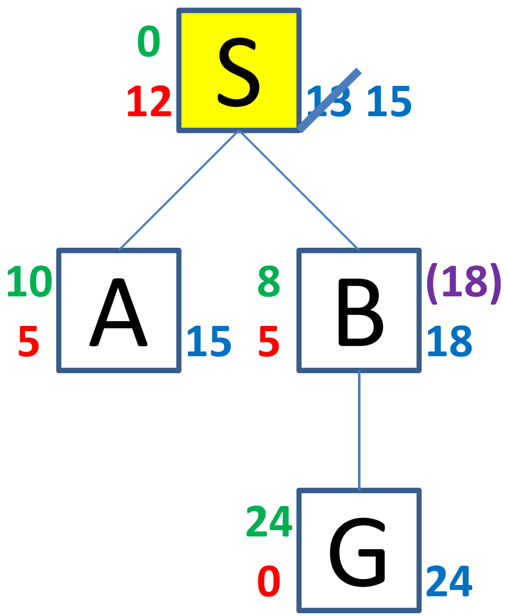


S
0
12
13
A
10
5
15
B
8
5
(18)
~~13~~ 18
G
24
0
24

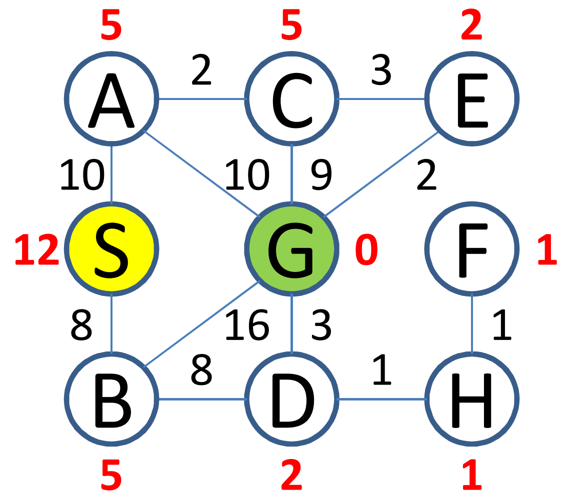


5 5 2
A 2 C 3 E
10 10 9 2
12 S G 0 F 1
8 16 3 1
B 8 D 1 H
5 2 1

Problem

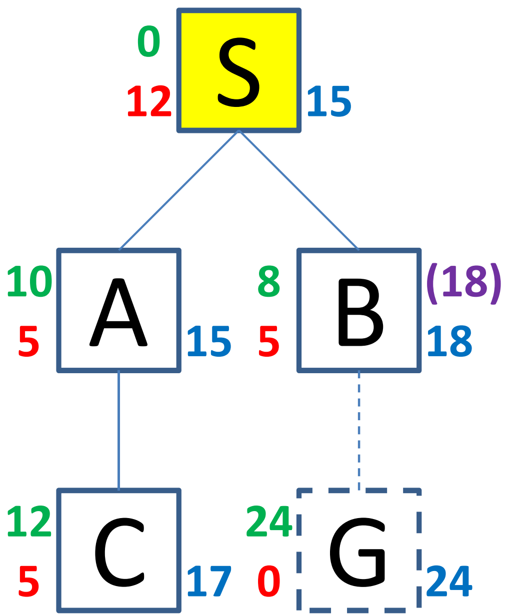


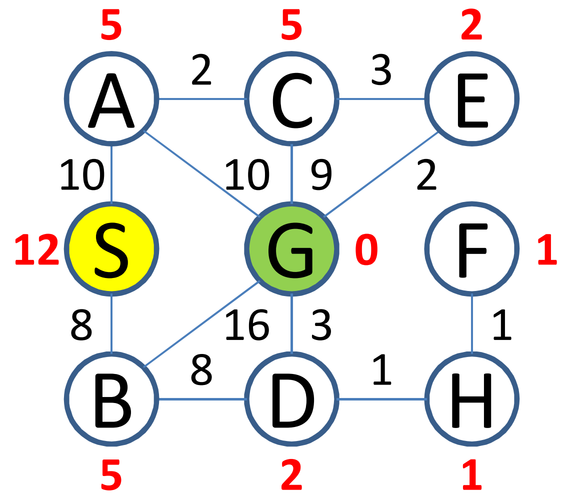
S
0
12
~~13~~ 15
A
10
5
15
B
8
5
(18)
18
G
24
0
24



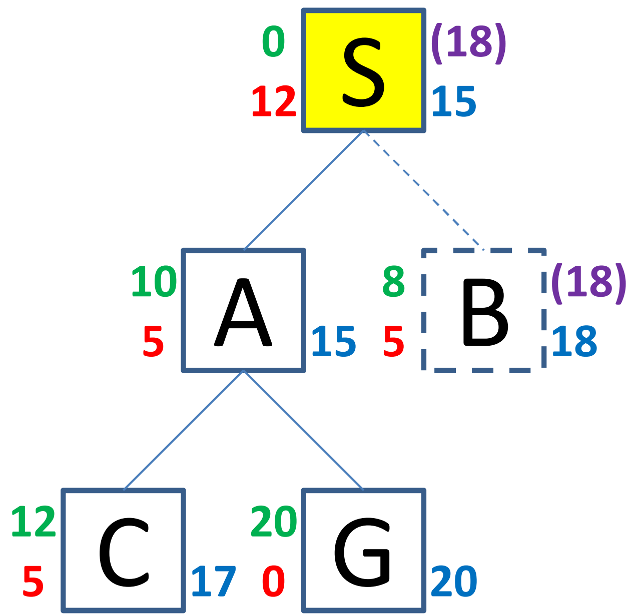
5
5
2
A
2
C
3
E
10
10
9
2
12
S
G
0
F
1
8
16
3
1
B
8
D
1
H
5
2
1

Problem

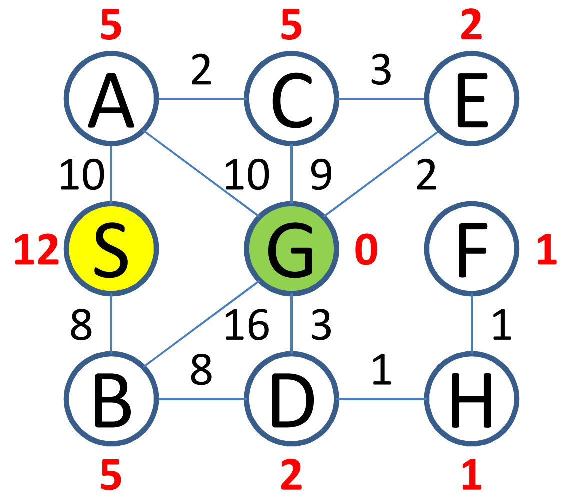

0
12
S
15
10
5
A
15
8
5
B
(18)
18
12
5
C
17
24
0
G
24


5
5
2
A
2
C
3
E
10
10
9
2
12
S
G
0
F
1
8
16
3
1
8
1
B
D
H
5
2
1

Problem

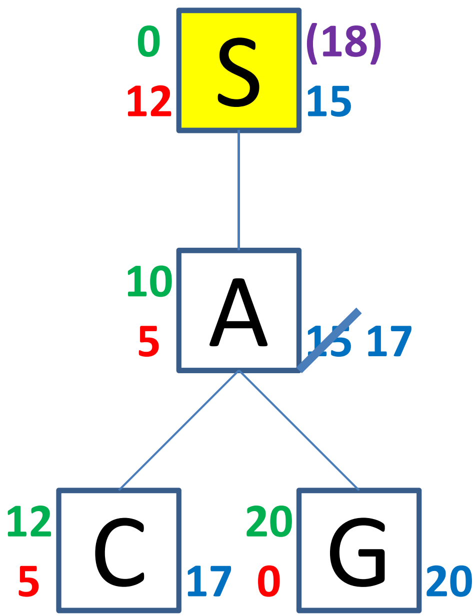


0
S
(18)
12
15
10
A
5
15
8
B
(18)
5
18
12
C
5
17
20
G
0
20

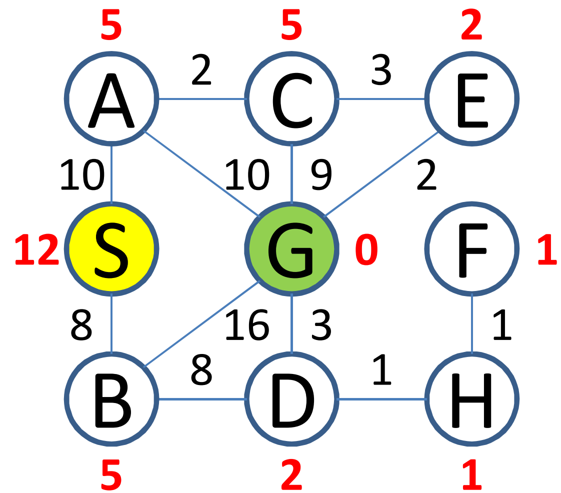


5
5
2
A
2
C
3
E
10
10
9
2
12
S
G
0
F
1
8
16
3
1
B
8
D
1
H
5
2
1

Problem

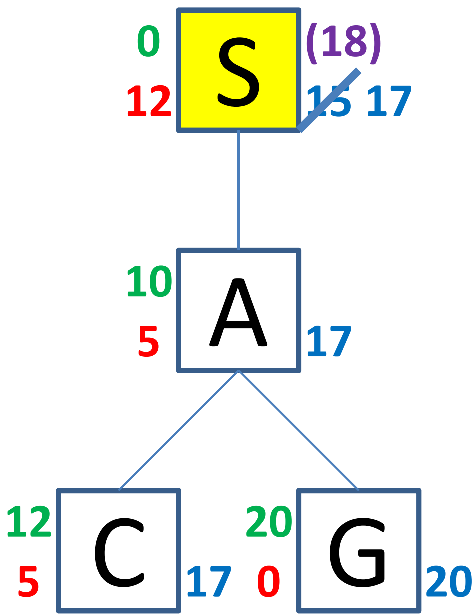


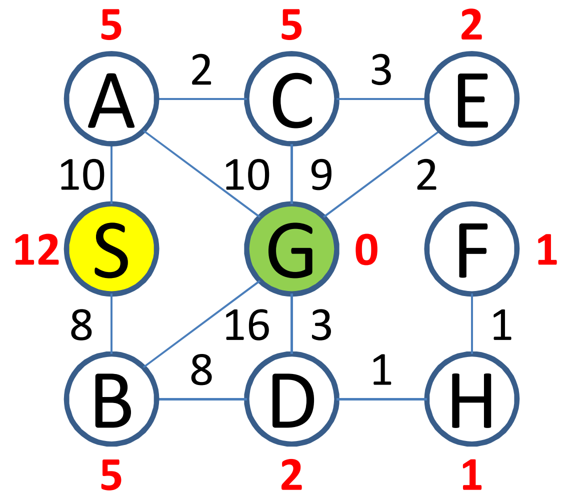
0
S
(18)
12
15
10
A
5
~~15~~ 17
12
C
5
17
20
G
0
20



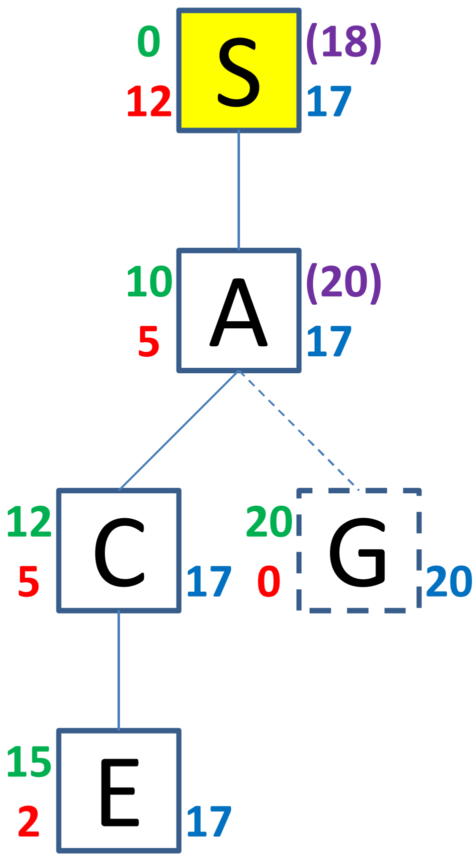
5
5
2
A
2
C
3
E
10
10
9
2
12
S
G
0
F
1
8
16
3
1
8
1
B
D
H
5
2
1

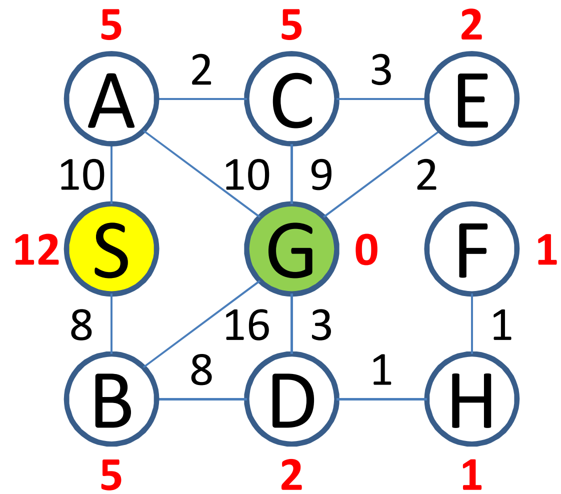
Problem


0
12
S
(18)
~~15~~ 17
10
5
A
17
12
5
C
17
20
0
G
20

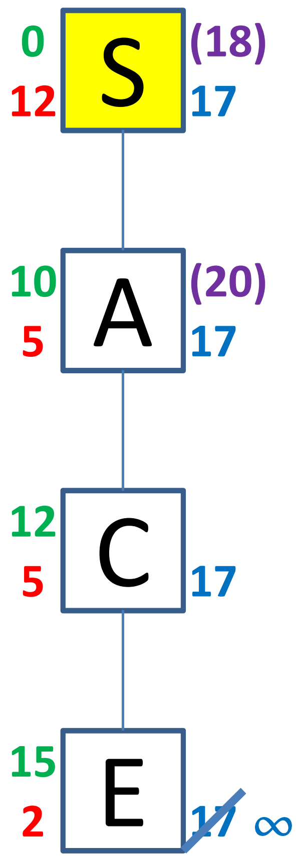

5
5
2
A
2
C
3
E
10
10
9
2
12
S
G
0
F
1
8
16
3
1
B
8
D
1
H
5
2
1

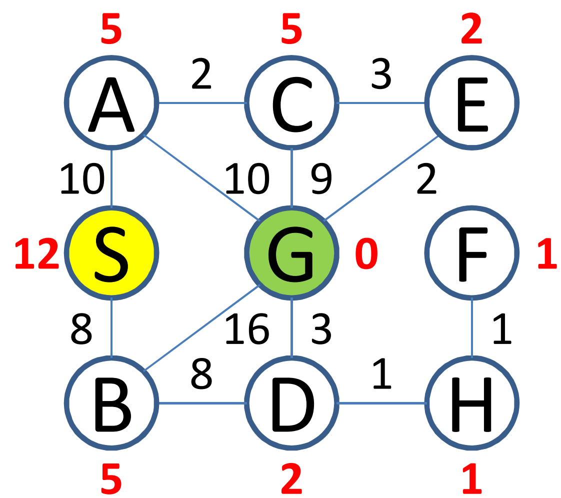
Problem


S
0
12
(18)
17
A
10
5
(20)
17
C
12
5
17
G
20
0
20
E
15
2
17

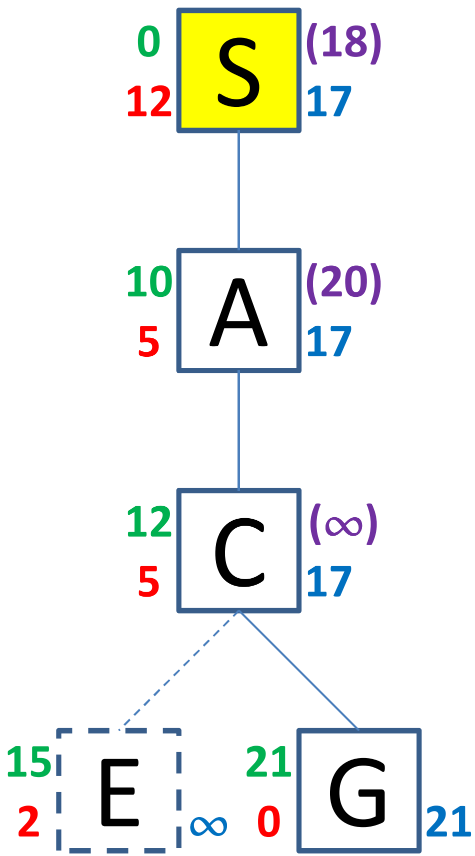

A
C
E
S
G
F
B
D
H
5
5
2
12
0
1
5
2
1
2
3
10
10
9
2
8
16
3
1
8
1

Problem

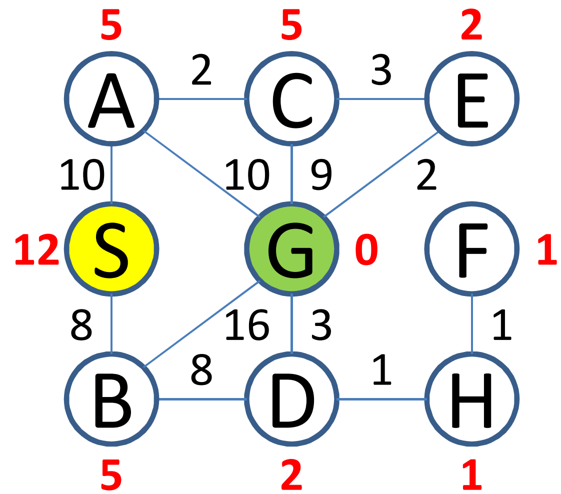

0
S
(18)
12
17
10
A
(20)
5
17
12
C
5
17
15
E
2
~~17~~ ∞


5
5
2
A
2
C
3
E
10
10
9
2
12
S
G
0
F
1
8
16
3
1
8
1
B
D
H
5
2
1

Problem

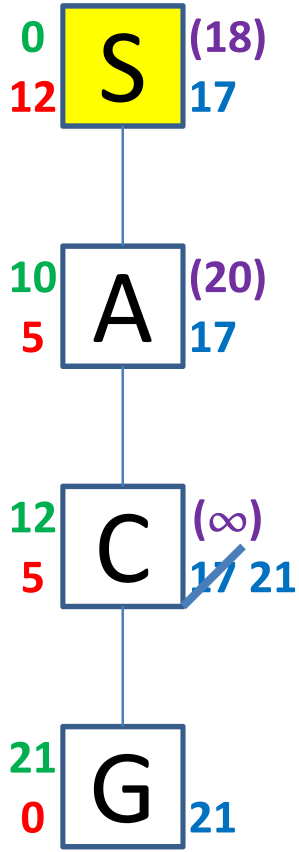


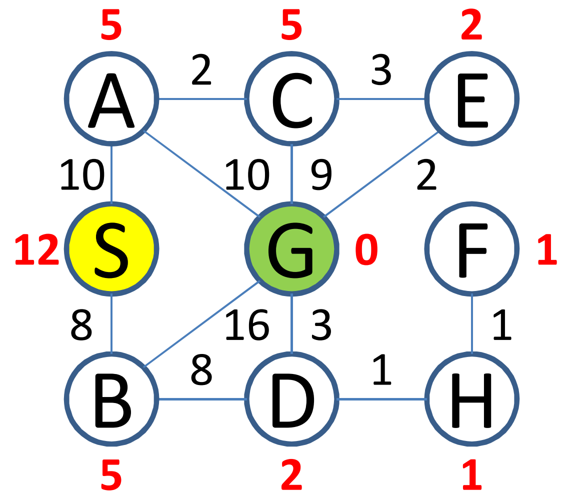
0
S
(18)
12
17
10
A
(20)
5
17
12
C
(∞)
5
17
15
E
2
∞
21
G
0
21



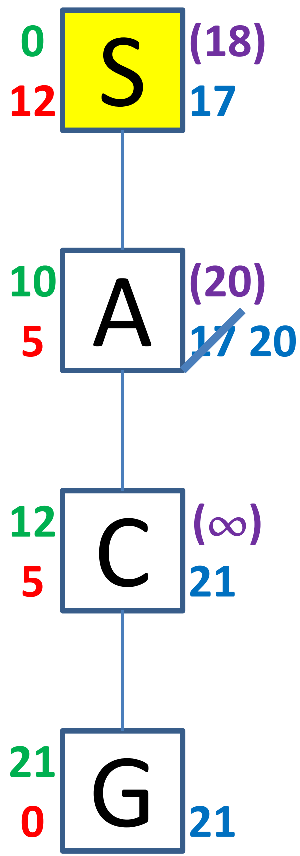
5
A
5
C
2
E
2
3
10
10
9
2
12
S
G
0
F
1
8
16
3
1
8
1
B
D
H
5
2
1

Problem

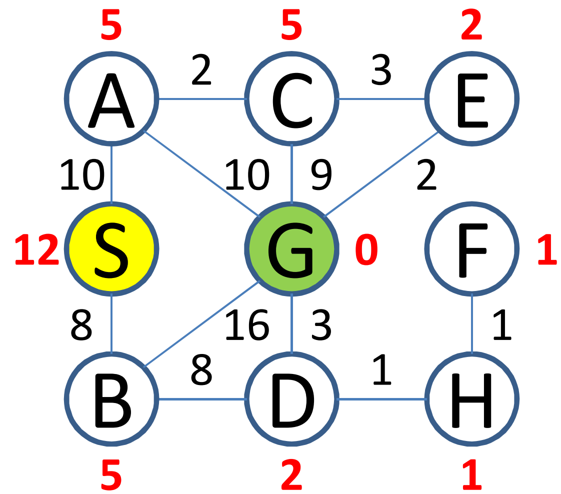

0
S
(18)
12
17
10
A
(20)
5
17
12
C
(∞)
5
17 21
21
G
0
21


5
5
2
A
2
C
3
E
10
10
9
2
12
S
G
0
F
1
8
16
3
1
8
1
B
D
H
5
2
1

Problem

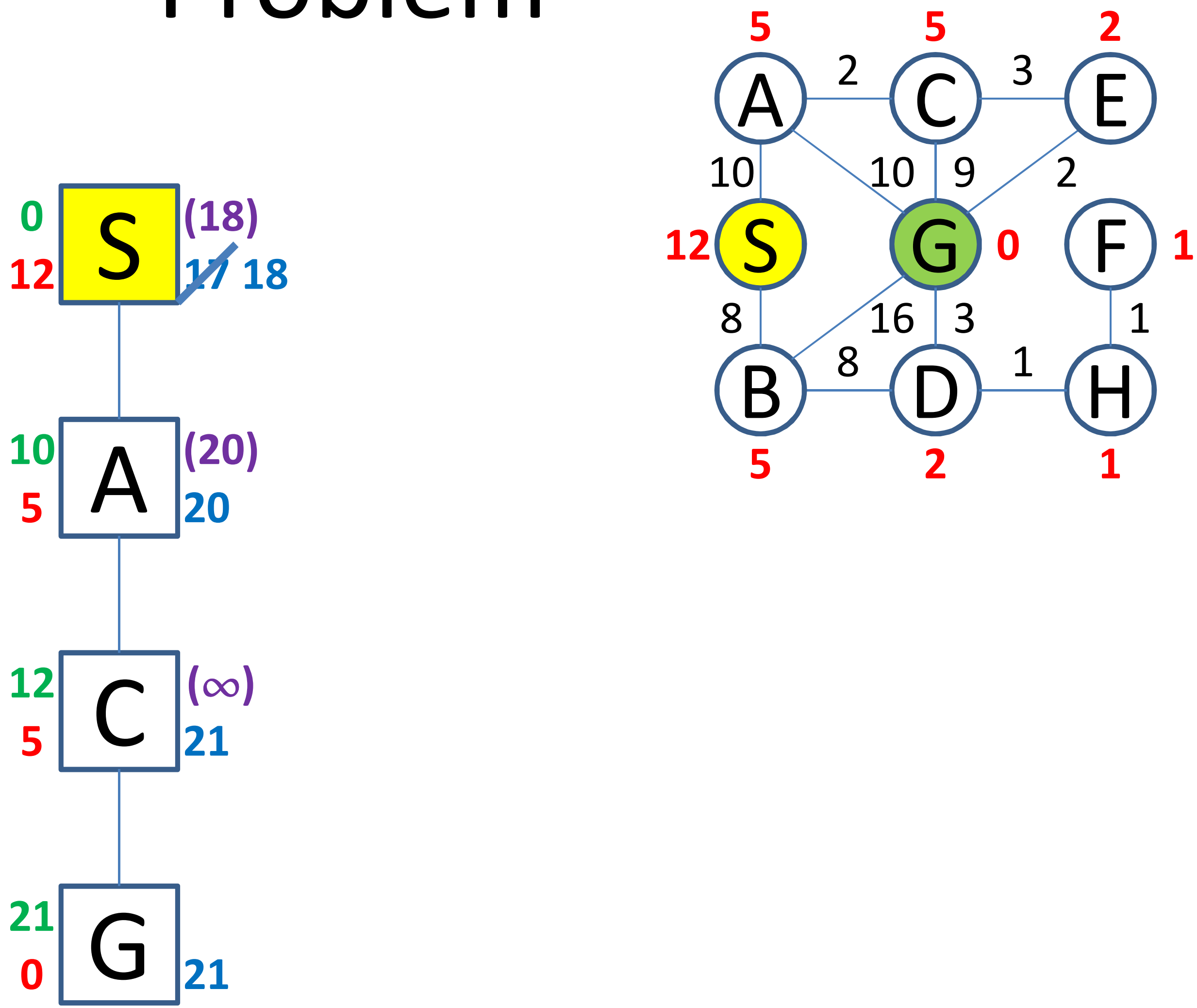


0
S
(18)
12
17
10
A
(20)
5
~~17~~ 20
12
C
(∞)
5
21
21
G
0
21

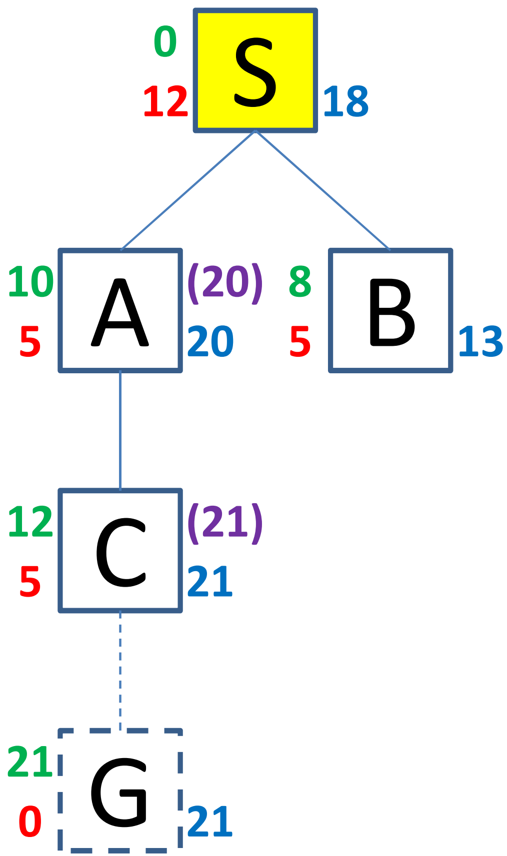


5
5
2
A
2
C
3
E
10
10
9
2
12
S
G
0
F
1
8
16
3
1
B
8
D
1
H
5
2
1

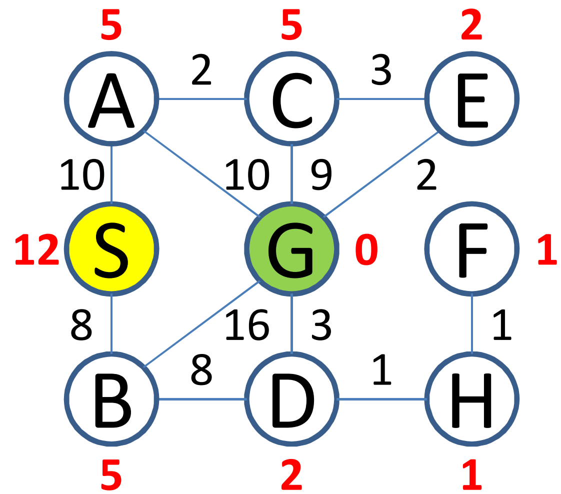
Problem


0
12
S
(18)
~~17~~ 18
10
5
A
(20)
20
12
5
C
(∞)
21
21
0
G
21
A
C
E
S
G
F
B
D
H
5
5
2
12
0
1
5
2
1
2
3
10
10
9
2
8
16
3
1
8
1

Problem

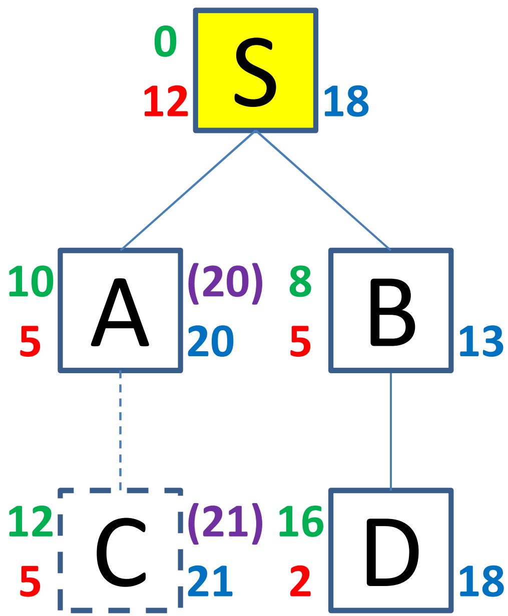


0
S
12
18
10
A
(20)
5
20
8
B
5
13
12
C
(21)
5
21
21
G
0
21

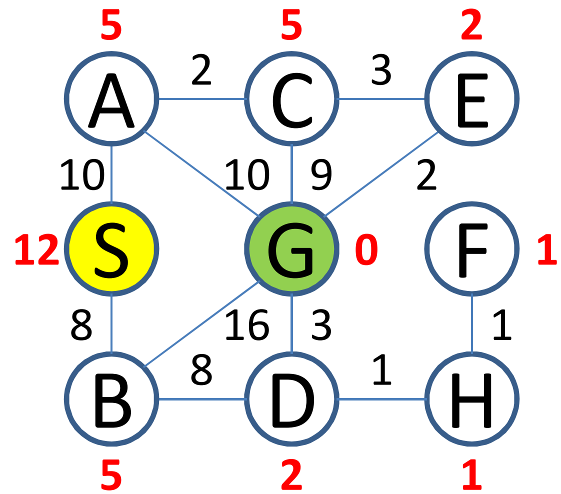


5
A
2
5
C
3
2
E
10
10
9
2
12
S
G
0
F
1
8
16
3
1
8
1
B
D
H
5
2
1

Problem

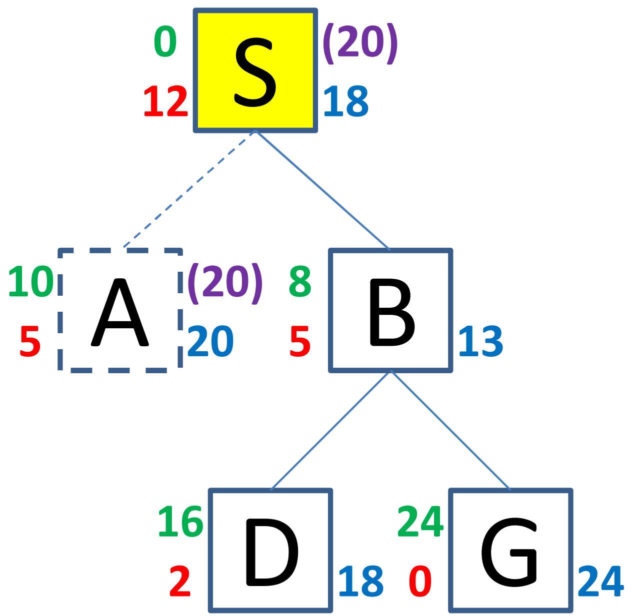


0
12
S
18
10
5
A
(20)
20
8
5
B
13
12
5
C
(21)
21
16
2
D
18

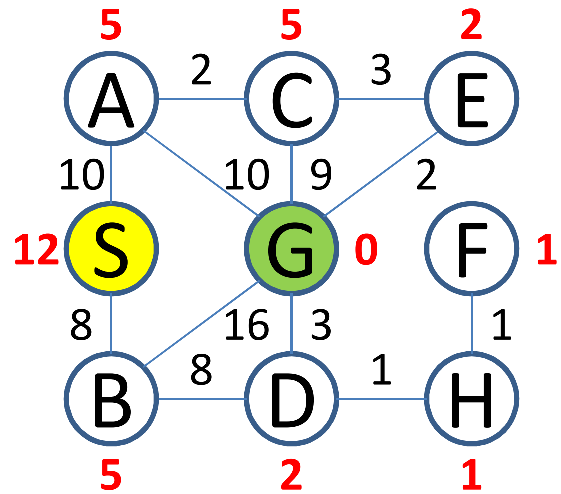


5
A
2
5
C
3
2
E
10
10
9
2
12
S
G
0
F
1
8
16
3
1
8
1
B
5
D
2
H
1

Problem

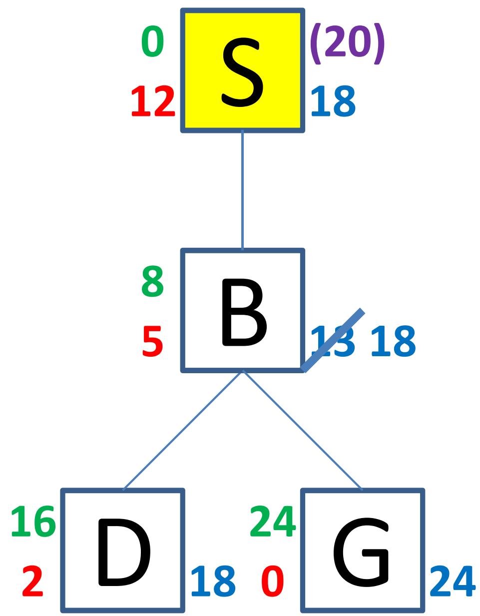


0
12
S
(20)
18
10
5
A
(20)
20
8
5
B
13
16
2
D
18
24
0
G
24

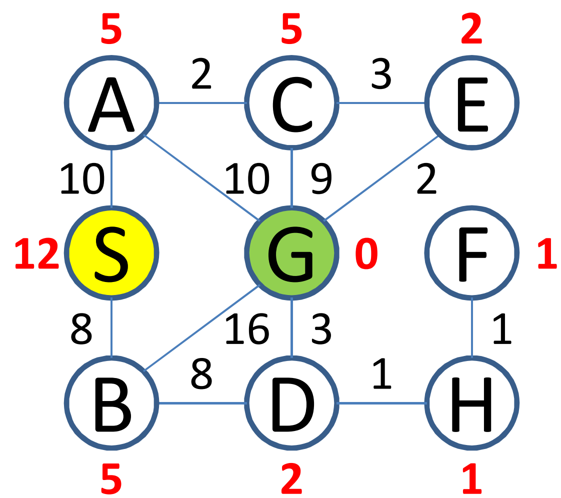


5
A
2
5
C
3
2
E
10
10
9
2
12
S
G
0
F
1
8
16
3
1
8
1
B
D
H
5
2
1

Problem

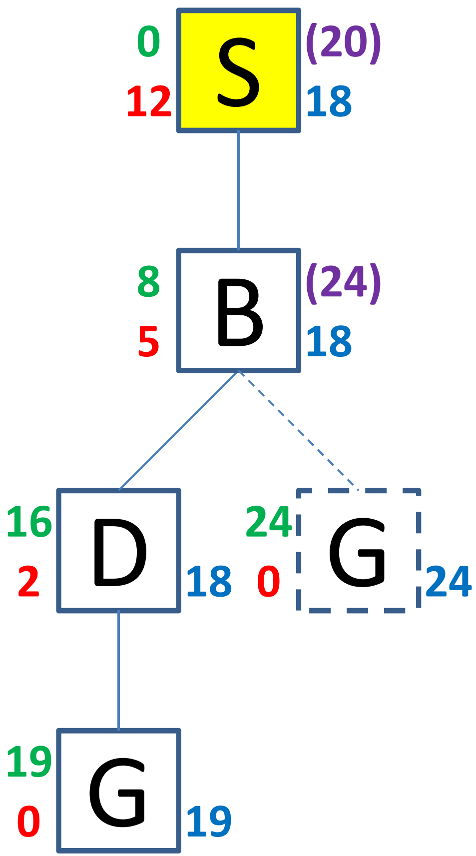


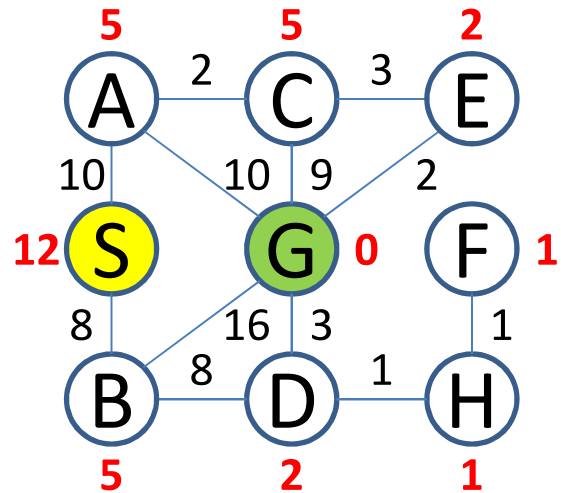
0
S
(20)
12
18
8
B
5
~~13~~ 18
16
D
2
18
24
G
0
24



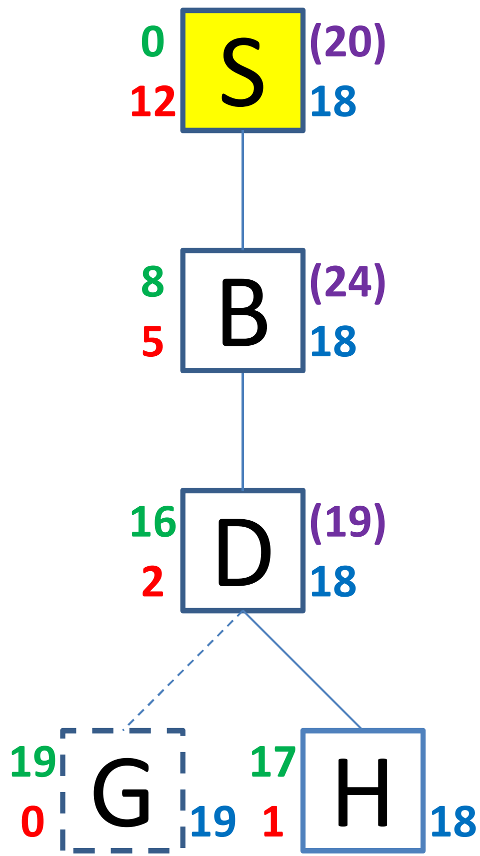
5
5
2
A
2
C
3
E
10
10
9
2
12
S
G
0
F
1
8
16
3
1
8
1
B
D
H
5
2
1

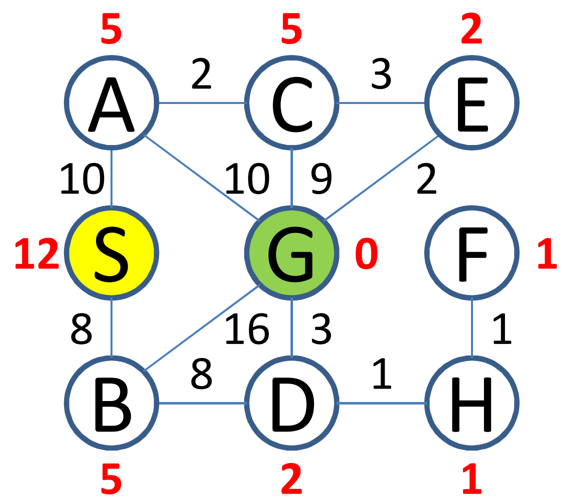
Problem


0
12
S
(20)
18
8
5
B
(24)
18
16
2
D
18
24
0
G
24
19
0
G
19

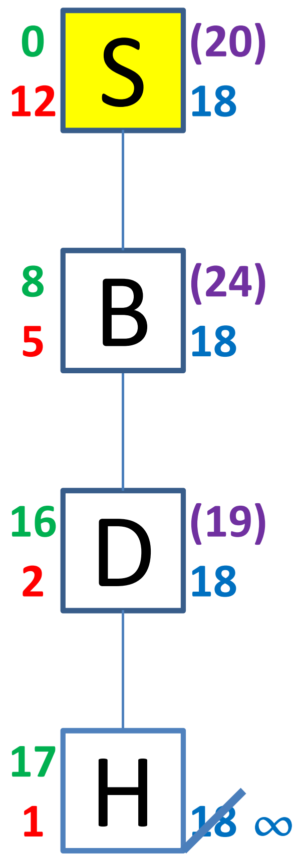

5
5
2
A
2
C
3
E
10
10
9
2
12
S
G
0
F
1
8
16
3
1
B
8
D
1
H
5
2
1

Problem

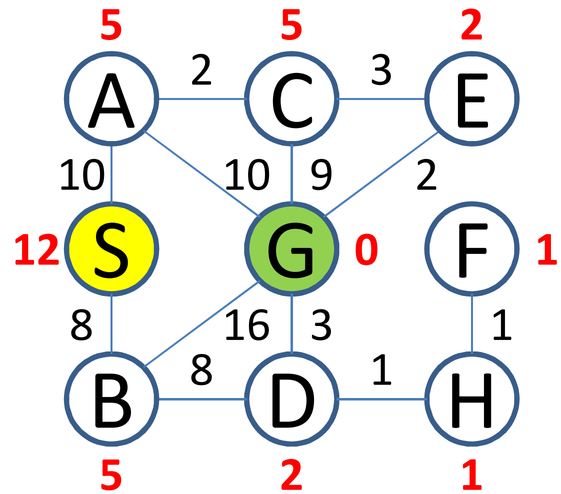

S
0
12
(20)
18
B
8
5
(24)
18
D
16
2
(19)
18
G
19
0
19
H
17
1
18


A
C
E
S
G
F
B
D
H
5
5
2
12
0
1
5
2
1
2
3
10
10
9
2
8
16
3
1
8
1

Problem

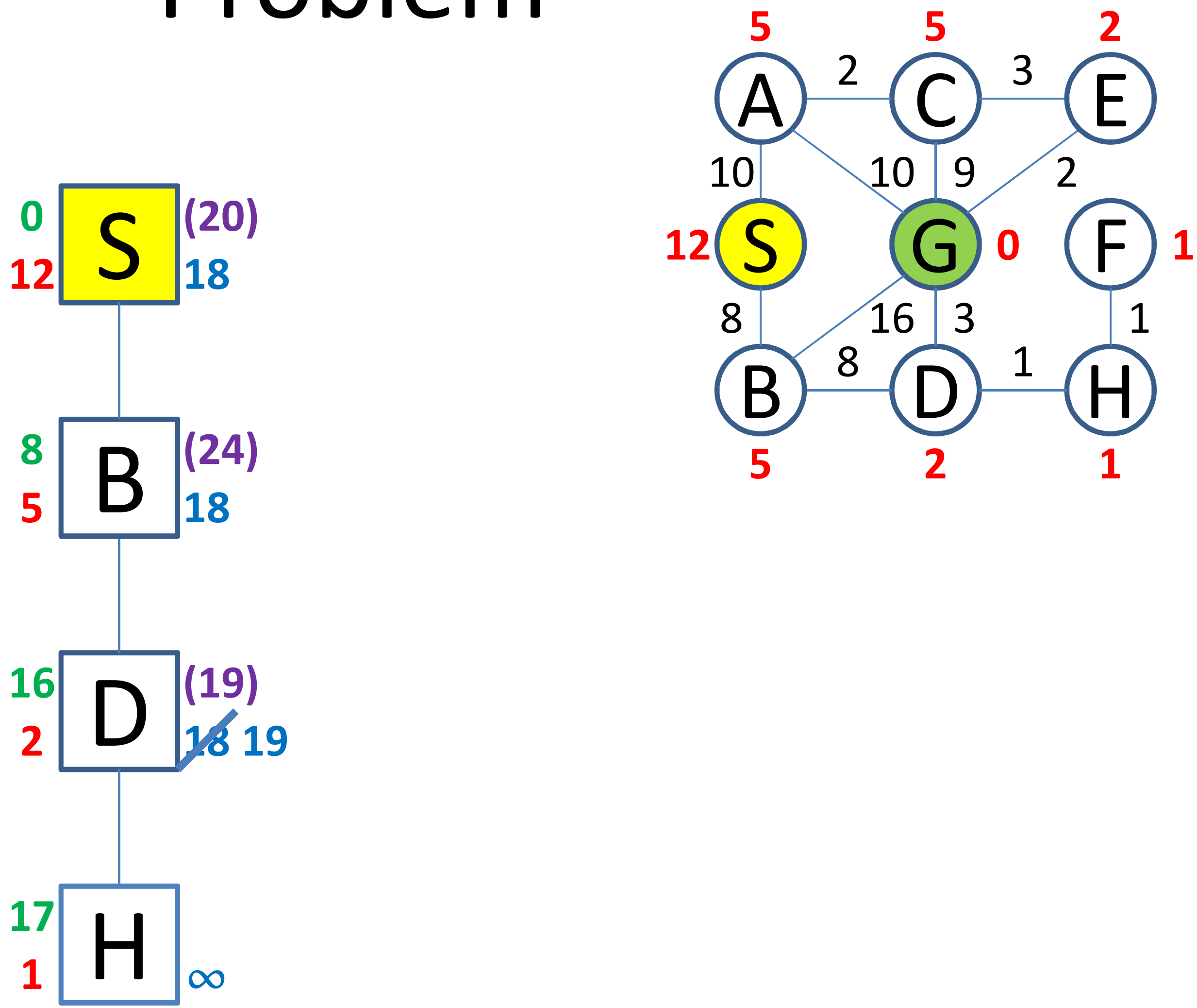


0
12
S
(20)
18
8
5
B
(24)
18
16
2
D
(19)
18
17
1
H
~~18~~ ∞



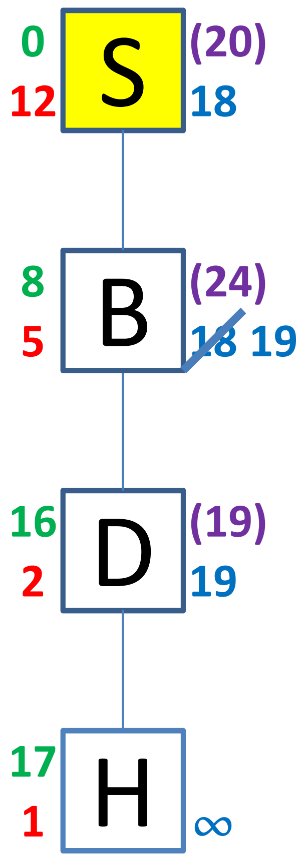
5
5
2
A
2
C
3
E
10
10
9
2
12
S
G
0
F
1
8
16
3
1
B
8
D
1
H
5
2
1

Problem

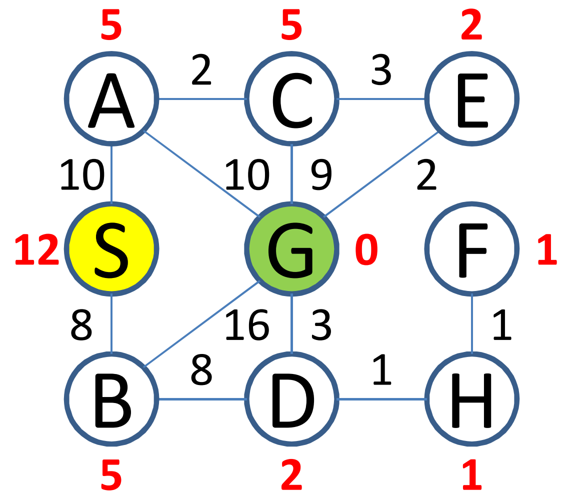


0 S (20)
12 18
8 B (24)
5 18
16 D (19)
2 ~~18~~ 19
17 H
1 ∞
A C E F S G B D H
5 5 2 12 0 1 5 2 1
2 3 10 10 9 2 8 16 3 1 8 1

Problem

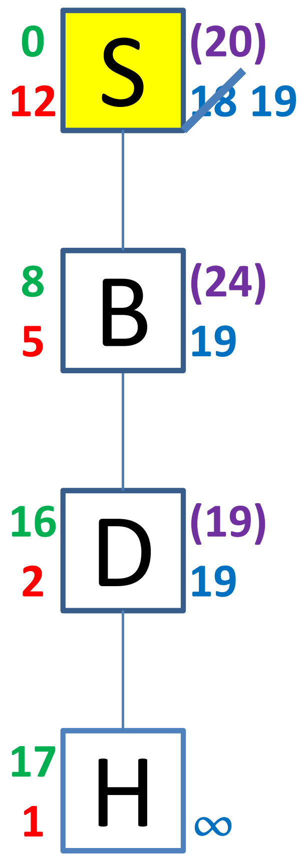


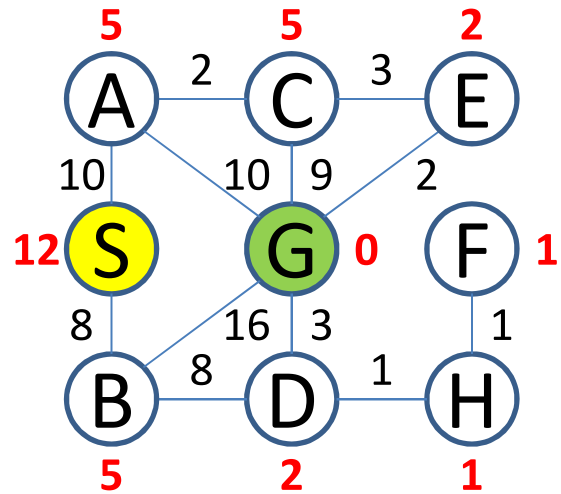
0
12
S
(20)
18
8
5
B
(24)
18 19
16
2
D
(19)
19
17
1
H
∞



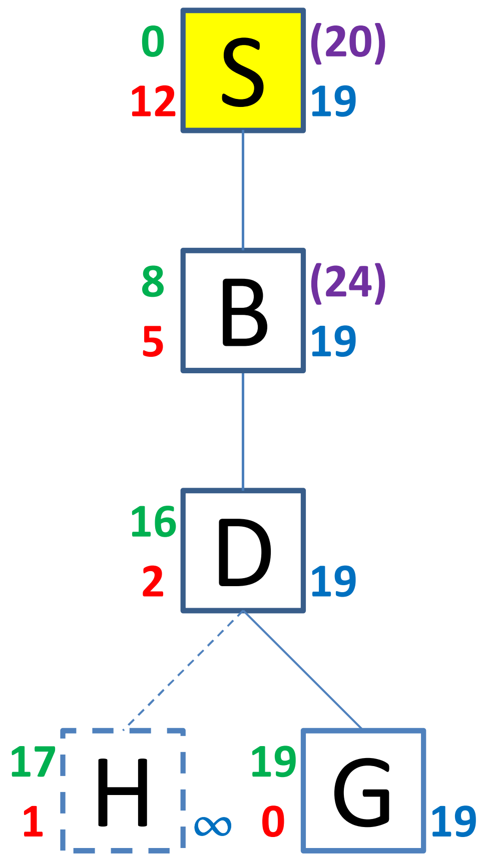
5
5
2
A
2
C
3
E
10
10
9
2
12
S
G
0
F
1
8
16
3
1
B
8
D
1
H
5
2
1

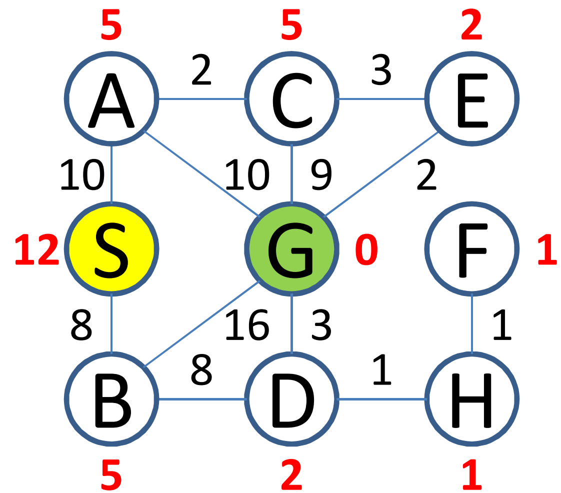
Problem


0
12
S
(20)
18 19
8
5
B
(24)
19
16
2
D
(19)
19
17
1
H
∞

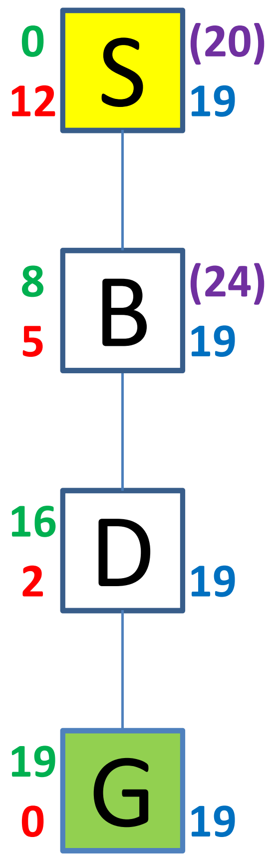

5
5
2
A
2
C
3
E
10
10
9
2
12
S
G
0
F
1
8
16
3
1
B
8
D
1
H
5
2
1

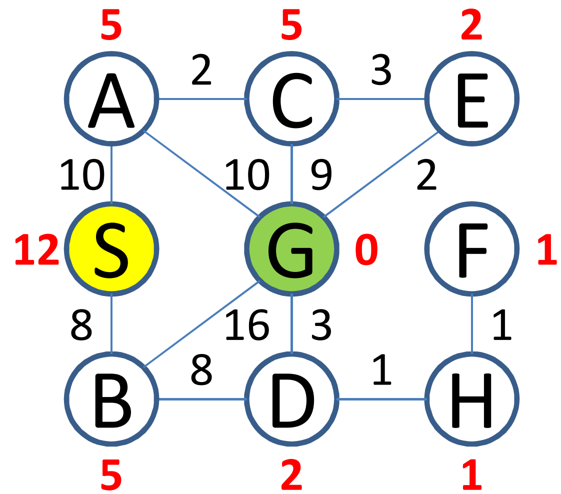
Problem


0
S
(20)
12
19
8
B
(24)
5
19
16
D
2
19
17
H
1
∞
19
G
0
19


5
5
2
A
2
C
3
E
10
10
9
2
12
S
G
0
F
1
8
16
3
1
B
8
D
1
H
5
2
1

Problem


0
S
(20)
12
19
8
B
(24)
5
19
16
D
2
19
19
G
0
19


5
5
2
A
2
C
3
E
10
10
9
2
12
S
G
0
F
1
8
16
3
1
8
1
B
D
H
5
2
1

Exercises: Artificial Intelligence

Monotonicity 1

Monotonicity 1

PROBLEM

Problem

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

Monotonicity 1

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

f(S…A) = cost(S…A) + **h(A)**

Monotonicity 1

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

f(S…A) = cost(S…A) + **h(A)**

≤ cost(S…A) + **cost(A…B) + h(B)**

Monotonicity 1

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

f(S…A) = cost(S…A) + **h(A)**

≤ cost(S…A) + **cost(A…B) + h(B)**

≤ cost(S…A…B) + h(B)

Monotonicity 1

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

f(S…A) = cost(S…A) + **h(A)**

≤ cost(S…A) + **cost(A…B) + h(B)**

≤ cost(S…A…B) + h(B)

≤ f(S…A…B)

Exercises: Artificial Intelligence

Monotonicity 2

Monotonicity 2

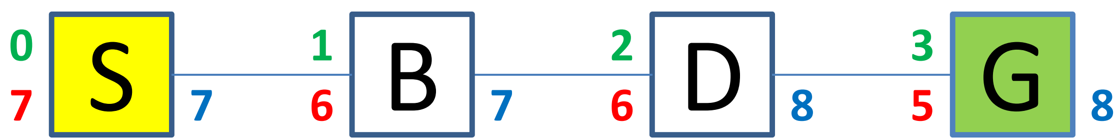
PROBLEM

Problem

- Prove or refute:
  - **IF** *f* is *monotonously non-decreasing*
    - f(s…x) ≤ f(s…xy)
  - **THEN** *h* is an *admissable 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 *mononously non-decreasing*
- *Proof (Counter-example):*



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

Monotonicity 2

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

  f(S…A) ≤ f(S…AB) ≤ … ≤ f(S…AB…G)

Monotonicity 2

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

  f(S…A) ≤ f(S…AB) ≤ … ≤ f(S…AB…G) ⟺

  f(S…A) ≤ f(S…G)

Monotonicity 2

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

  f(S…A) ≤ f(S…AB) ≤ … ≤ f(S…AB…G) $\Leftrightarrow$

  f(S…A) ≤ f(S…G) $\Leftrightarrow$

  cost(S…A) + h(A) ≤ cost(S…G) + h(G)

Monotonicity 2

- *Given:*
  - ***f*** is ***mononousl******y non-decreasing***
  - Extra constraint: ***h(G) = 0***
- *Proof:*

f(S…A) ≤ f(S…AB) ≤ … ≤ f(S…AB…G) $\Leftrightarrow$

f(S…A) ≤ f(S…G) $\Leftrightarrow$

cost(S…A) + h(A) ≤ cost(S…G) + h(G) $\Leftrightarrow$

cost(S…A) + h(A) ≤ **cost(S…A)** + cost(A…G) + h(G)

Monotonicity 2

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

  f(S…A) ≤ f(S…AB) ≤ … ≤ f(S…AB…G) ⇔

  f(S…A) ≤ f(S…G) ⇔

  cost(S…A) + h(A) ≤ cost(S…G) + h(G) ⇔

  cost(S…A) + h(A) ≤ **cost(S…A)** + cost(A…G) + h(G) ⇔

  h(A) ≤ cost(A…G) + **h(G)**

Monotonicity 2

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

  f(S…A) ≤ f(S…AB) ≤ … ≤ f(S…AB…G) ⇔

  f(S…A) ≤ f(S…G) ⇔

  cost(S…A) + h(A) ≤ cost(S…G) + h(G) ⇔

  cost(S…A) + h(A) ≤ **cost(S…A)** + cost(A…G) + h(G) ⇔

  h(A) ≤ cost(A…G) + **h(G)** ⇔

  h(A) ≤ cost(A…G)