Fundamentals of Artificial Intelligence

Laboratory

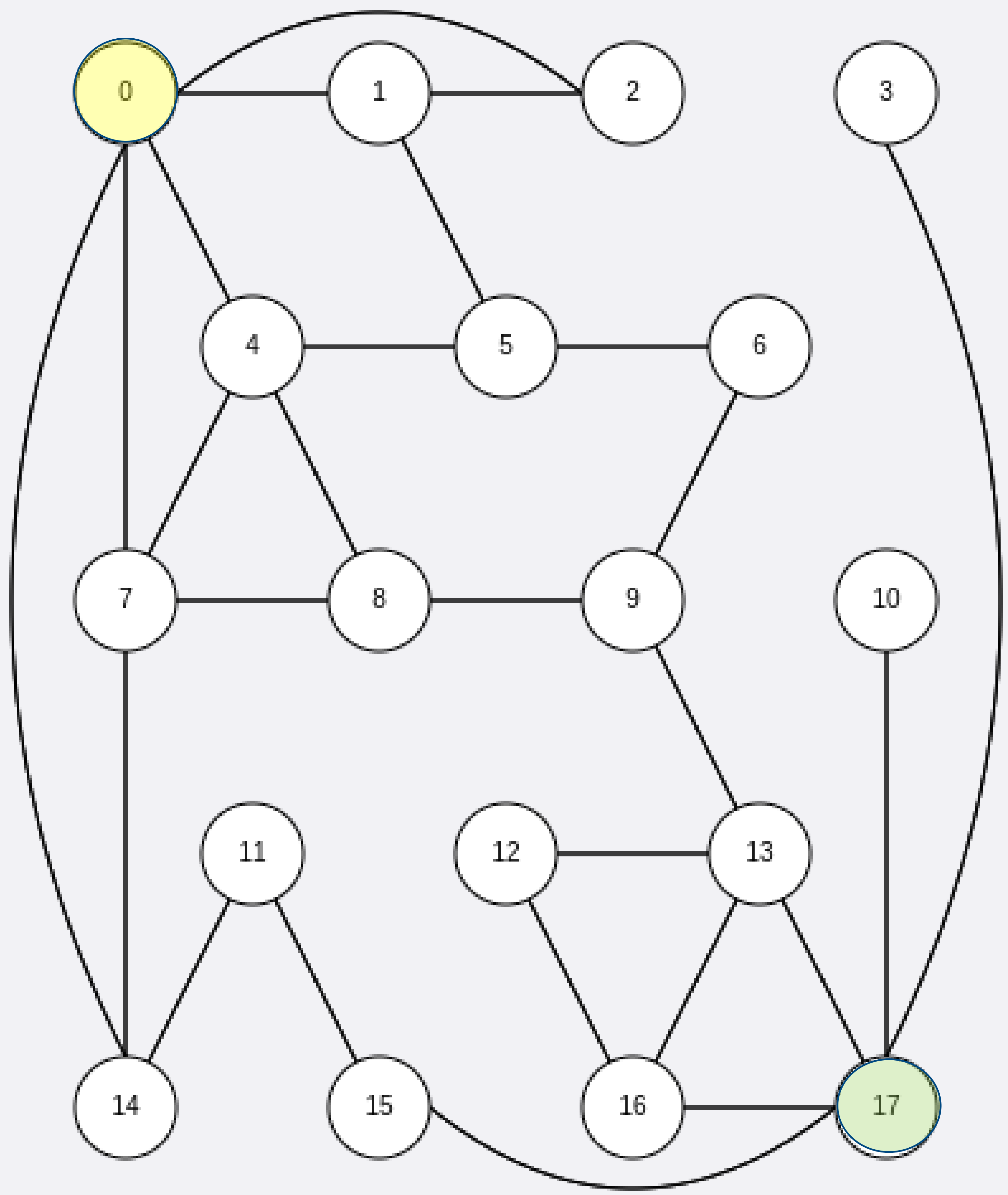
Dr. Mauro Dragoni

Department of Information Engineering and Computer Science

Academic Year 2021/2022

Exercise 3.10

- Apply both the **iterative deepening depth-first search** and the **bidirectional search** for reaching the goal (N-17) from the start (N-0)



Exercise 3.10 - Solution

- In order to avoid misunderstanding and to do not create confusion, we apply the algorithm as it is explained in the book without considering possible variants.

- **Iterative deepening**

  d0 = {0}
  d1 = {0,1,2,4,7,14}
  d2 = {0,1,2,4,7,14,5,8,11}
  d3 = {0,1,2,4,7,14,5,8,11,6,9,15}
  d4 = {0,1,2,4,7,14,5,8,11,6,9,15,13,**17**}

Exercise 3.10 - Solution

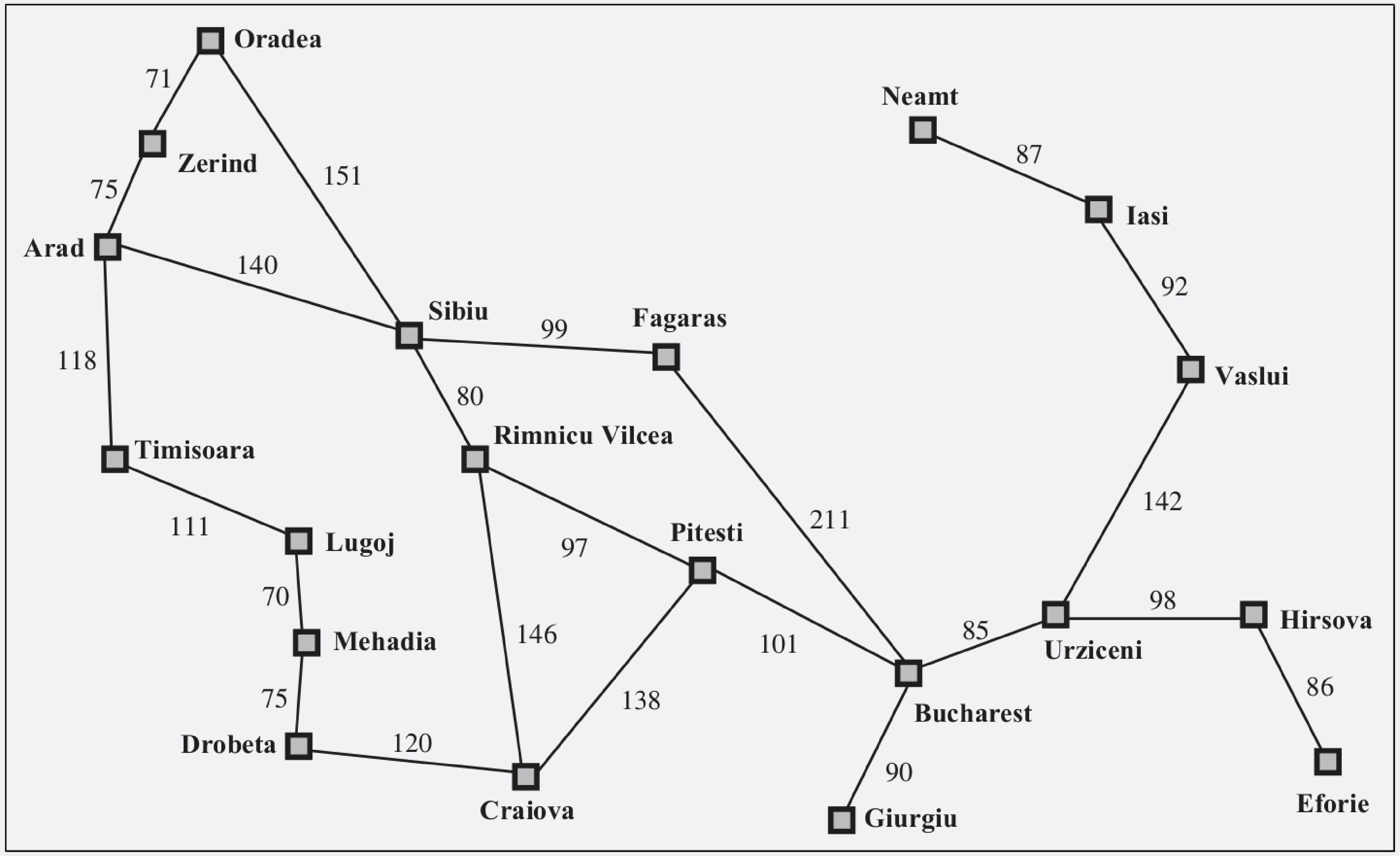
- In order to avoid misunderstanding and to do not create confusion, we apply the algorithm as it is explained in the book without considering possible variants.

- **Bidirectional search (by applying breadth-first)**
  Step0 = {0} {17}
  Step1 = {0,1,2,4,7,14} {17,3,10,13,15,16}
  Step2 = {0,1,2,4,7,14,5,8,**11**} {17, 3,10,13,15,16,9,12,**11**}

- **Bidirectional search (by applying depth-first)**
  Step0 = {0} {17}
  Step1 = {0,1} {17,3}
  Step2 = {0,1,2} {17,3,10}
  Step3 = {0,1,2,5} {17,3,10,13}
  Step4 = {0,1,2,5,4} {17,3,10,13,9}
  Step5 = {0,1,2,5,4,7} {17,3,10,13,9,6}
  Step6 = {0,1,2,**5**,4,7,8} {17,3,10,13,9,6,**5**}

Exercise 3.11

- Apply the **greedy best-first search** strategy for finding the route from Lugoj to Bucharest.



Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

Exercise 3.11 - Solution

- Apply the **greedy best-first search** strategy for finding the route from Lugoj to Bucharest.

- Initial state: Lugoj(244)

  Step1, expanding Lugoj: Mehadia(241), Timisoara(329)

  Step2, expanding Mehadia: Lugoj(244), Drobeta(242)

  Step3, expanding Drobeta: Mehadia(241), Craiova(160)

  Step4, expanding Craiova: Drobeta(242), Rimnicu Vilcea(193), Pitesti(100)

  Step5, expanding Pitesti: Craiova(160), Rimnicu Vilcea(193), **Bucharest(0)**

Exercise 3.12

- A* algorithm

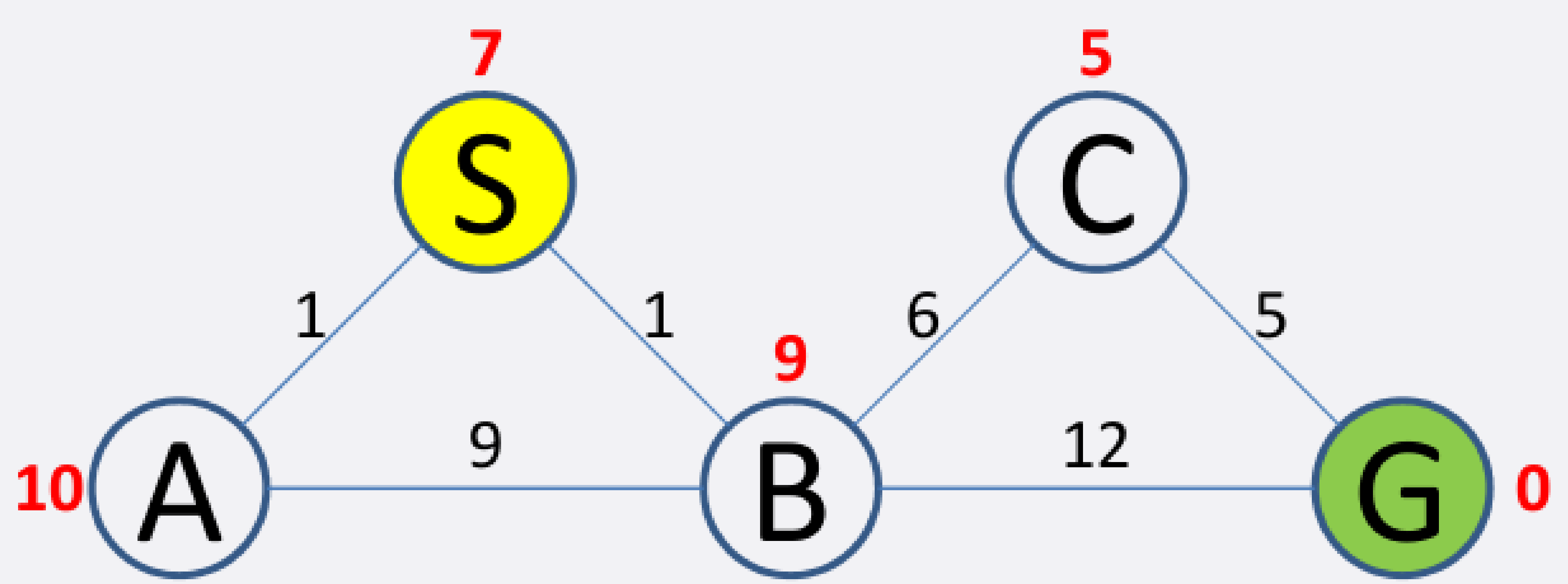
```
------------------
WHILE (QUEUE not empty && first path not reach goal) DO
    Remove first path from QUEUE
    Create paths to all children
    Reject paths with loops
    Add paths and sort QUEUE (by f = cost + heuristic)
    IF QUEUE contains paths: P, Q
        AND P ends in node Ni && Q contains node Ni
        AND cost(P) ≥ cost(Q)
    THEN remove P

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

Exercise 3.12

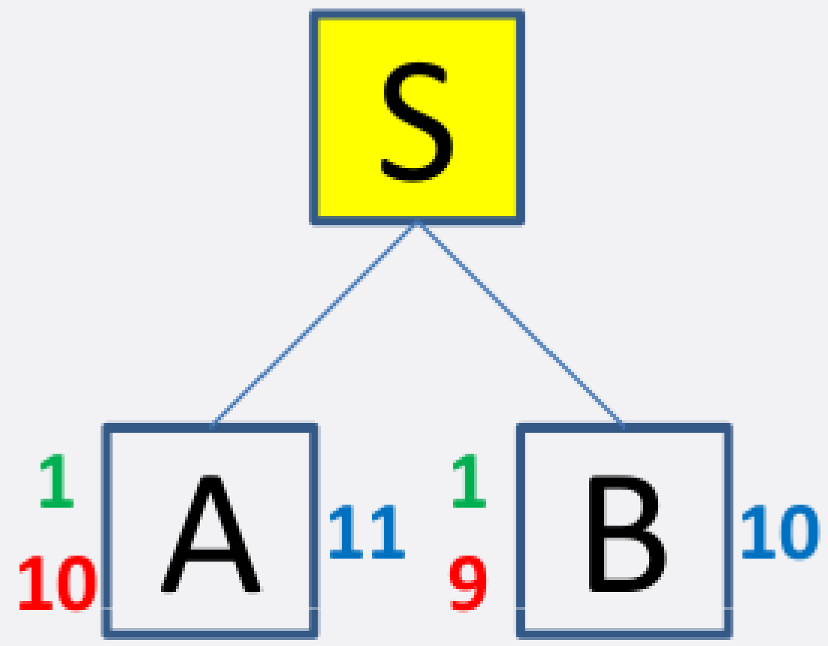
f = accumulated path cost + heuristic

QUEUE = path containing root

QUEUE = <S>



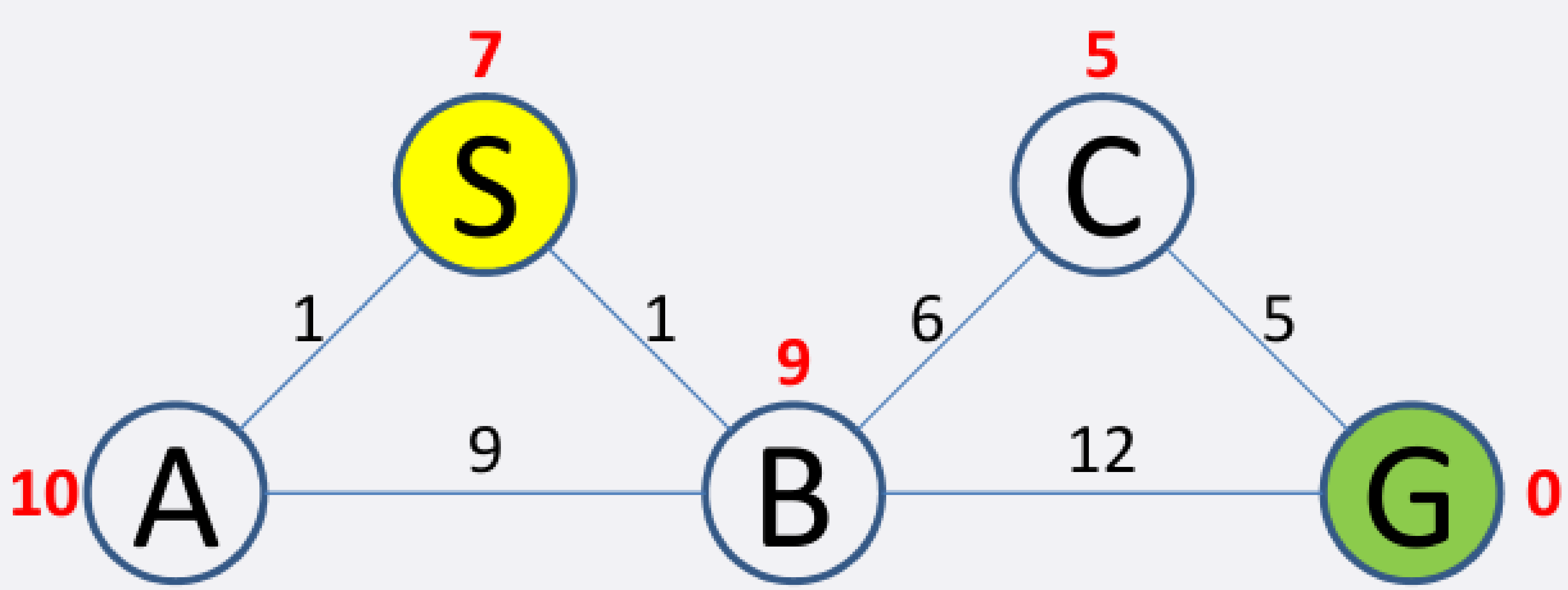
Exercise 3.12



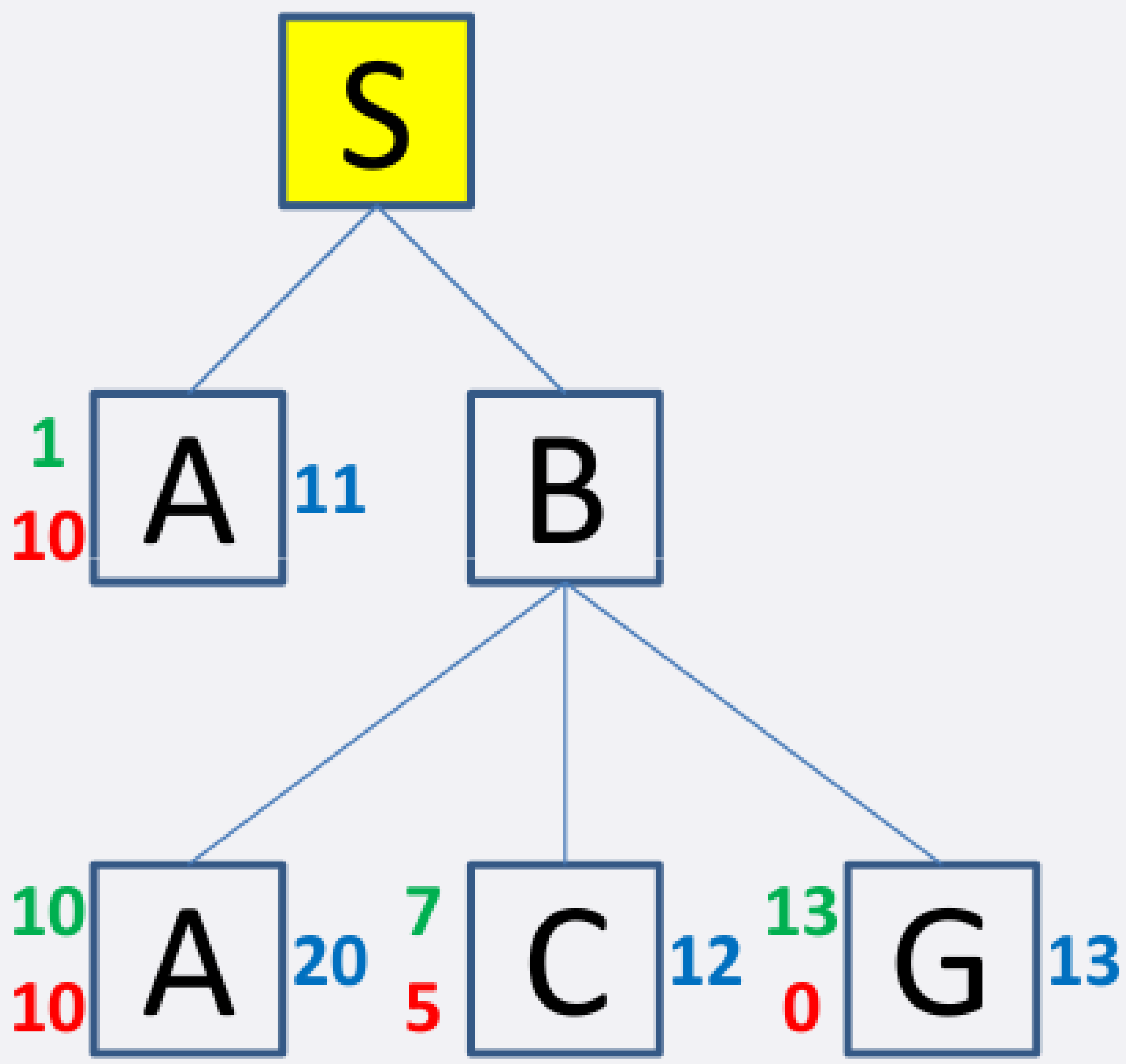
f = accumulated path cost + heuristic

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

QUEUE = <SB,SA>



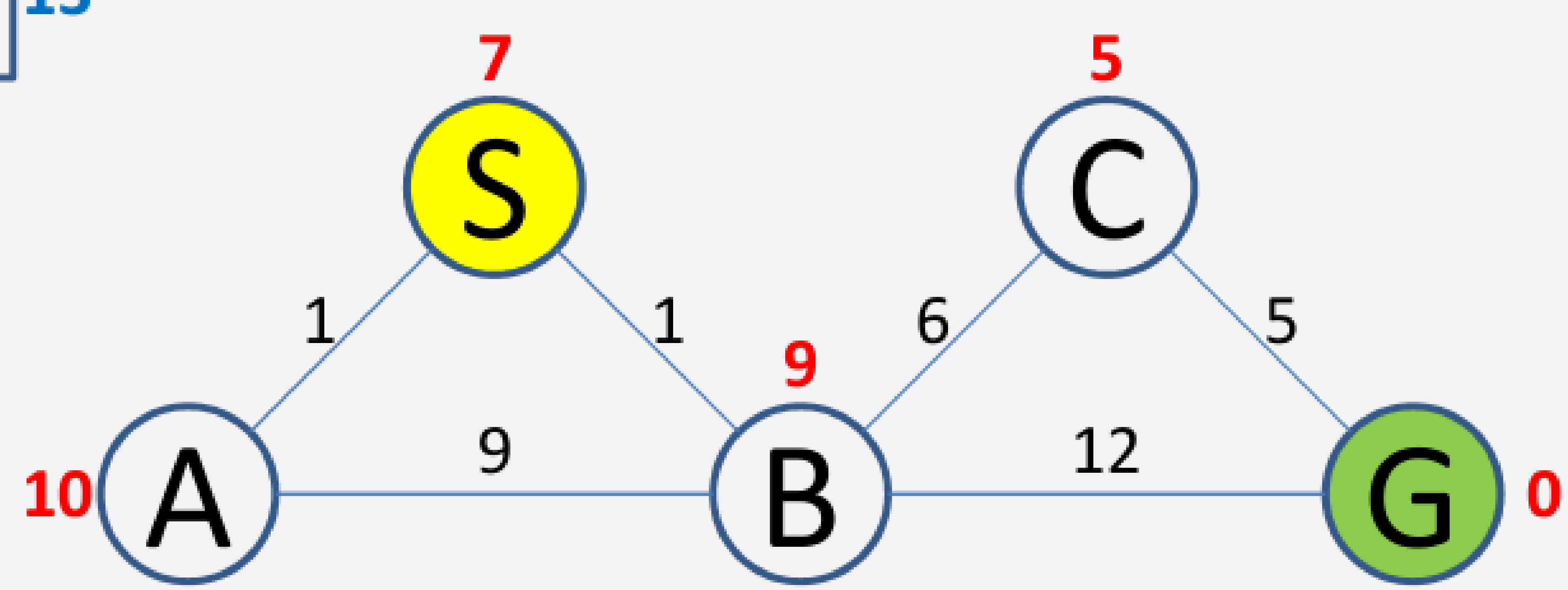
Exercise 3.12



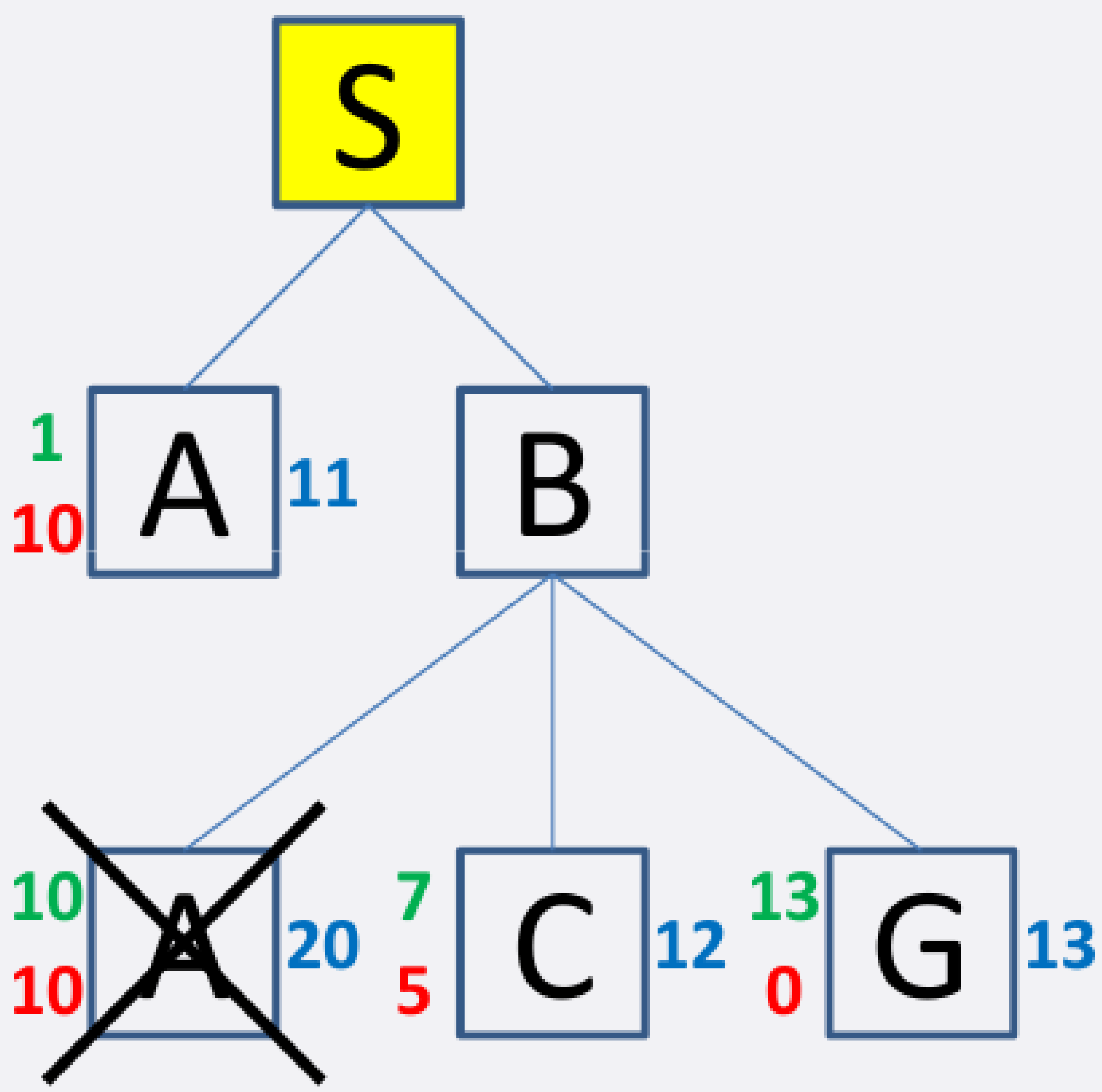
f = accumulated path cost + heuristic

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

QUEUE = <SA,SBC,SBG,SBA>



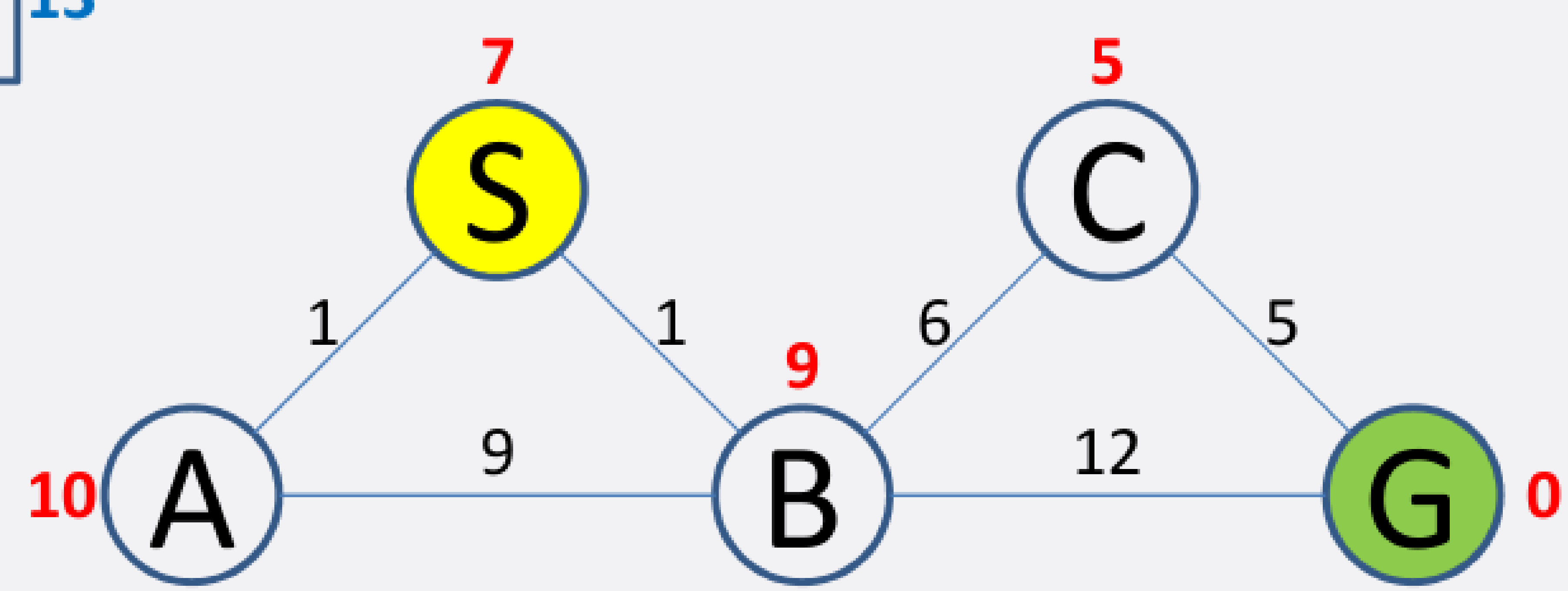
Exercise 3.12



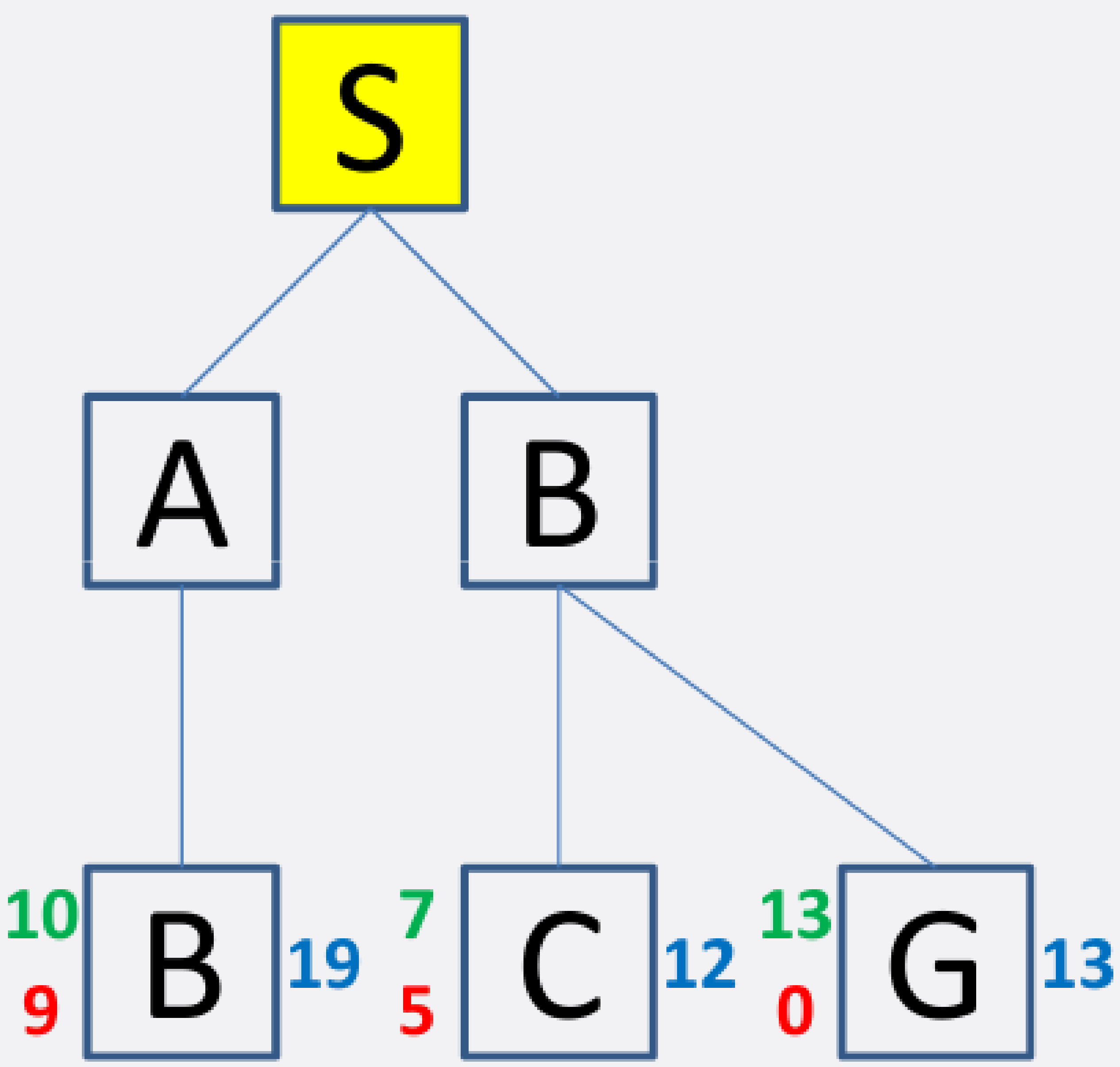
f = accumulated path cost + heuristic

```
IF QUEUE contains paths: P, Q
   AND P ends in node Ni && Q contains node Ni
   AND cost(P) ≥ cost(Q)
THEN remove P
```

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



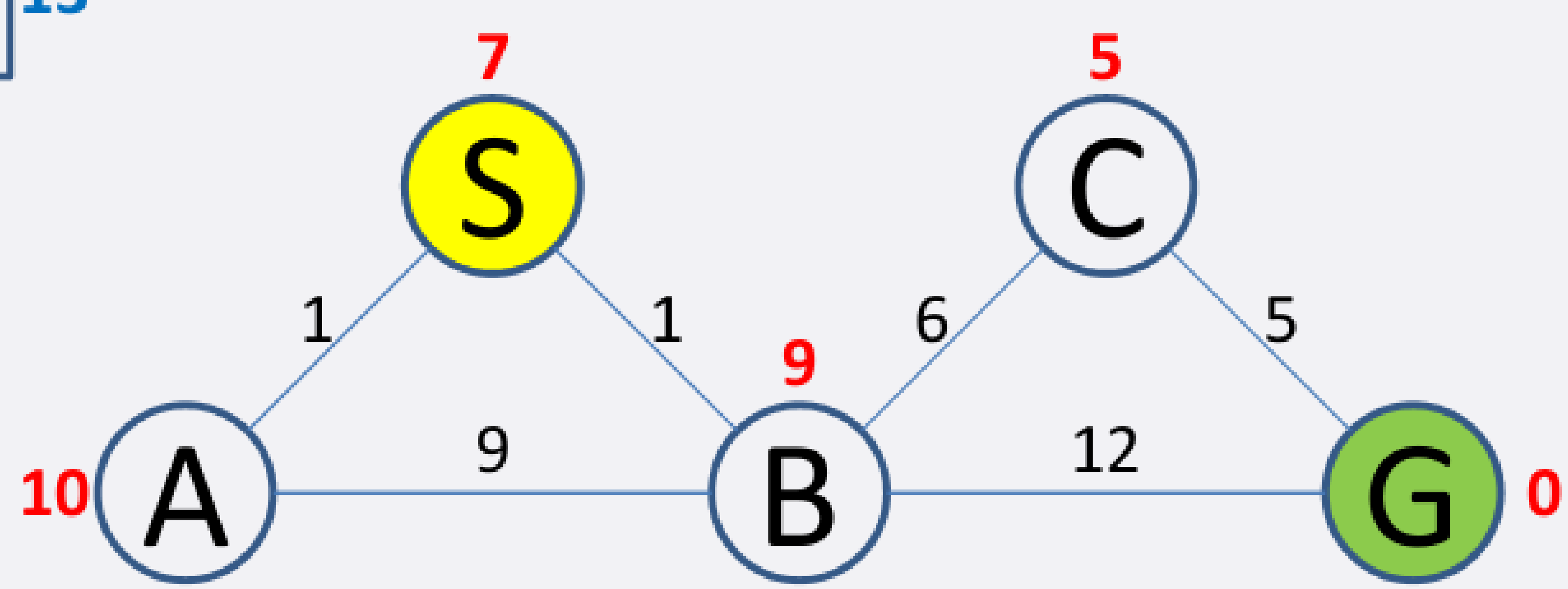
Exercise 3.12



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>

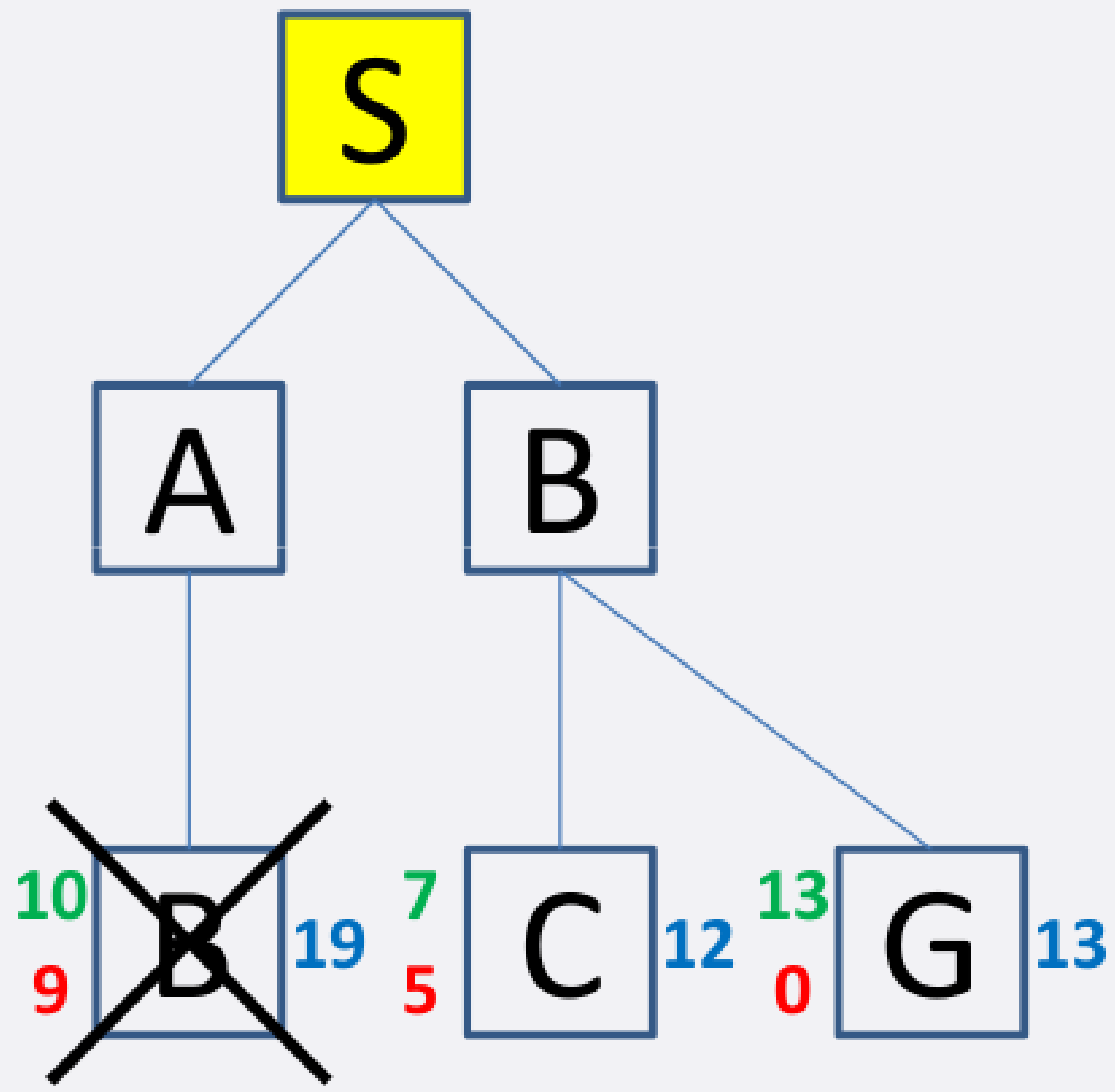


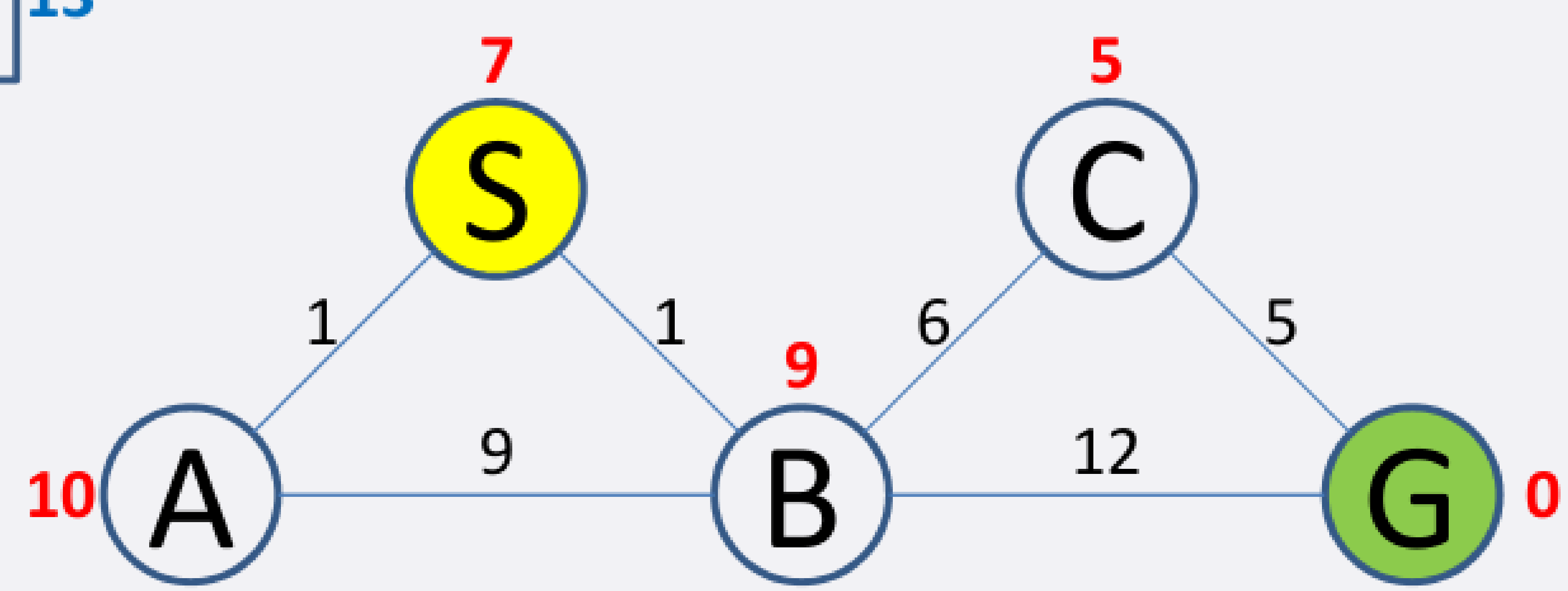
Exercise 3.12

f = accumulated path cost + heuristic

```
IF QUEUE contains paths: P, Q
   AND P ends in node Ni && Q contains node Ni
   AND cost(P) ≥ cost(Q)
THEN remove P
```

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



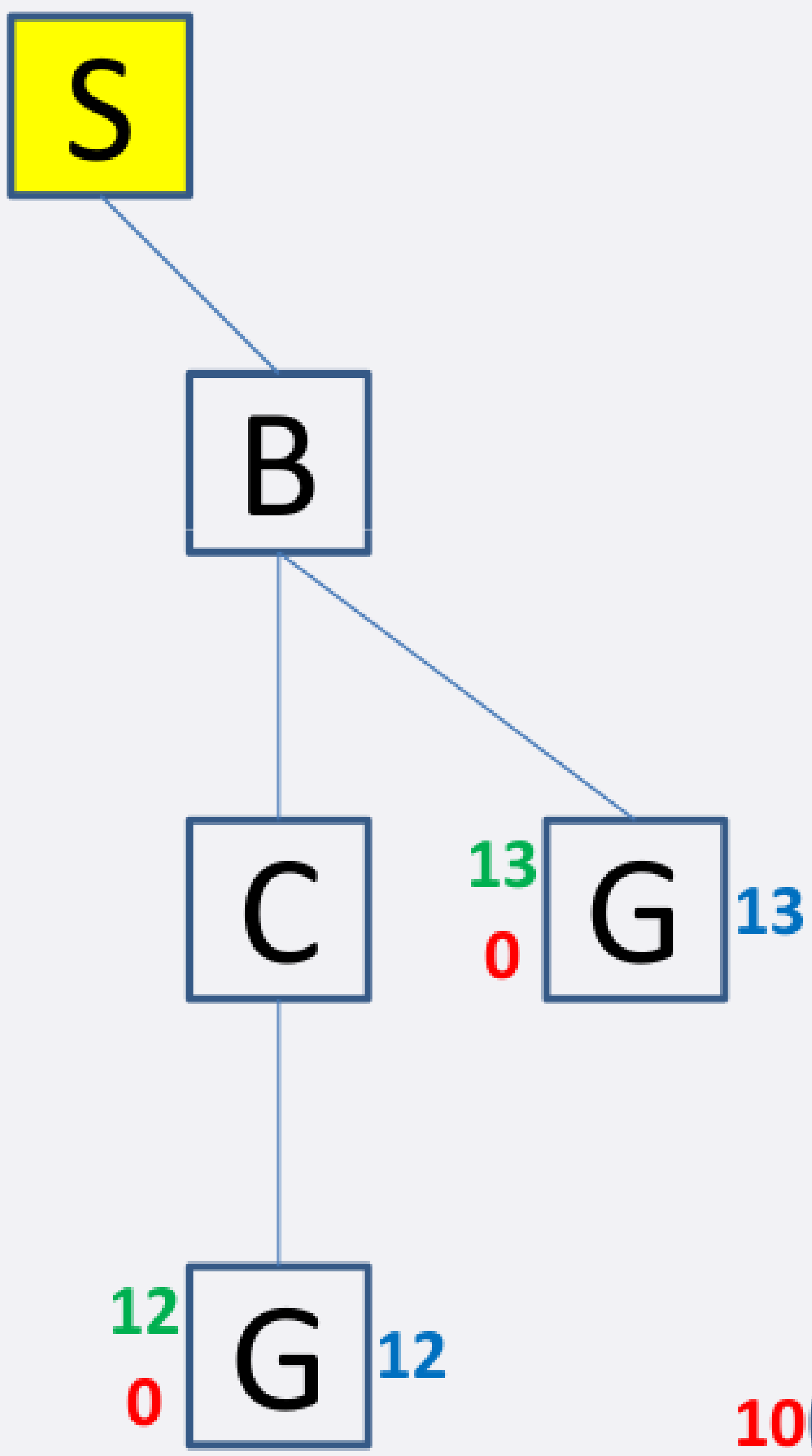


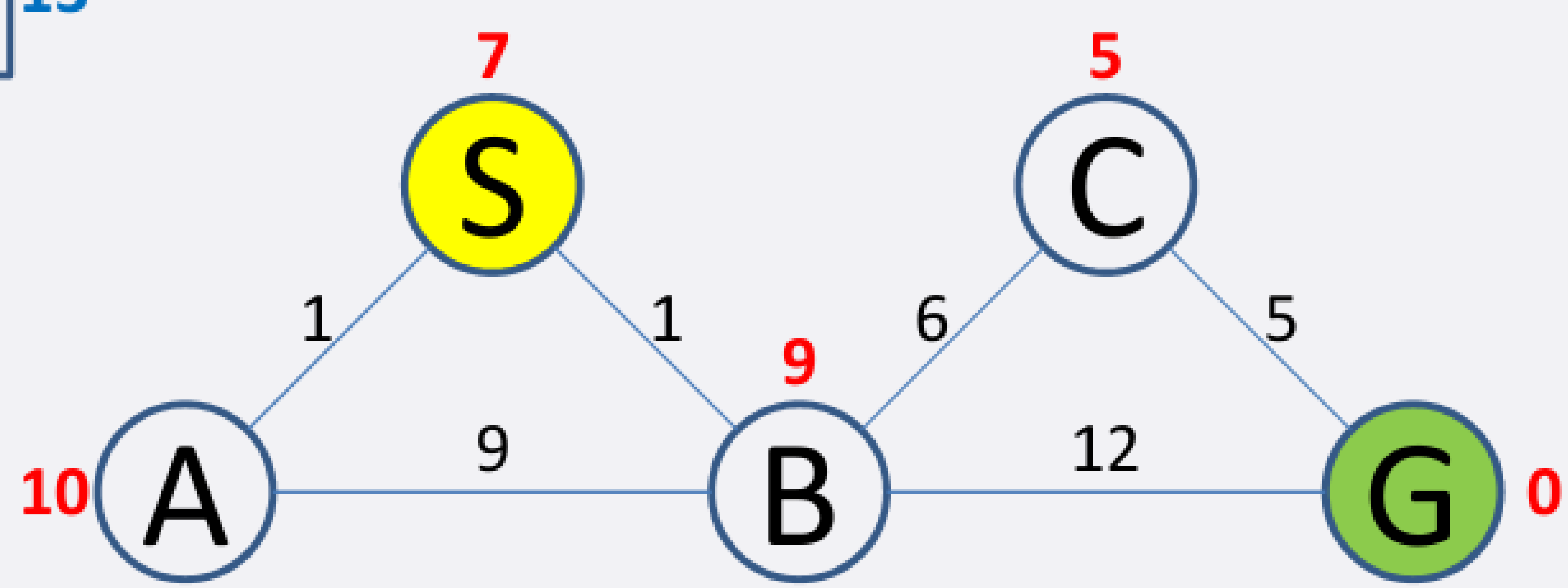
Exercise 3.12

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>



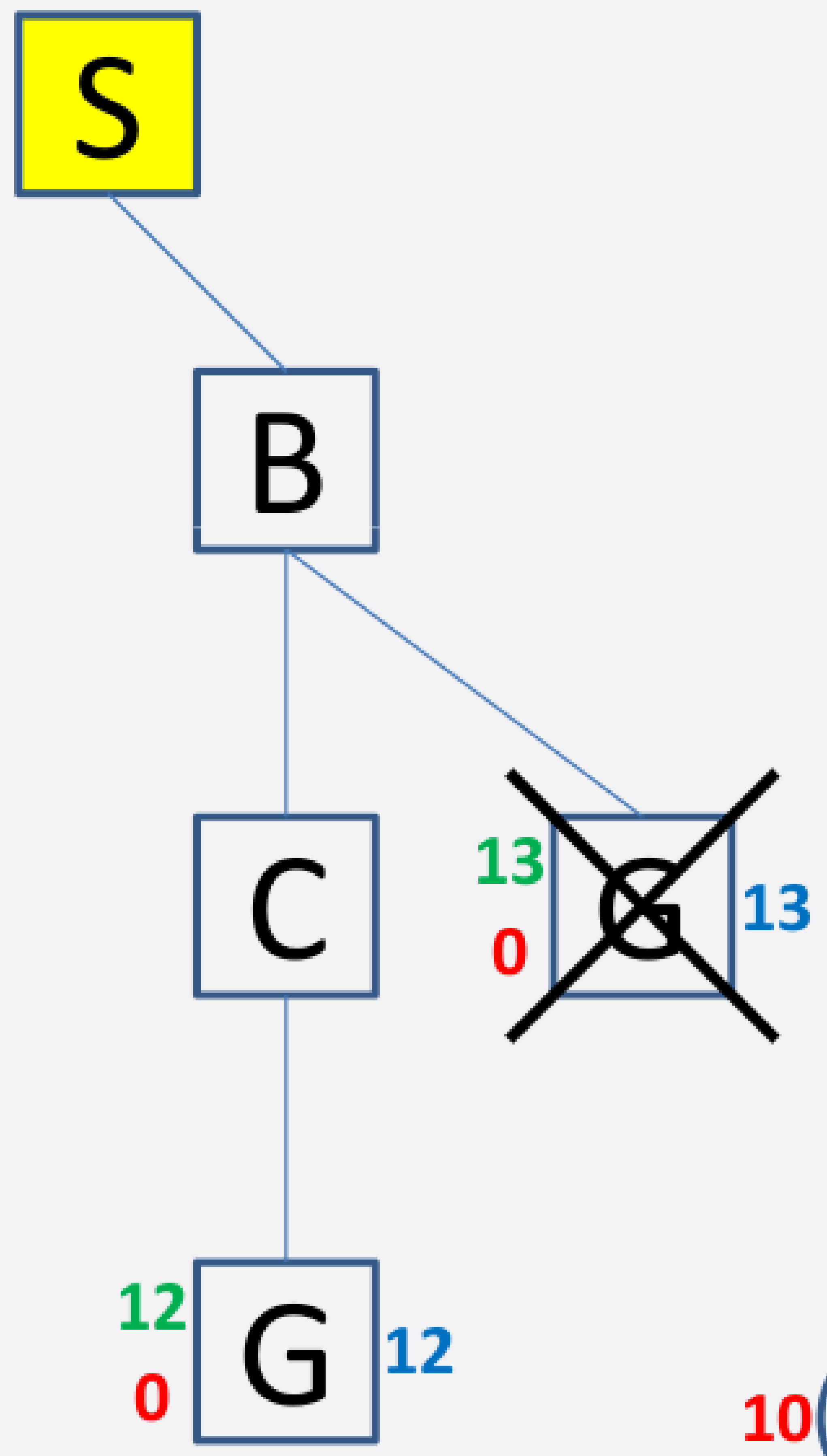


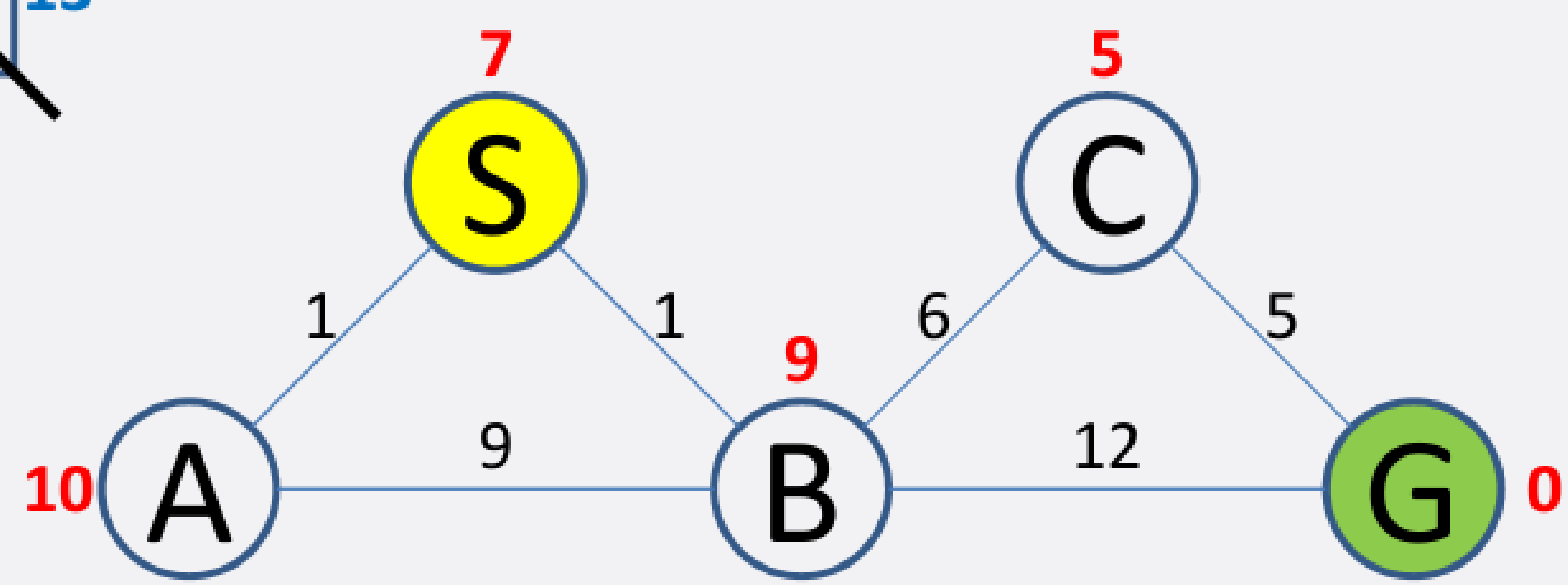
Exercise 3.12

f = accumulated path cost + heuristic

```
IF QUEUE contains paths: P, Q
   AND P ends in node Ni && Q contains node Ni
   AND cost(P) ≥ cost(Q)
THEN remove P
```

QUEUE = < SBCG,**SBG**>



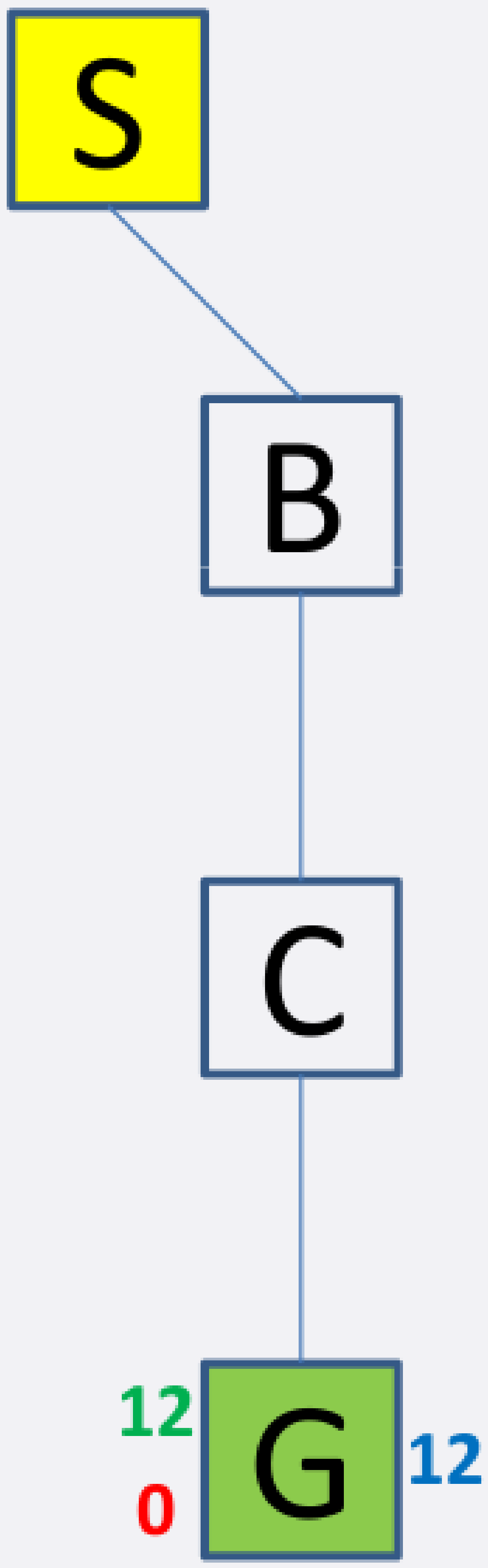


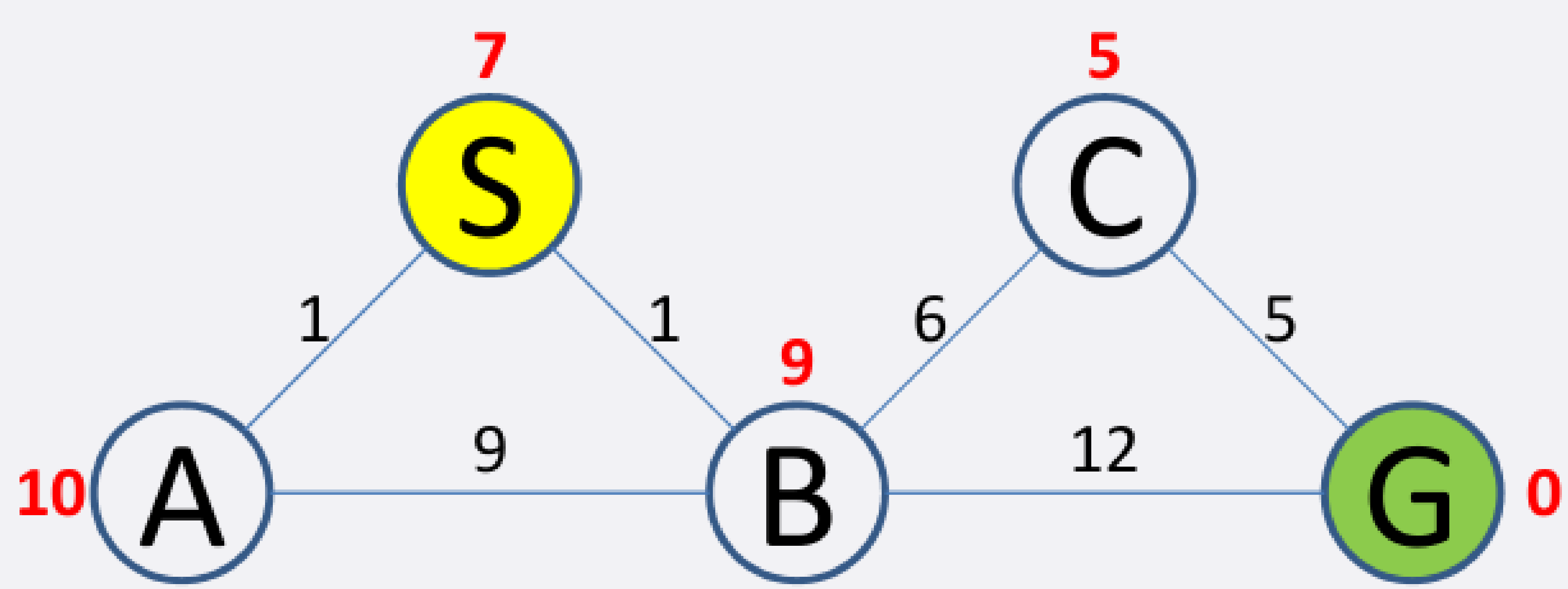
Exercise 3.12

f = accumulated path cost + heuristic

SUCCESS

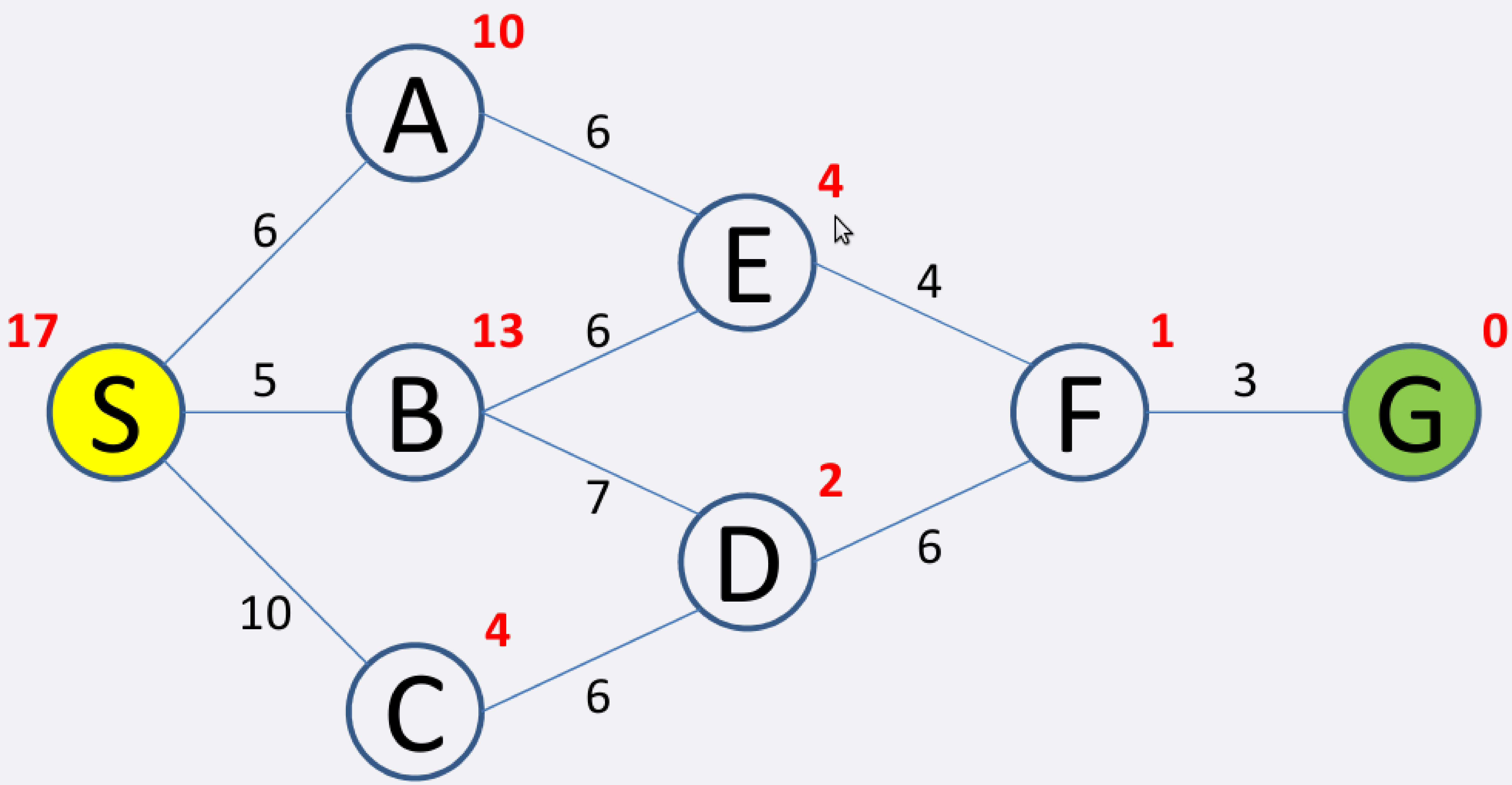
QUEUE = < SBCG>





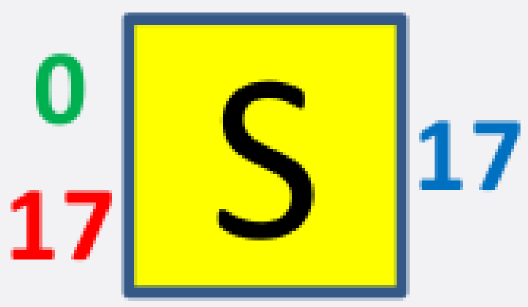
Exercise 3.13

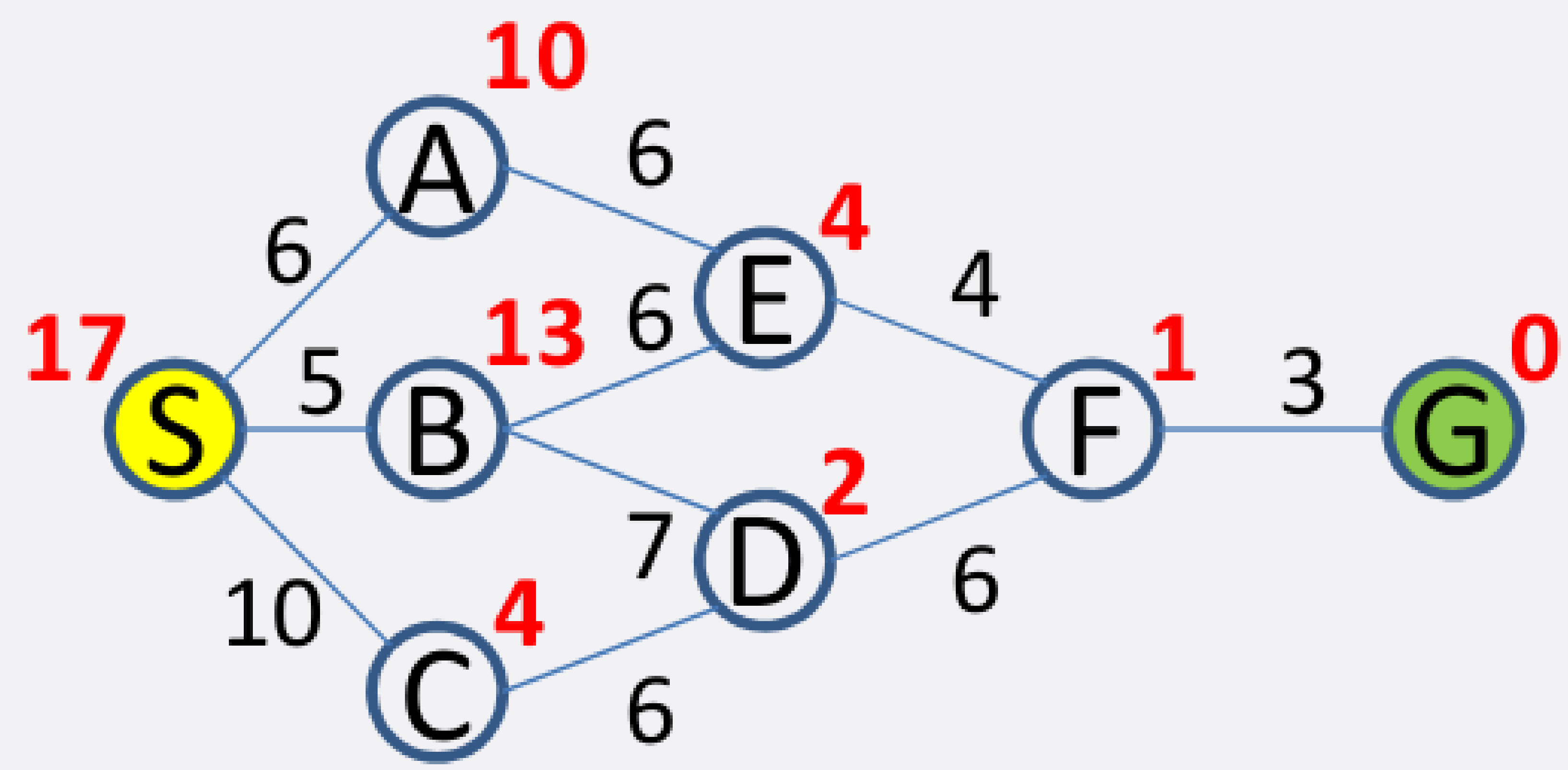
- Perform the A* Algorithm on the following figure. Explicitly write down the queue at each step.



Exercise 3.13

- Step 1



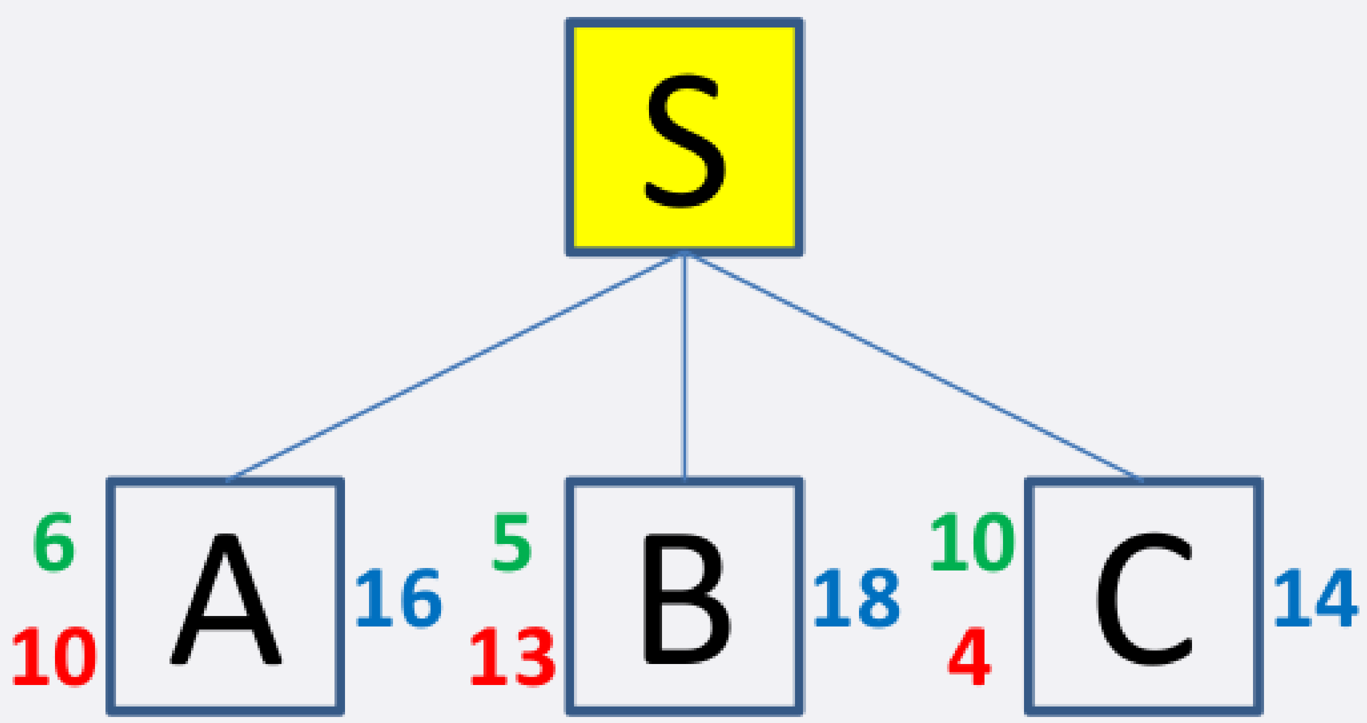


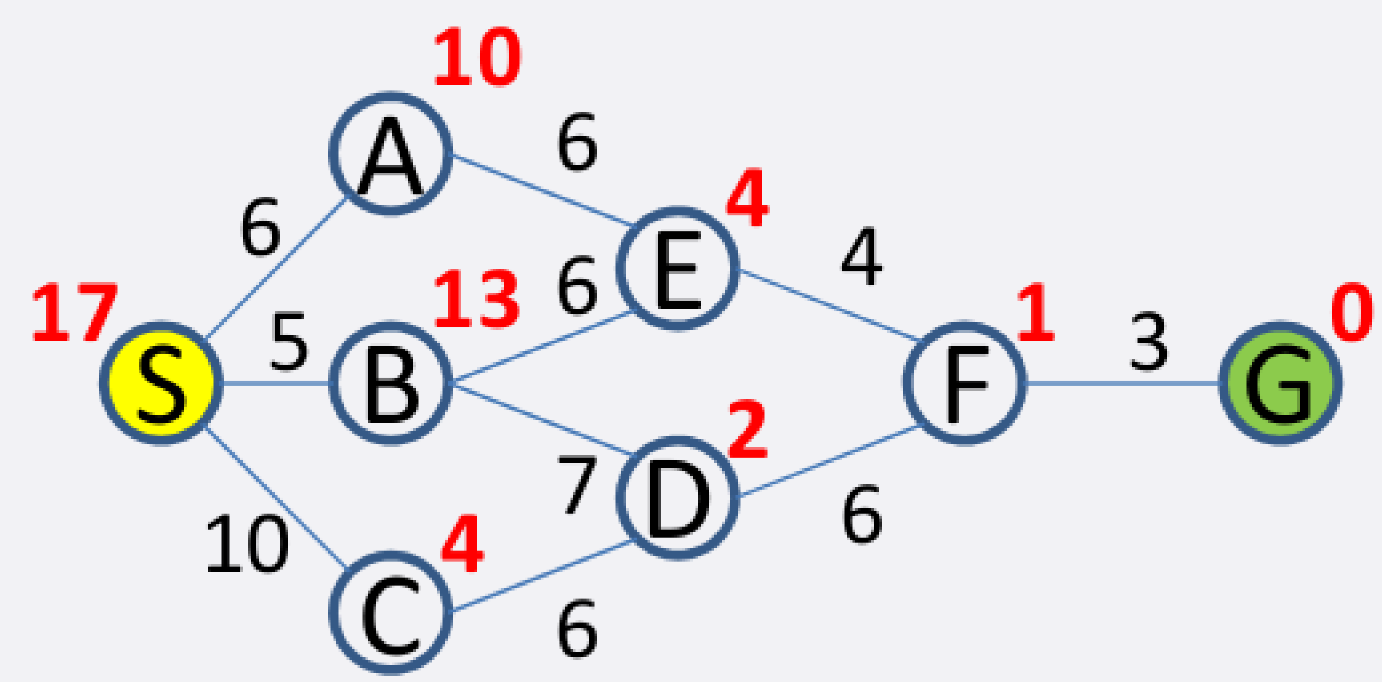
QUEUE:

S

Exercise 3.13

- Step 2





<u>QUEUE:</u>

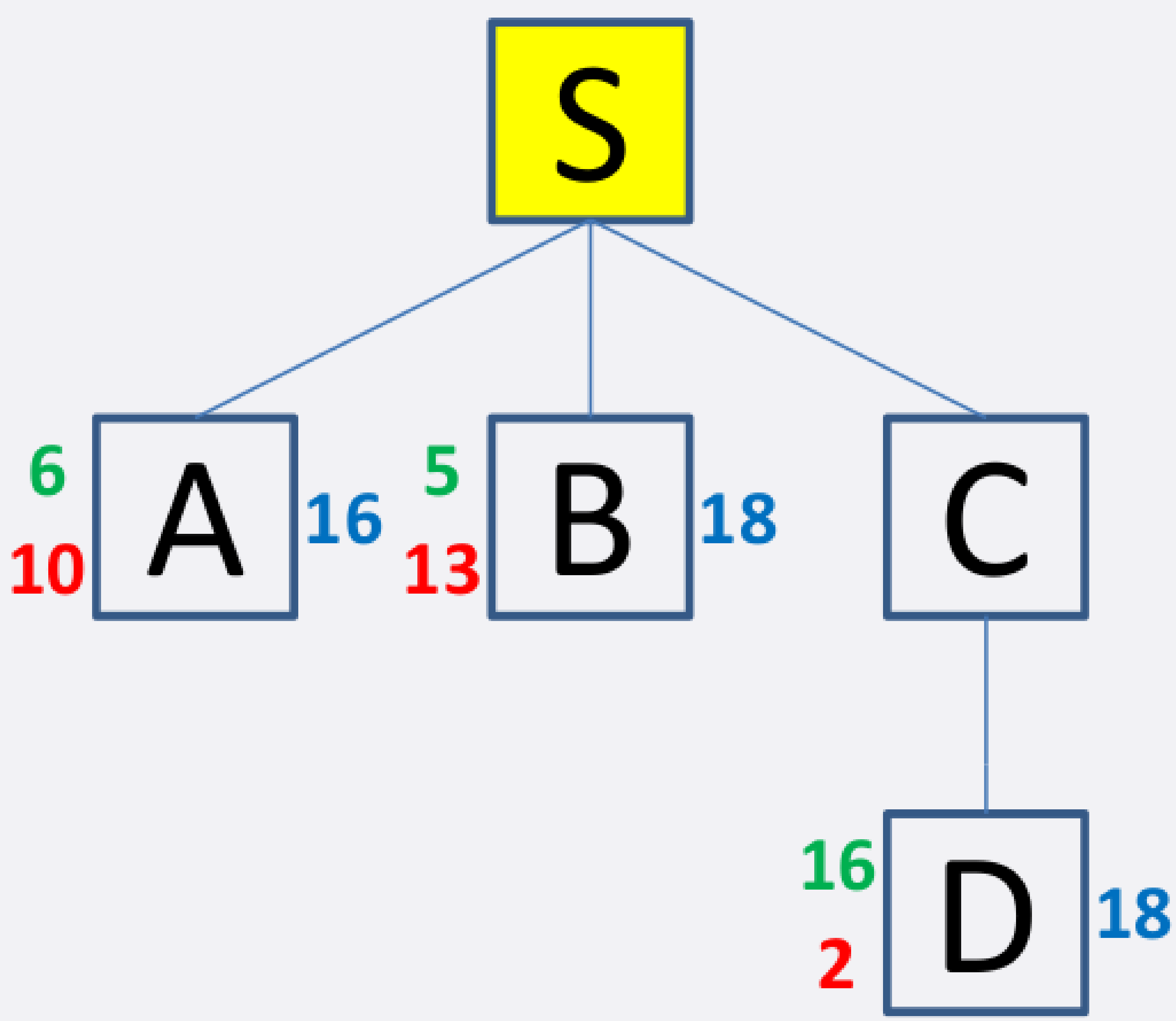
SC

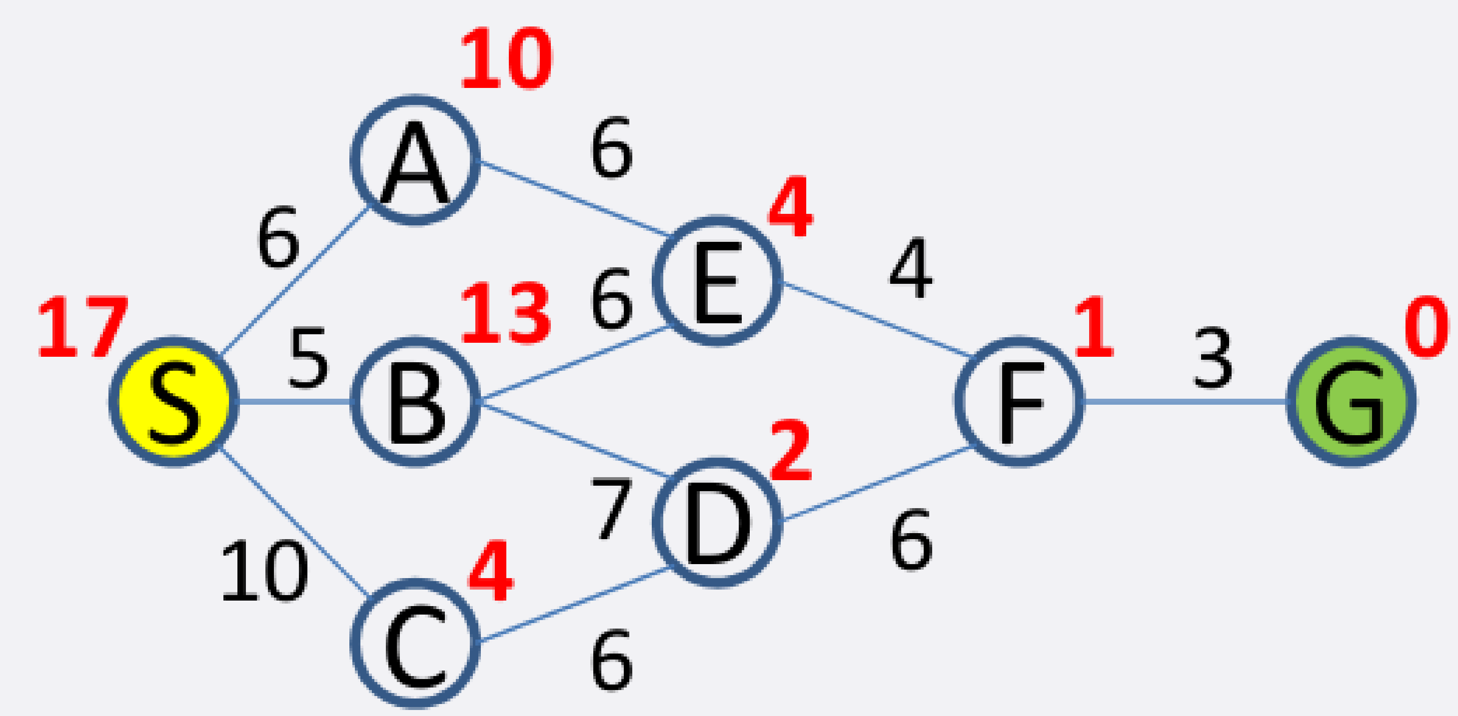
SA

SB

Exercise 3.13

- Step 3





QUEUE:

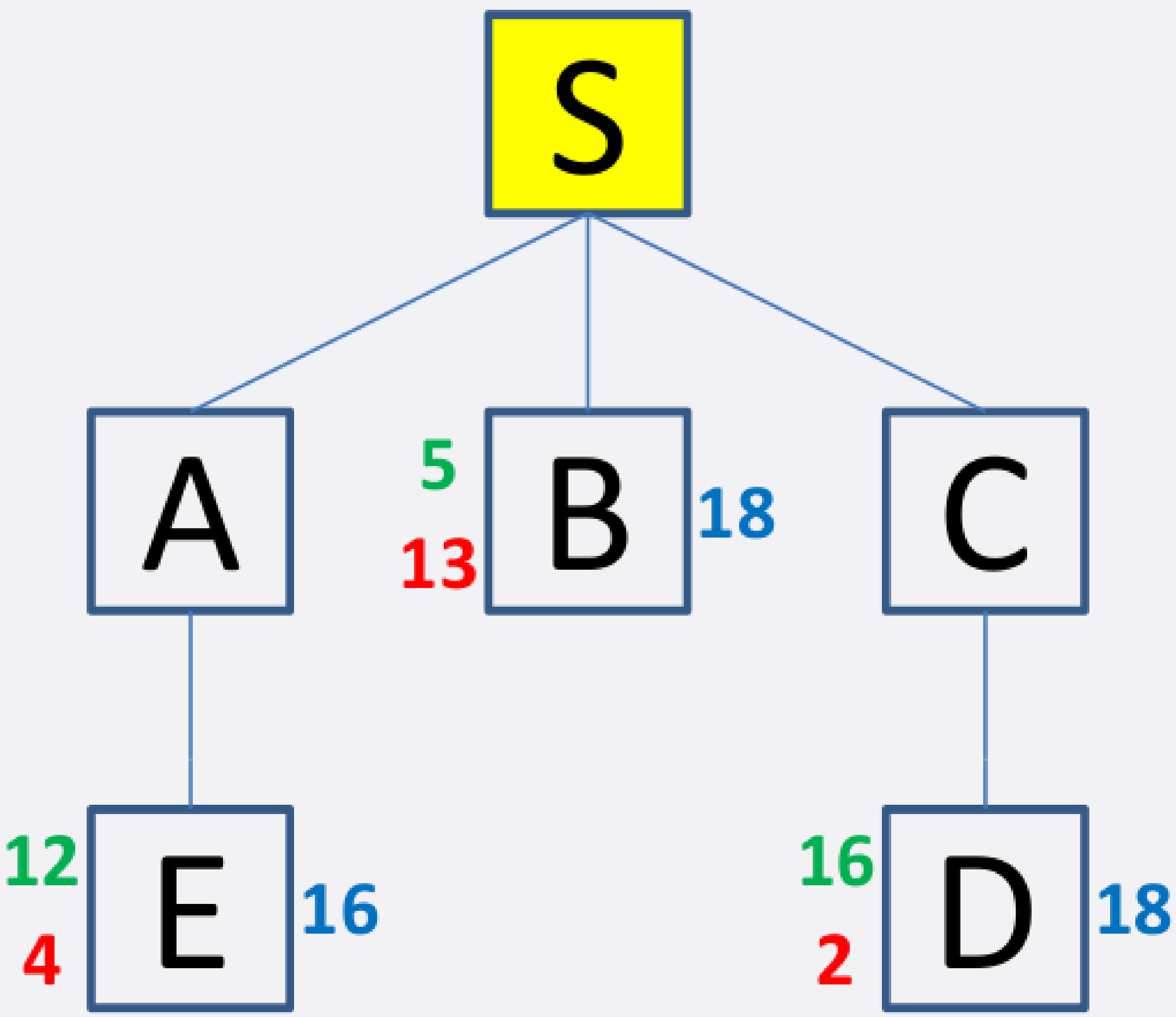
SA

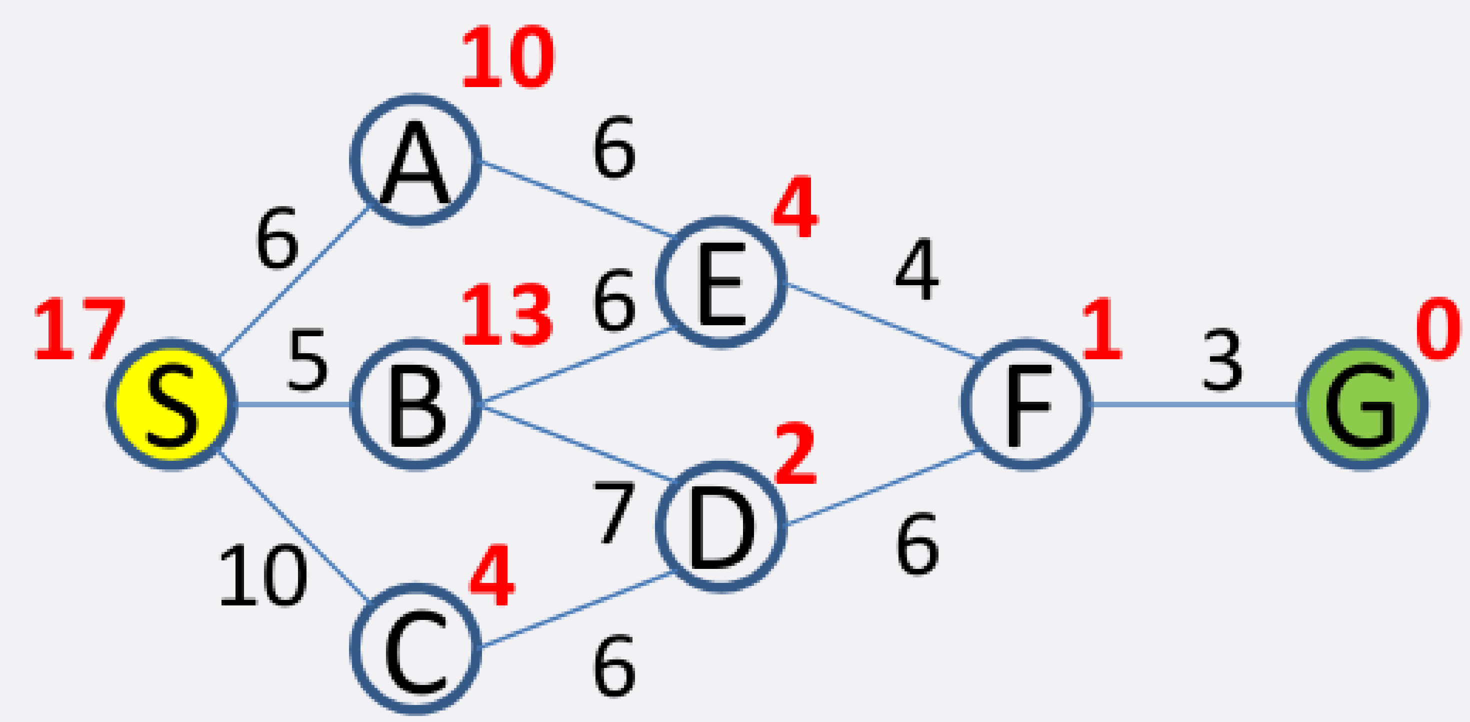
SCD

SB

Exercise 3.13

- Step 4





QUEUE:

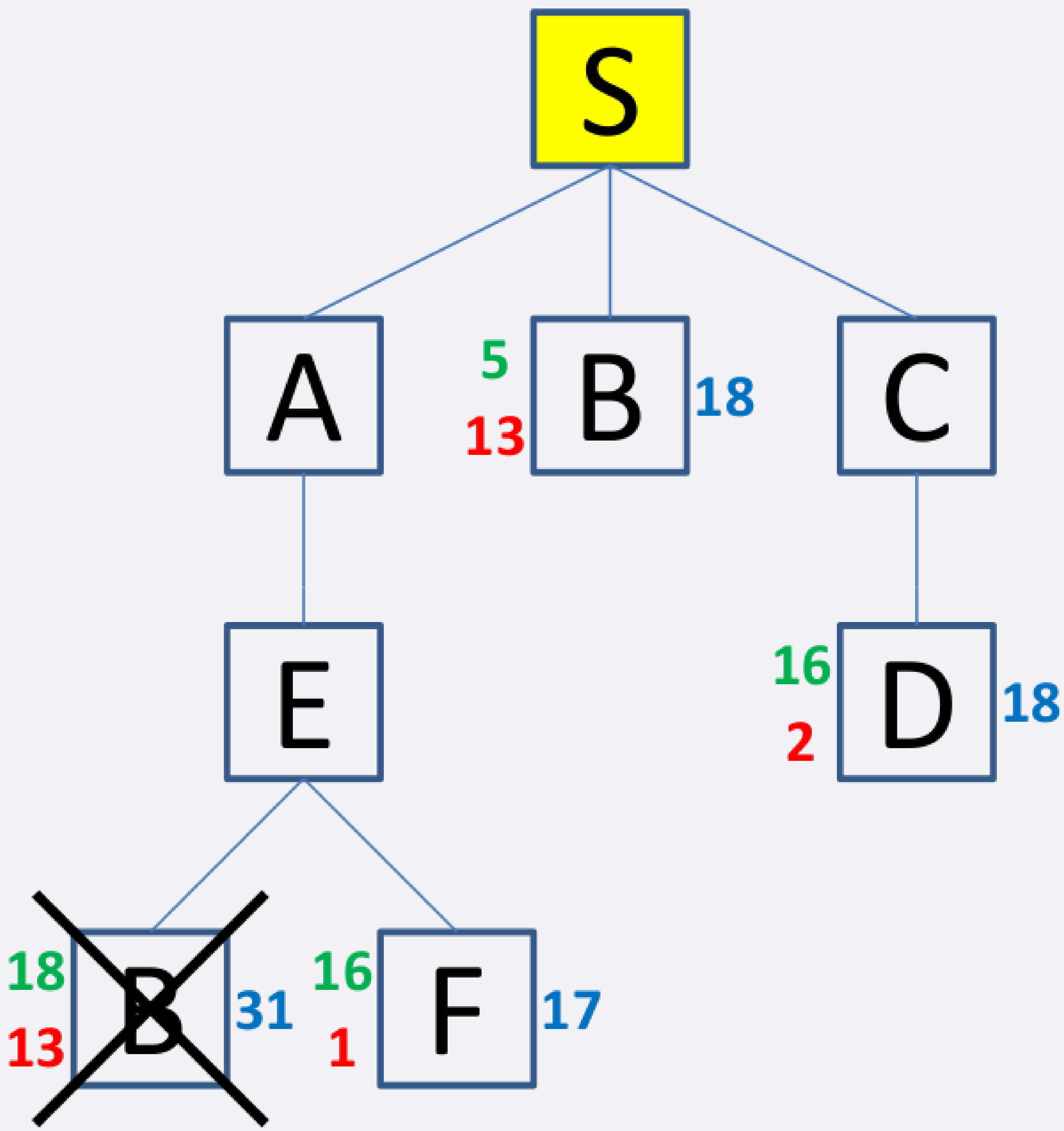
SAE

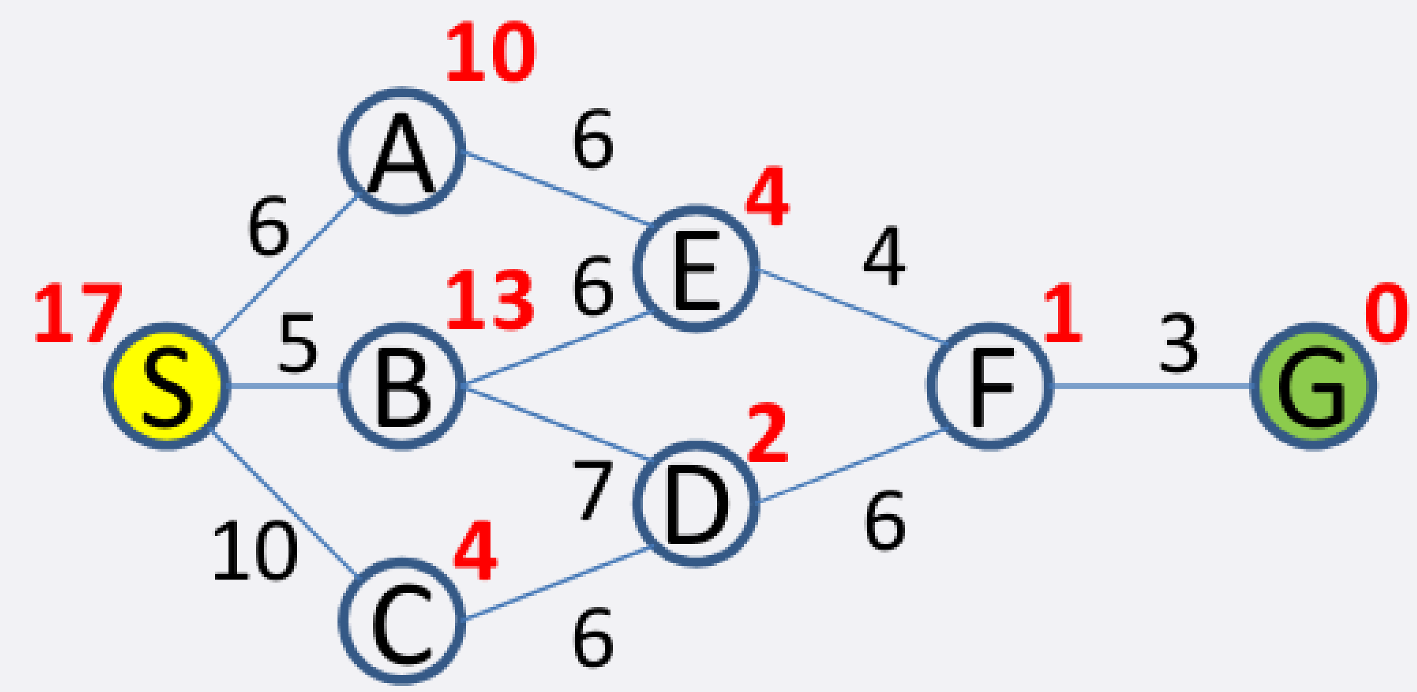
SCD

SB

Exercise 3.13

- Step 5





QUEUE:

SAEF

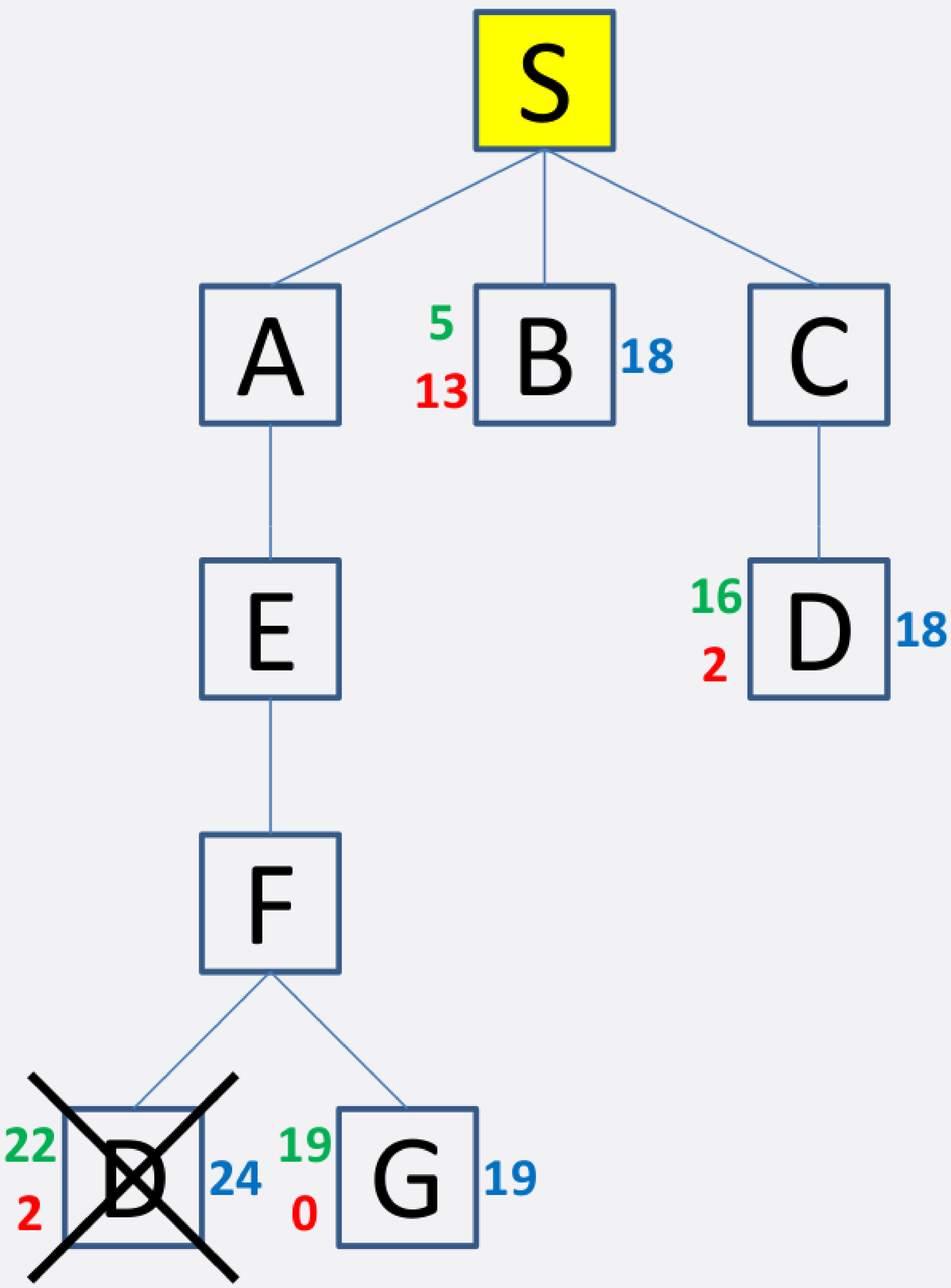
SCD

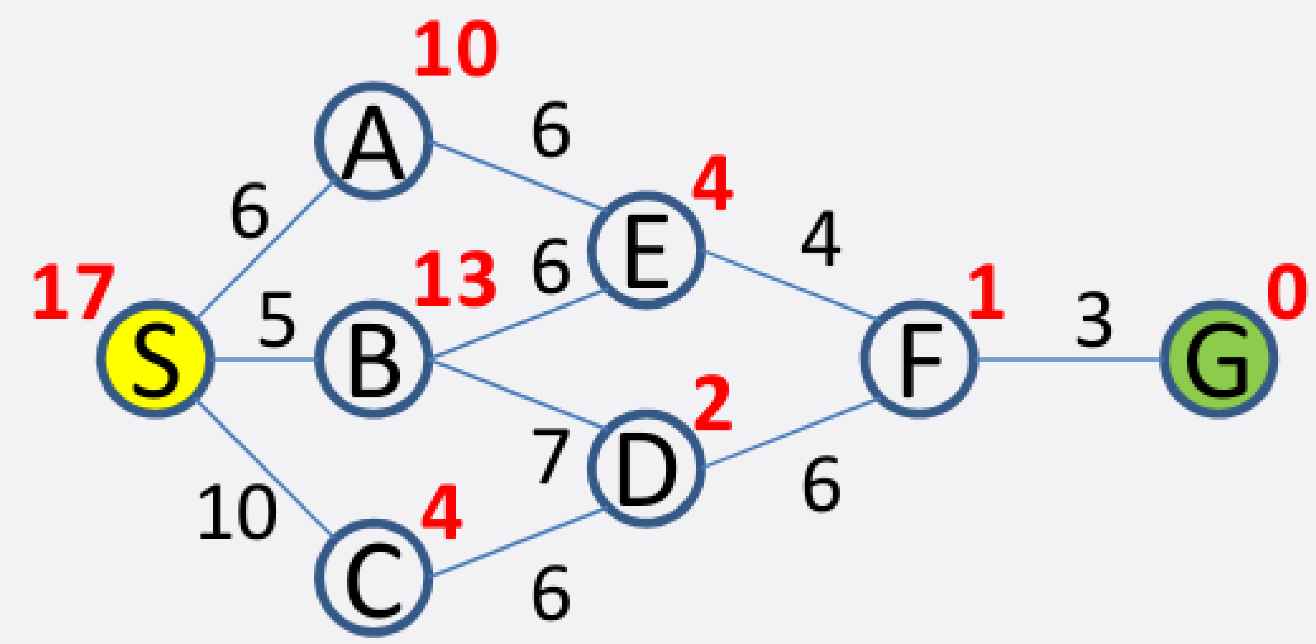
SB

SAEB

Exercise 3.13

- Step 6





QUEUE:

SCD

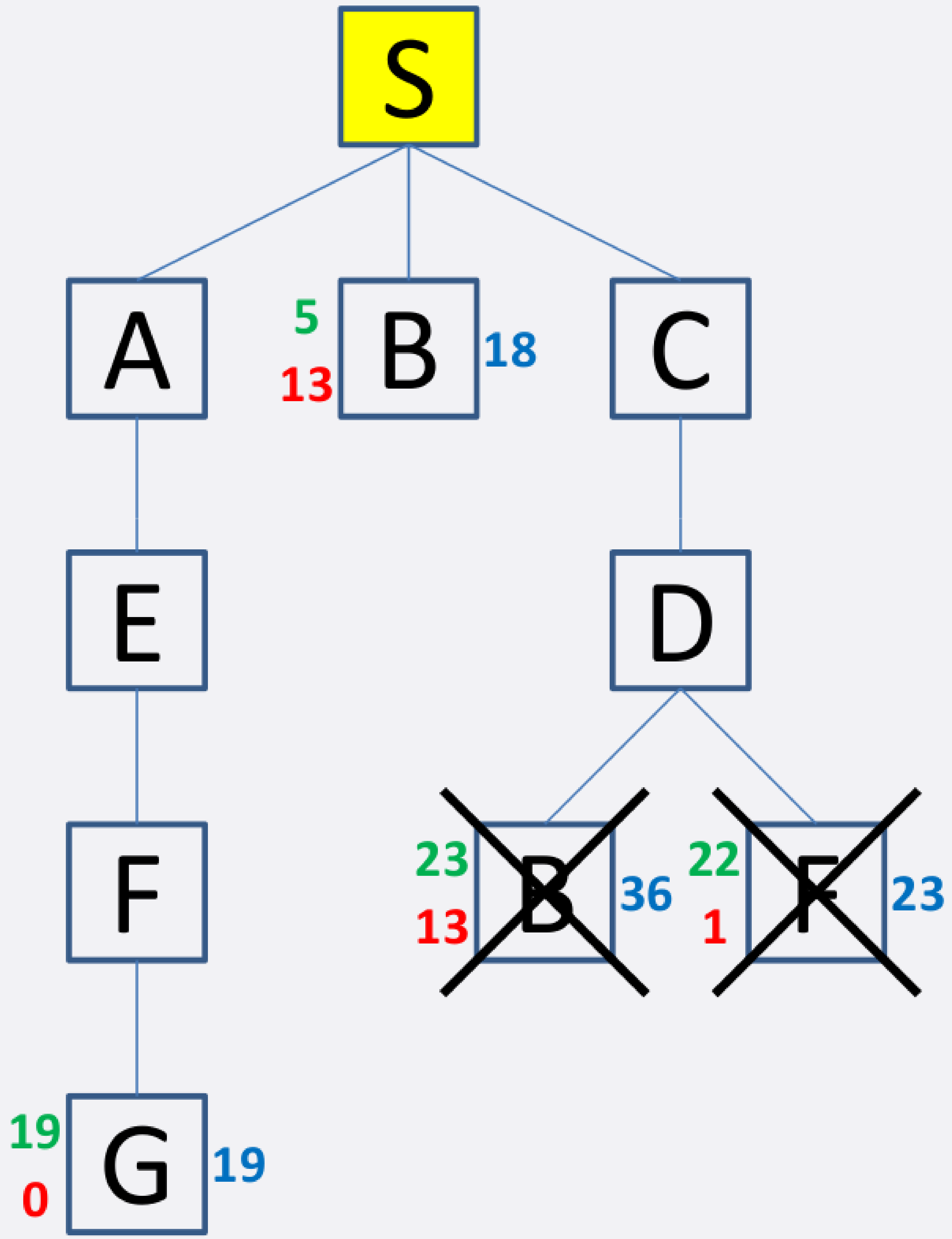
SB

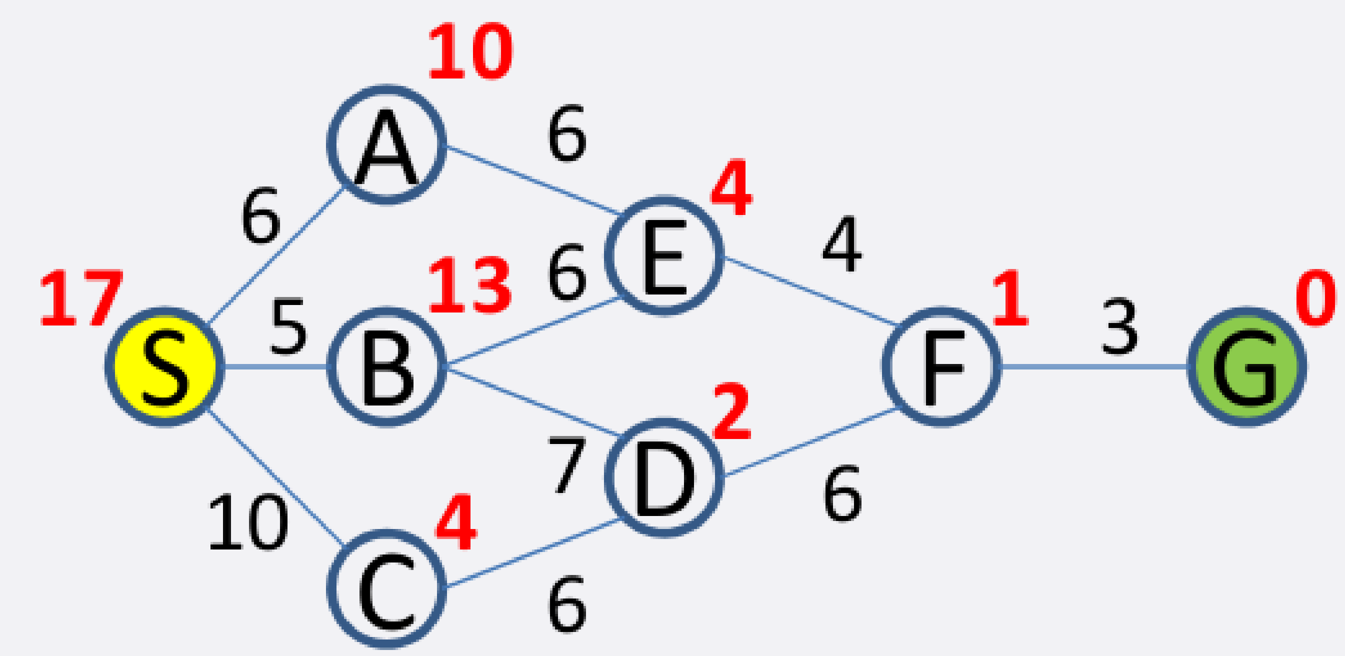
SAEFG

SAEFD

Exercise 3.13

- Step 7





<u>QUEUE:</u>

SB

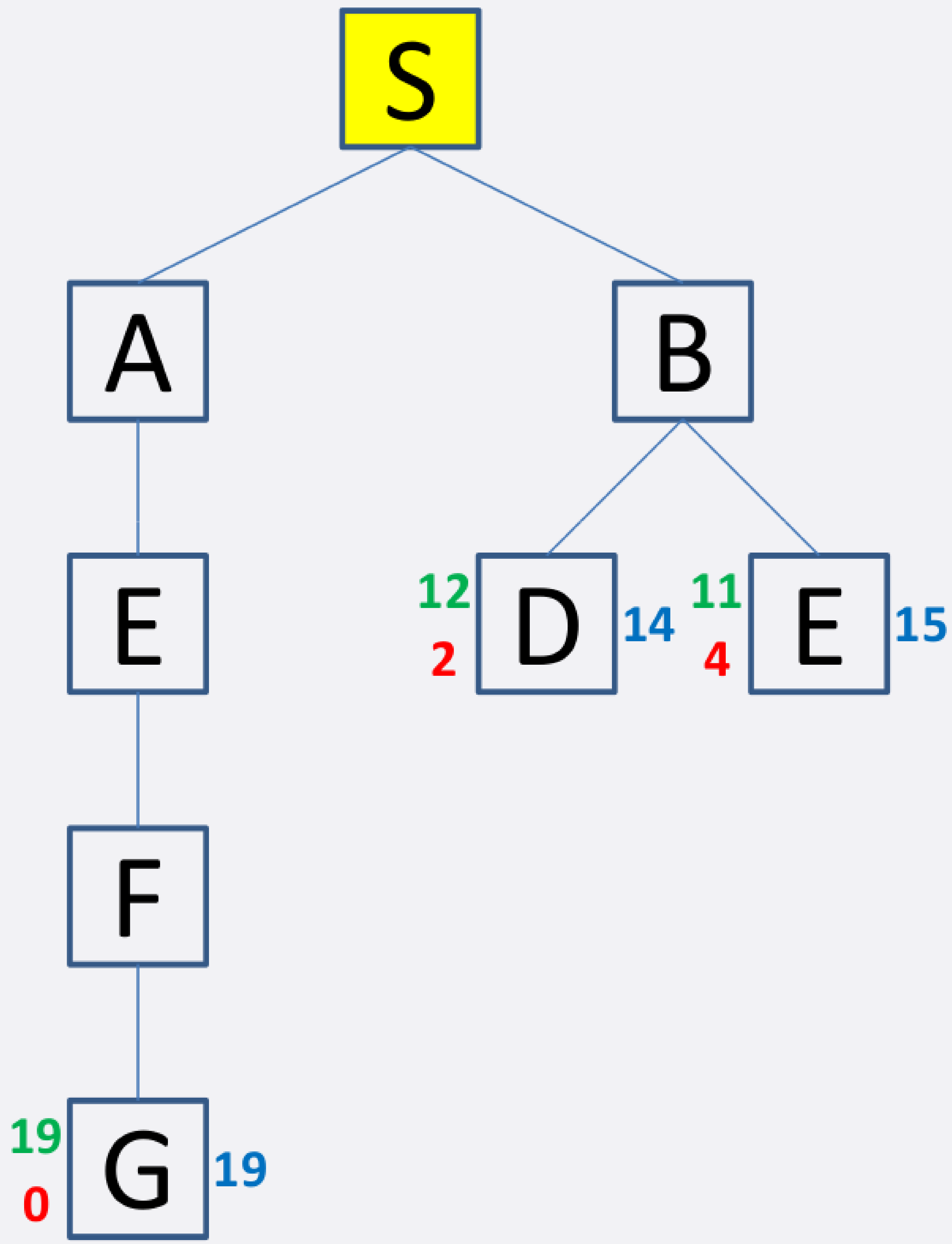
SAEFG

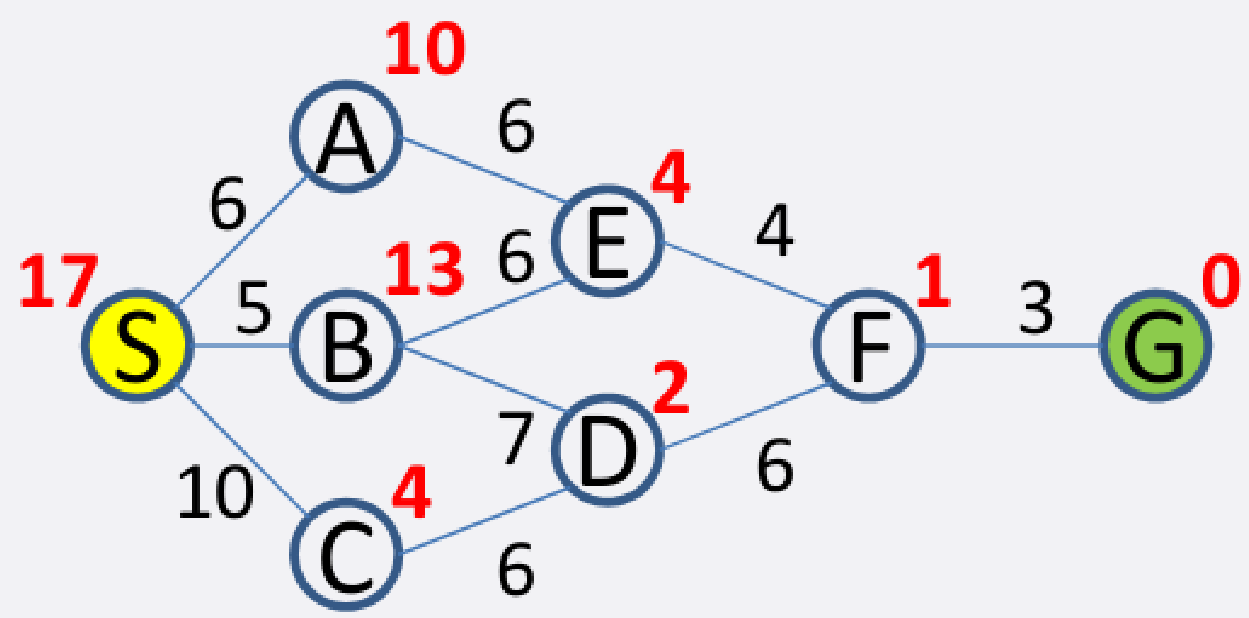
<u>SCDF</u>

<u>SCDB</u>

Exercise 3.13

- Step 8





QUEUE:

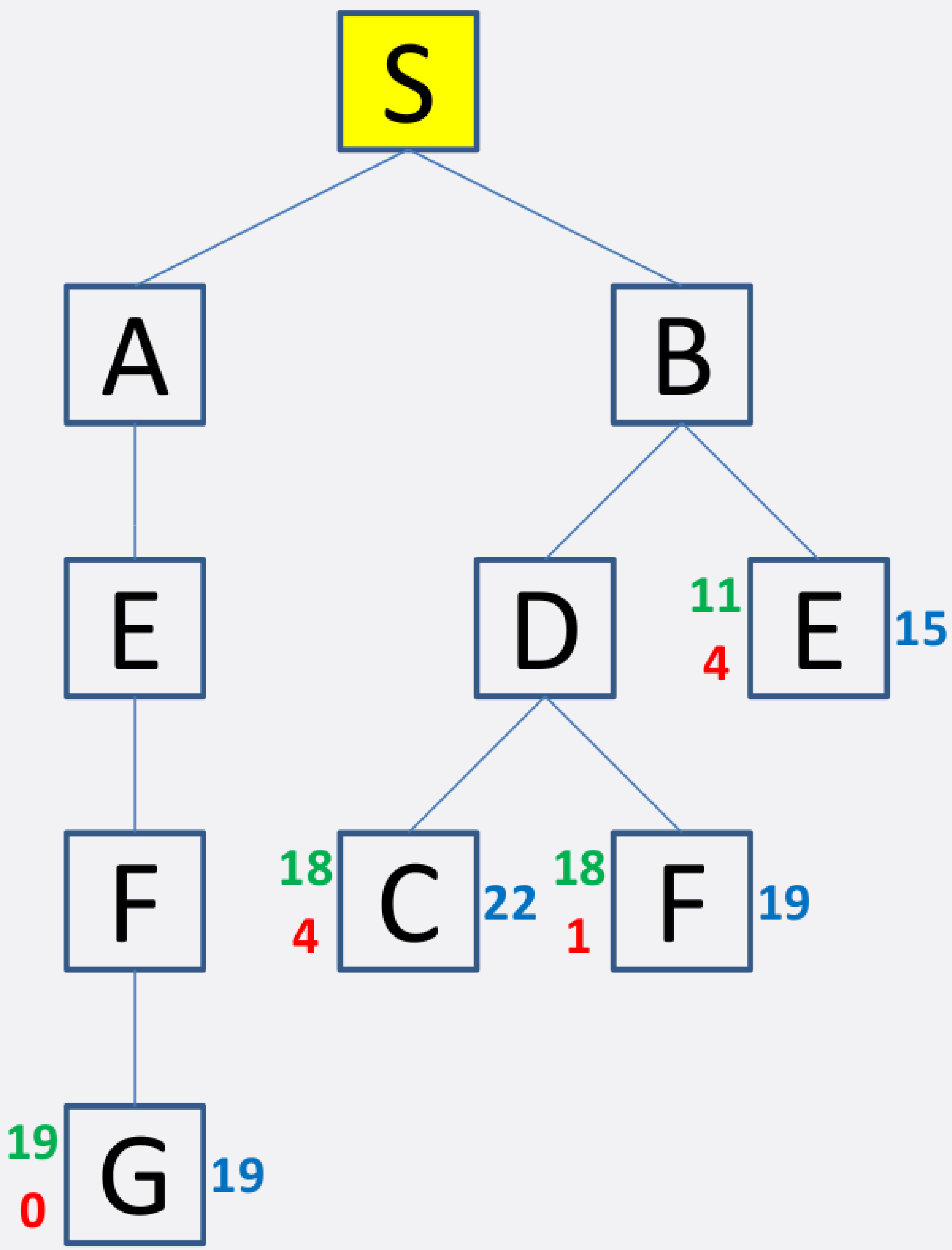
SBD

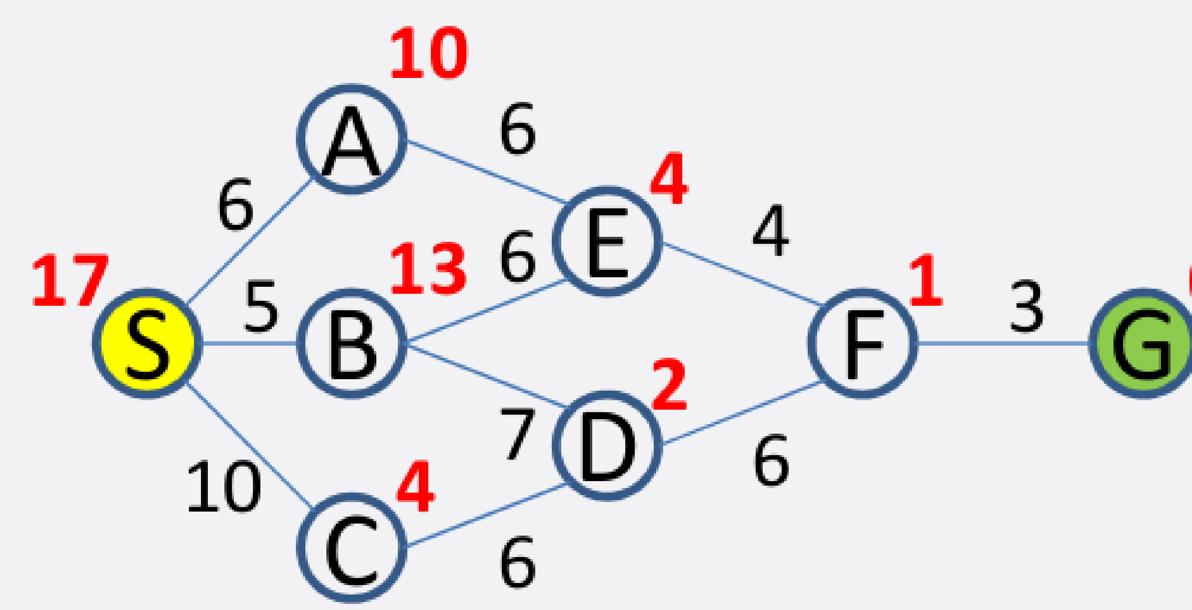
SBE

SAEFG

Exercise 3.13

- Step 9





QUEUE:

SBE

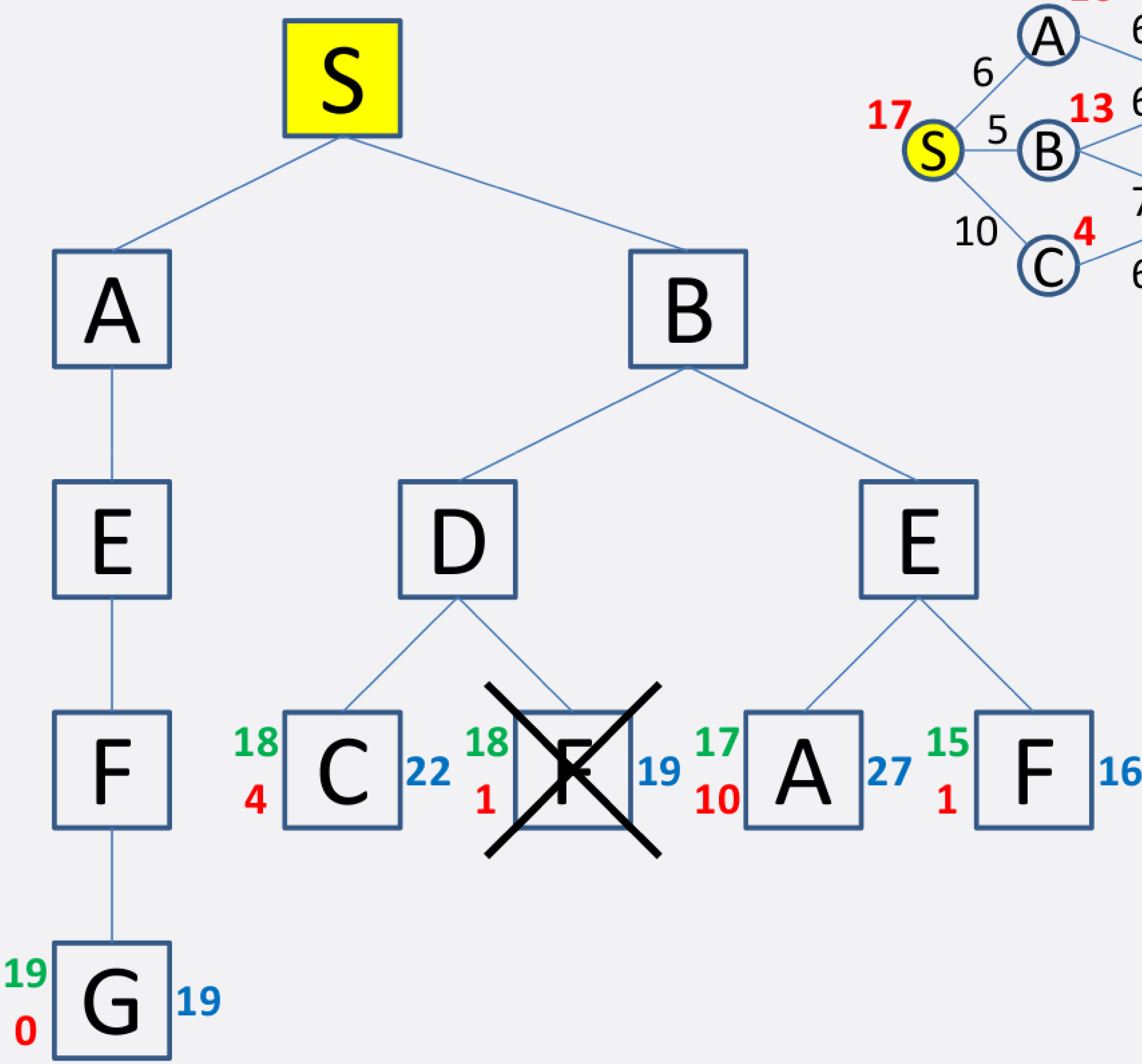
SBDF

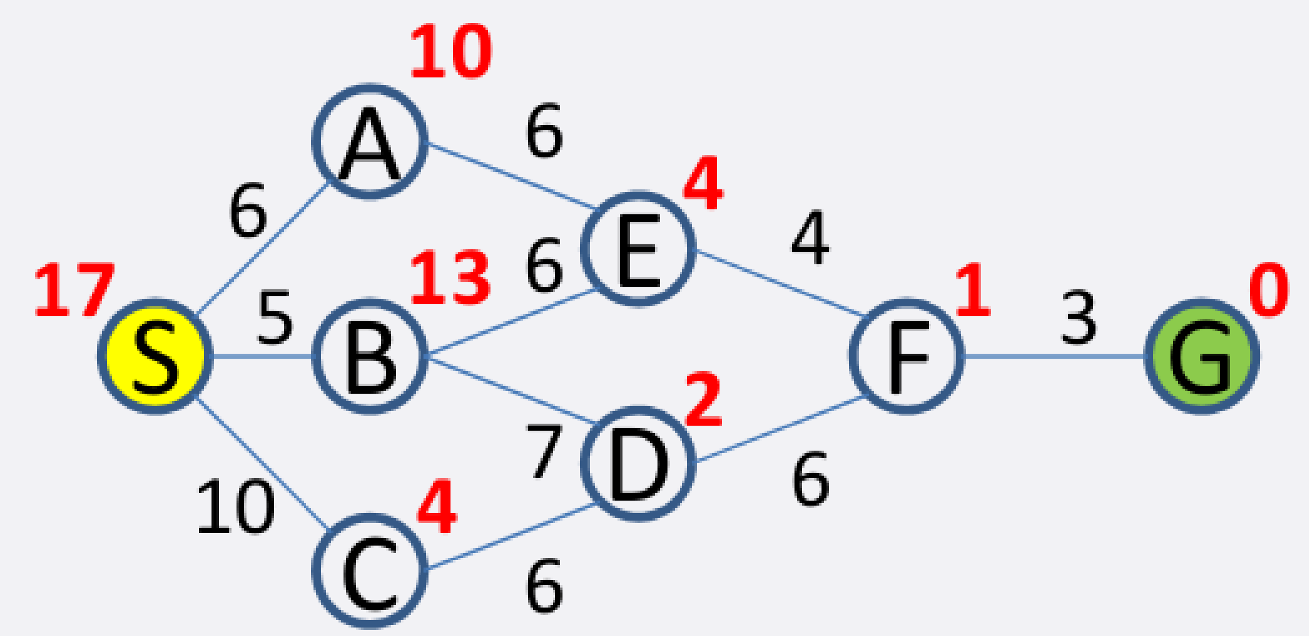
SAEFG

SBDC

Exercise 3.13

- Step 10





<u>QUEUE:</u>

SBEF

SAEFG

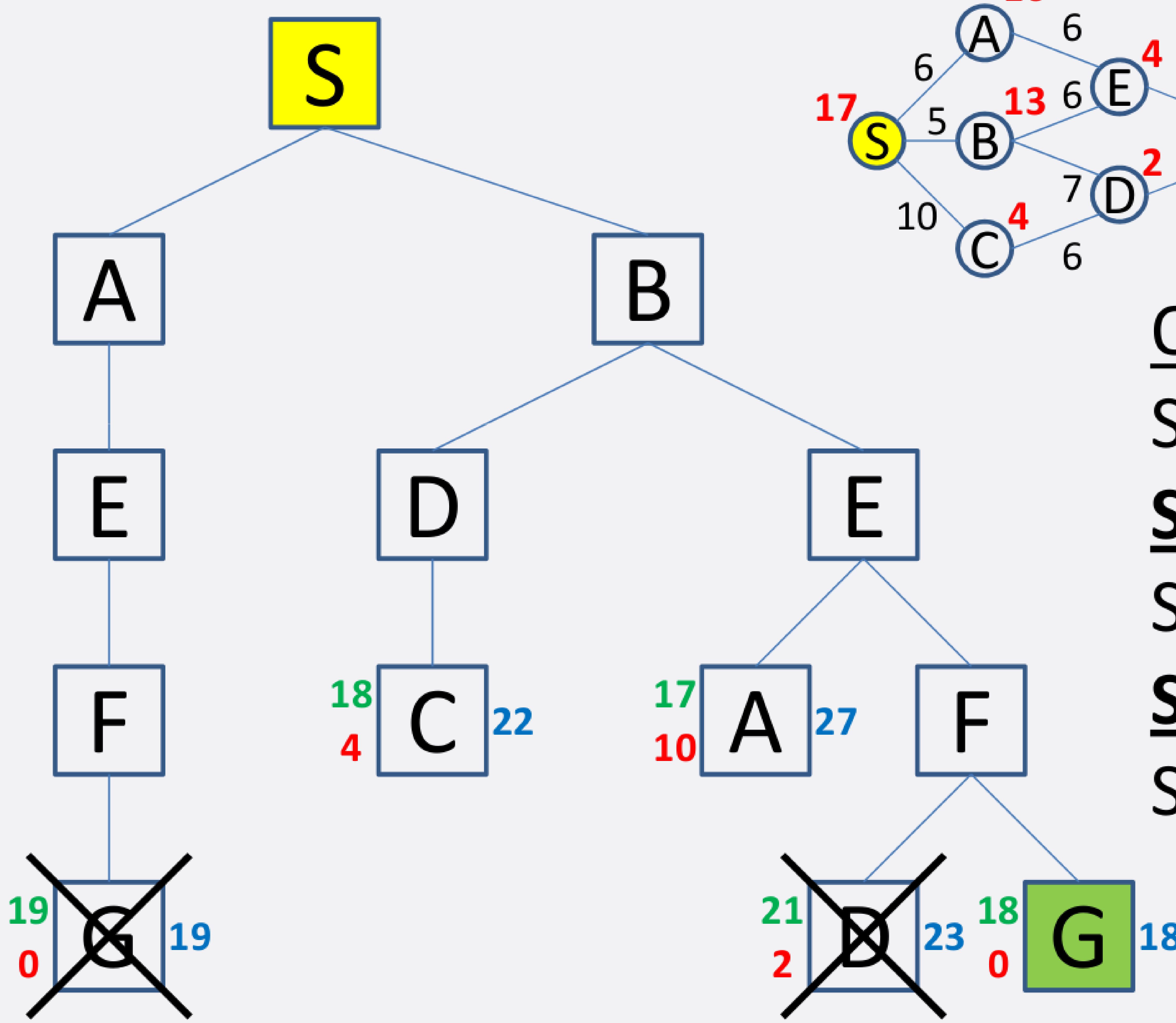
<u>**SBDF**</u>

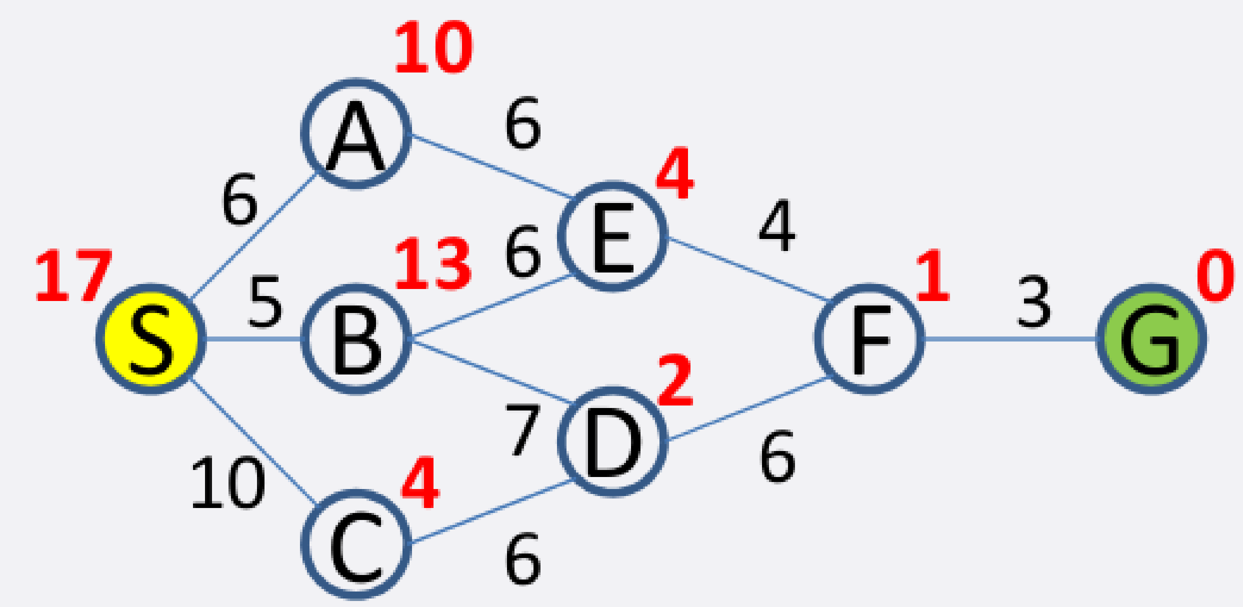
SBDC

SBEA

Exercise 3.13

- Step 11





QUEUE:

SBEFG

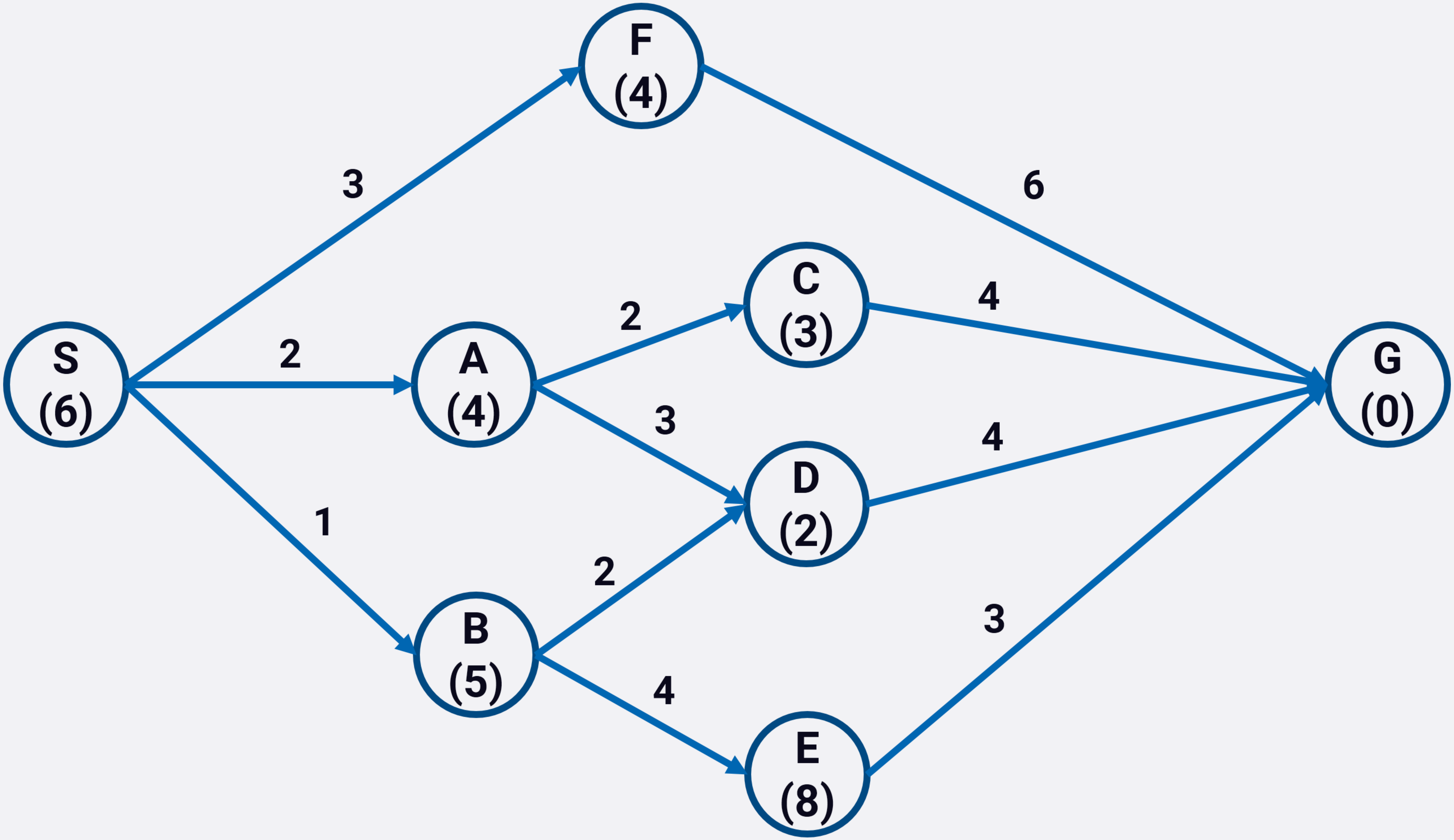
SAEFG

SBDC

SBEFD

SBEA

Exercise 3.14



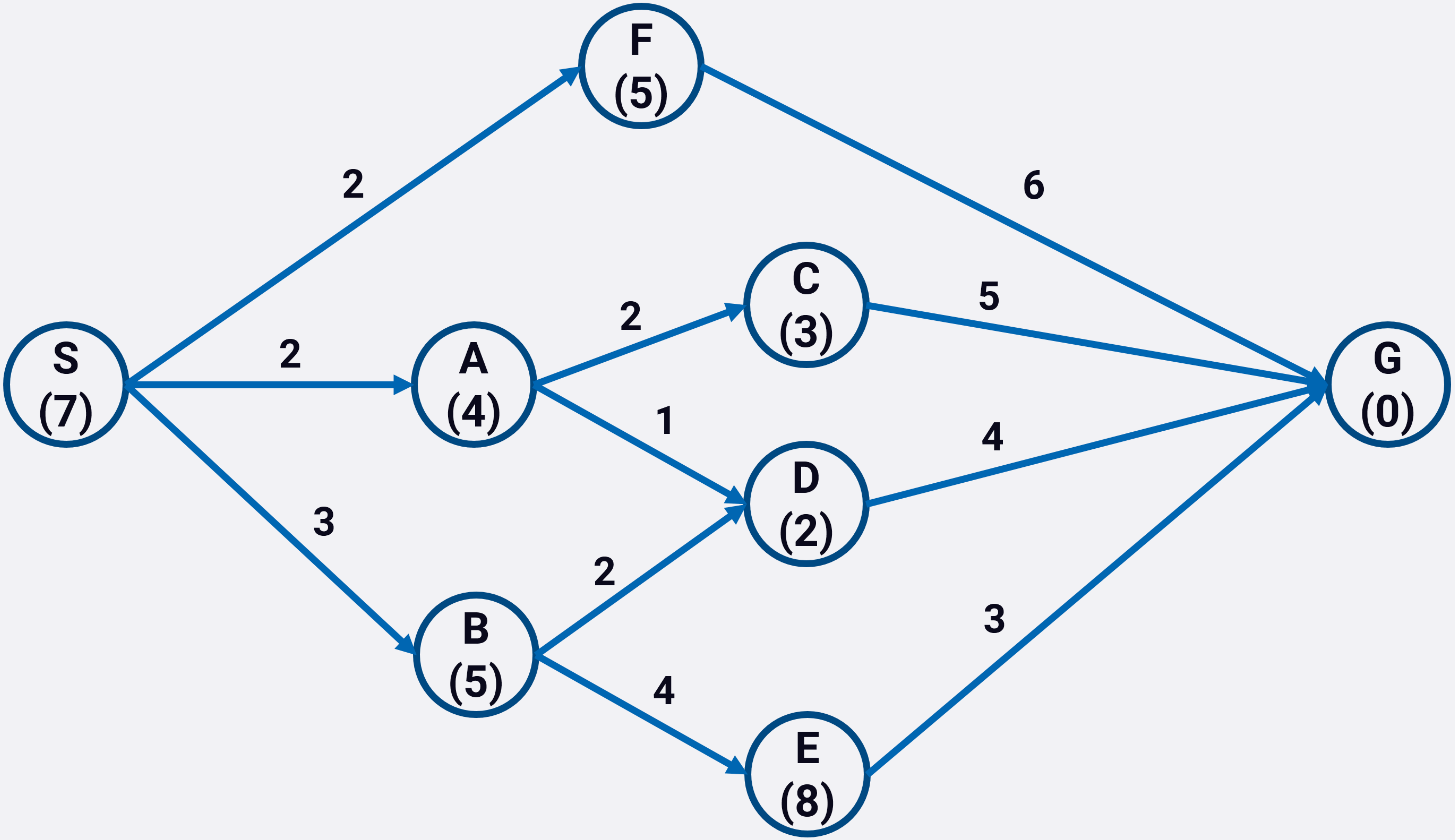
Exercise 3.14

Step	Queue	Processed Nodes	Children
1	S(6)	S(6)	A(2+4) B(1+5) F(3+4)
2	A(6) B(6) F(7)	A(6)	C(4+3) D(5+2)
3	B(6) C(7) D(7) F(7)	B(6)	D(3+2) E(5+8)
4	D(5) C(7) F(7) E(13)	D(5)	G(7+0)
5	C(7) F(7) G(7) E(13)	C(7)	G(8+0)
6	F(7) G(7) E(13)	F(7)	G(9+0)
7	G(7) E(13)	G(7)	

Node	h	h*
S	6	7
A	4	6
B	5	6
C	3	4
D	2	4
E	**8**	**3**
F	4	6

The **h** function is **not admissible** because for the **node E** the actual cost for reaching the goal is higher than the estimated one.

Exercise 3.15



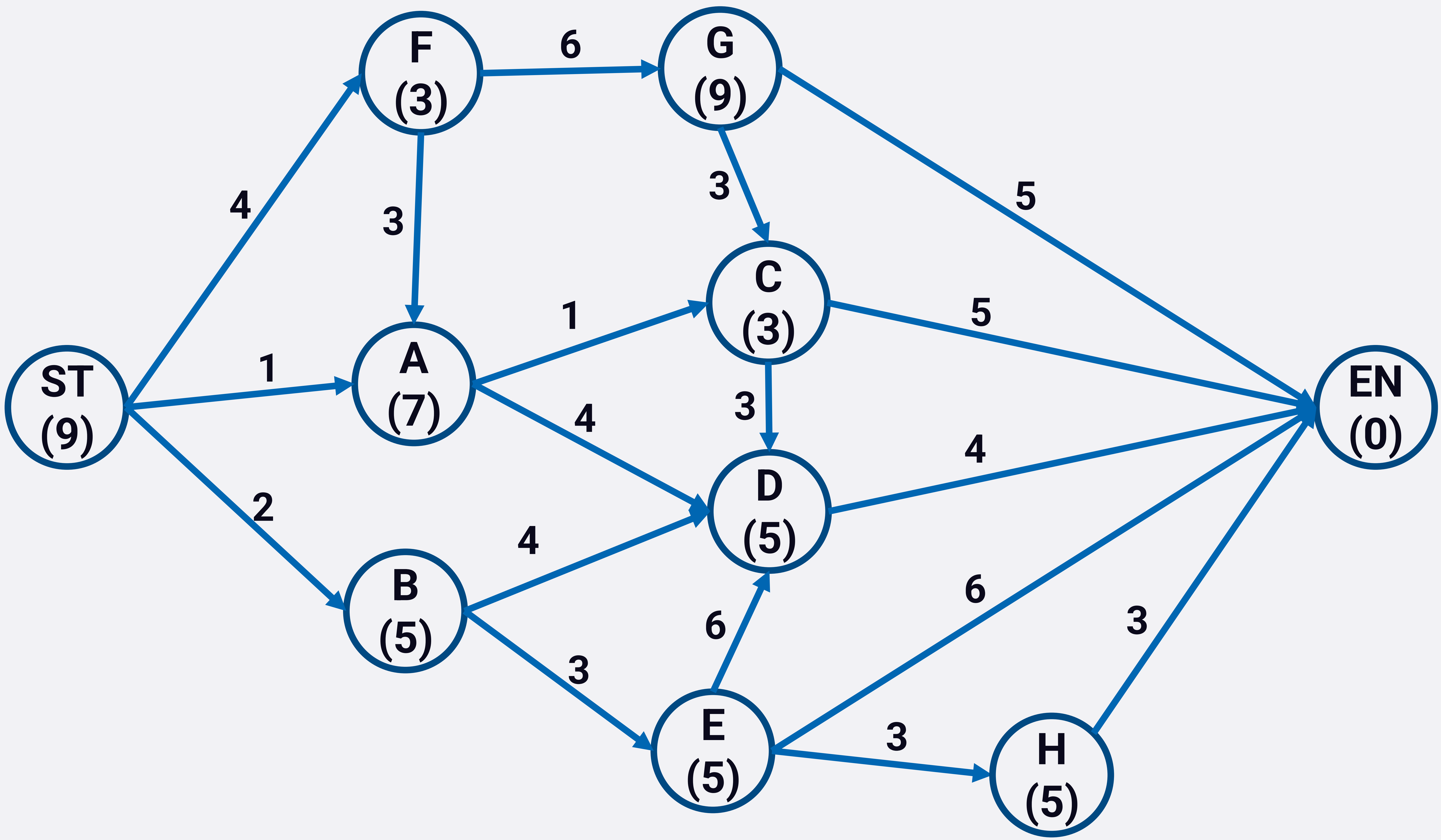
Exercise 3.15

Step	Queue	Processed Nodes	Children
1	S(7)	S(7)	A(2+4) B(3+5) F(2+5)
2	A(6) F(7) B(8)	A(6)	C(4+3) D(3+2)
3	D(5) C(7) F(7) B(8)	D(5)	G(7+0)
4	C(7) F(7) G(7) B(8)	C(7)	G(9+0)
5	F(7) G(7) B(8)	F(7)	G(8+0)
6	G(7) B(8)	G(7)	

Node	h	h*
S	7	7
A	4	5
B	5	6
C	3	5
D	2	4
E	**8**	**3**
F	5	6

The **h** function is **not admissible** because for the **node E** the actual cost for reaching the goal is higher than the estimated one.

Exercise 3.16



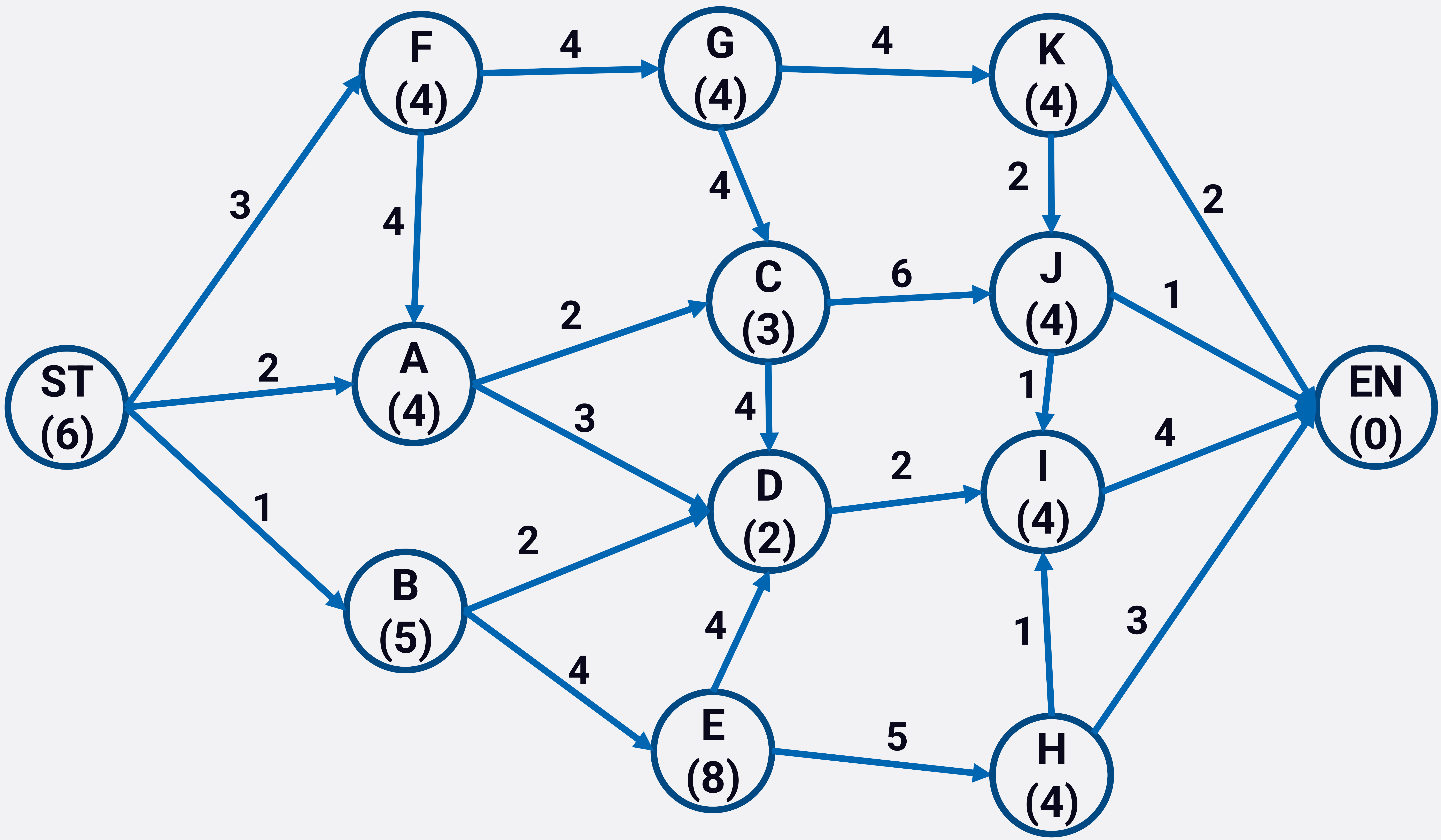
Exercise 3.16

Step	Queue	Processed Nodes	Children
1	ST(7)	ST(7)	A(1+7) B(2+5) F(4+3)
2	B(7) F(7) A(8)	B(7)	D(6+5) E(5+5)
3	F(7) A(8) E(10) D(11)	F(7)	A(7+7) G(10+9)
4	A(8) E(10) D(11) G(19)	A(8)	C(2+3) D(5+5)
5	C(5) D(10) E(10) G(19)	C(5)	EN(7+0) D(5+5)
6	EN(7) D(10) E(10) G(19)	EN(7)	

Node	h	h*
ST	**9**	**7**
A	**7**	**6**
B	5	8
C	3	5
D	**5**	**4**
E	5	6
F	3	9
G	**9**	**5**
H	**5**	**3**

The **h** function is **not admissible** because for the **nodes ST, A,D,G,** and **H** the estimated cost for reaching the goal is higher than the actual one.

Exercise 3.17



Exercise 3.17

Step	Queue	Processed Nodes	Children
1	ST(6)	ST(6)	A(2+4) B(1+5) F(3+4)
2	A(6) B(6) F(7)	A(6)	C(4+3) D(5+2)
3	B(6) C(7) D(7) F(7)	B(6)	D(3+2) E(5+8)
4	D(5) C(7) F(7) E(13)	D(5)	I(5+4)
5	C(7) F(7) I(9) E(13)	C(7)	D(8+2) J(10+4)
6	F(7) I(9) E(13) J(14)	F(7)	A(7+4) G(7+4)
7	I(9) G(11) E(13) J(14)	I(9)	EN(9+0)
8	EN(9) G(11) E(13) J(14)	EN(9)	

Node	h	h*
ST	6	9
A	4	7
B	5	8
C	3	7
D	2	6
E	8	8
F	4	10
G	4	6
H	**4**	**3**
I	4	4
J	**4**	**1**
K	**4**	**2**

The **h** function is **not admissible** because for the nodes **H, J, and K** the estimated cost for reaching the goal is higher than the actual one.