Ford-Fulkerson's Labeling Algorithm

Luke Zhou

December 10, 2018

Ford-Fulkerson's Method

FORD-FULKERSON-METHOD(G, s, t)

1 initialize flow f to 0
2 **while** there exists an augmenting path p in the residual network G_f
3 augment flow f along p
4 **return** f

Ford-Fulkerson's Method

FORD-FULKERSON-METHOD(G, s, t)

1 initialize flow f to 0
2 **while** there exists an augmenting path p in the residual network G_f
3 augment flow f along p
4 **return** f

"We call it a 'method' rather than an 'algorithm' because it encompasses several implementations with differing running times."

The Ford-Fulkerson Labeling Algorithm: Convention

First, a convention:

- The algorithm begins with a linear order on the vertex set which establishes a notion of **precedence**.
- Typically, the first vertex in this linear order is the *source* while the second is the *sink*.
- After that, the vertices can be listed in any order.
- We will use the following convention: the vertices will be labeled with capital letters of the English alphabet and the linear order will be $(S, T, A, B, C, D, E, G, ...)$, which we will refer to as the **pseudo-alphabetic** order

Three States

In carrying out the labeling algorithm, vertices will be classified as:

- labeled
  - scanned
  - unscanned
- unlabeled

The form of a label (a triple):

$$\bigstar(VERTEX, SIGN(\pm), POTENTIAL)$$

where *VERTEX* is the ID of a vertex, *SIGN* is $+$ or $-$, and *POTENTIAL* is a positive real number.

Two Routines

Routine A

(**General Step**) Select a node, x, that is labeled and unscanned.

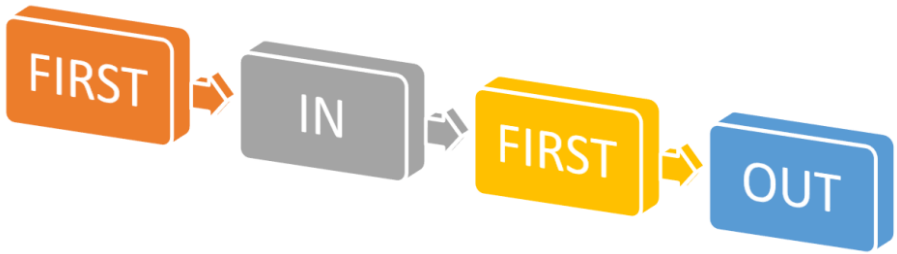
- To all unlabeled successor nodes, y, such that $f(x,y) < c(x,y)$, assign the label $(x, +, p(y))$ where $p(y) = \min\{p(x), c(x,y) - f(x,y)\}$.
- To all unlabeled predecessor nodes, y, such that $f(y,x) > 0$, assign the label $(x, -, p(y))$ where $p(y) = \min\{p(x), f(y,x)\}$.

Repeat the general step until

- the sink is labeled and unscanned: go to Routine B;
- no more labels can be assigned: terminate.

Two Rules

- **First Labeled, First Scanned**: the important rule is that we scan vertices in the order that they are labeleduntil we label the sink. (This aspect of the algorithm results in a **breadth-first search** of the vertices looking for ways to label previously unlabeled vertices.)



- **Never Relabel a Vertex**: Once a vertex is labeled, we do not change its label.

Two Routines

Routine B

(**Flow Updating**) The sink has been labeled $(y, \pm, p(t))$.

- If the second part of the label is $+$; replace $f(y,t)$ with $f(y,t) + p(t)$;
- otherwise, replace $f(t,y)$ with $f(t,y) - p(t)$.

Go to node y, and treat it the same way. Repeat until the source is reached. Then discard all labels and return to Routine A.

What Happens When the Sink is Labeled?

The labeling algorithm halts if the sink is ever labeled.

Now suppose that the sink is labeled with the triple $(u, +, a)$. We claim that we can find an augmenting path p which results in an increased flow with $\delta = a$, the potential on the sink.

To see this, we merely back-track. The sink T got its label from $u = u_1$, u_1 got its label from $u2$, and so forth. Eventually, we discover a vertex u_m which got its label from the source. The augmenting path is then $p = (S, u_m, u_{m-1}, ..., u_1, T)$.

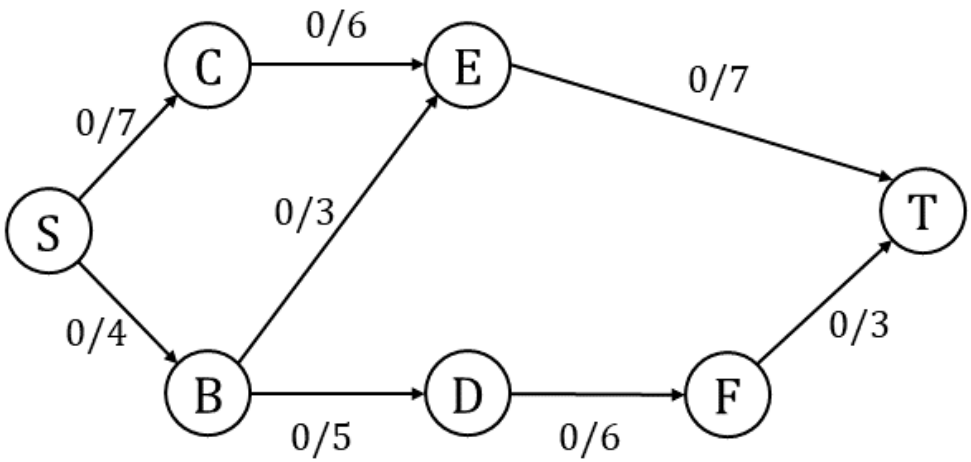
The value of δ for this path is the potential $p(T)$ on the sink since we've carefully ensured that
$p(u_m) \geq p(u_{m-1}) \geq ... \geq p(u_1) \geq p(T)$.

Termination

This process will be repeated until we reach a point where the labeling **halts** with **some vertices labeled** (one of these is the source) and **some vertices unlabeled** (one of these is the sink). We will then note that the partition $V = L \cup U$ into labeled and unlabeled vertices is a cut whose capacity is exactly equal to the value of the current flow.

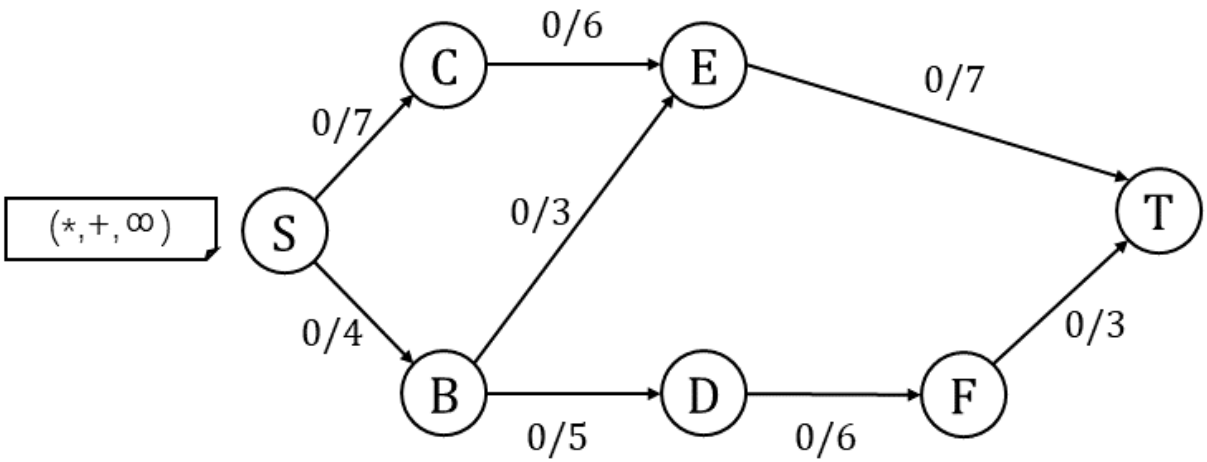
Example

EXAMPLE



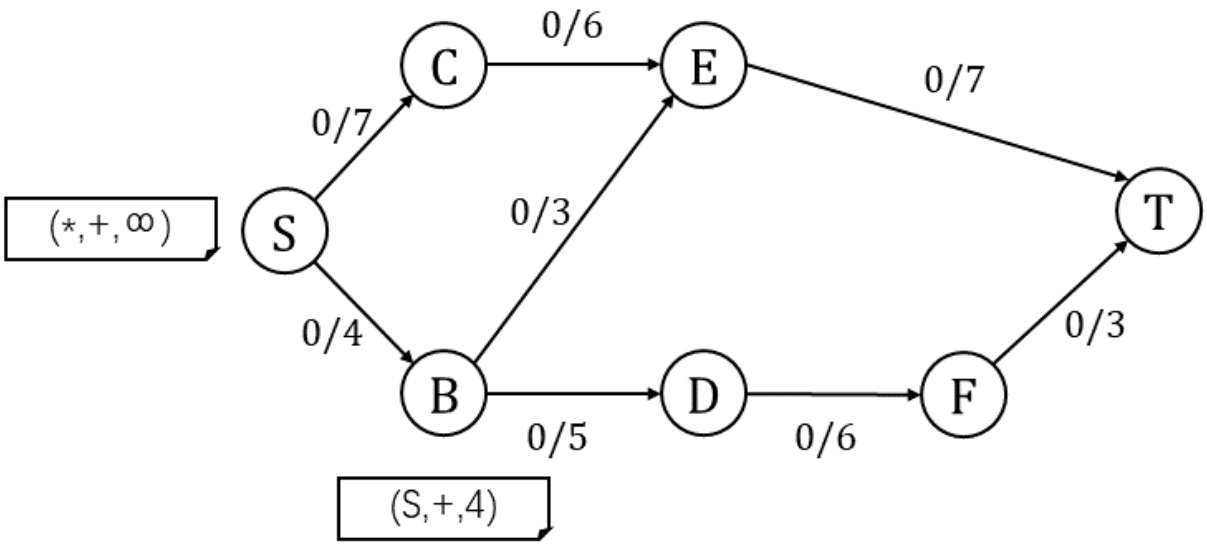
Example

INIT



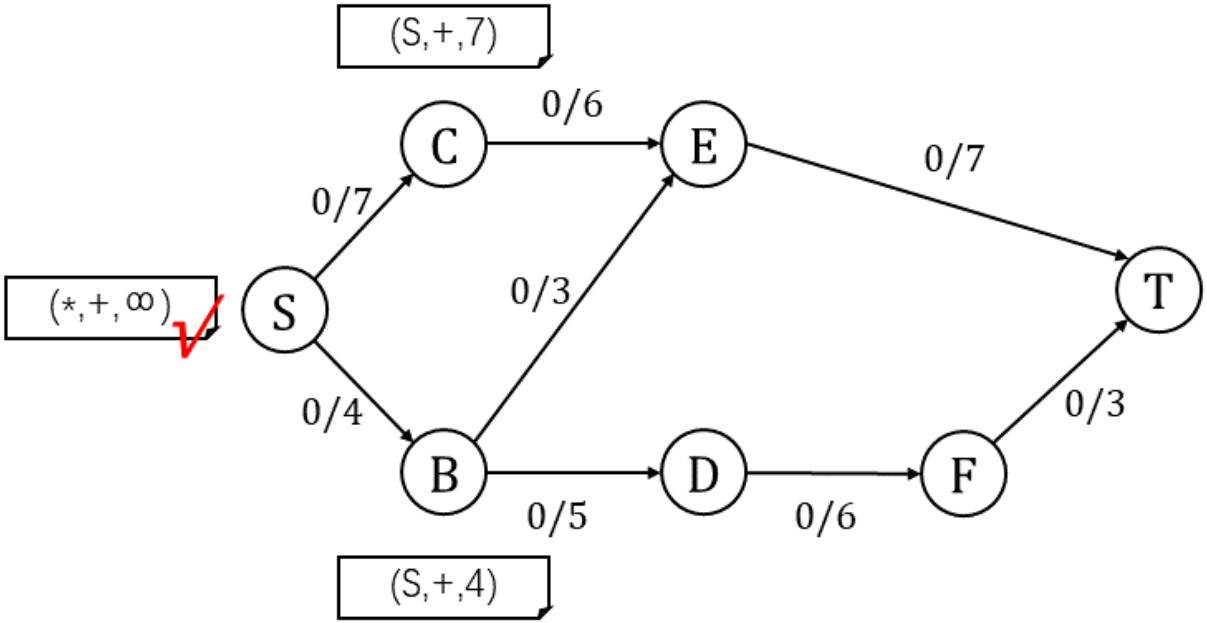
Example

ROUTINE A, ROUND 1



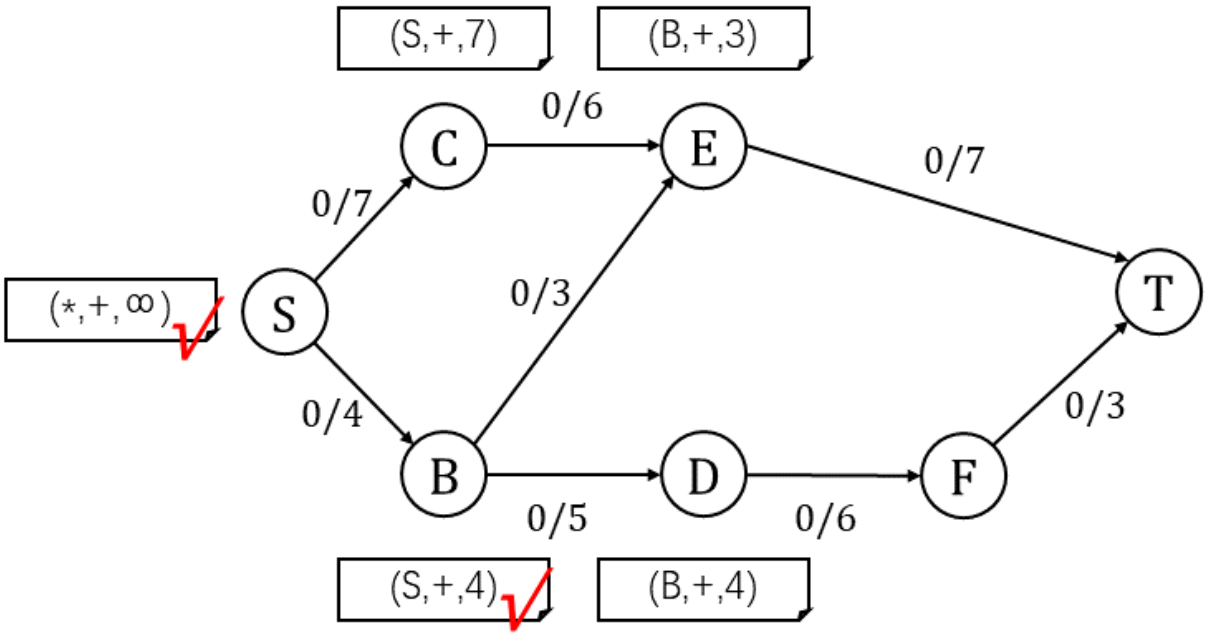
Example

ROUTINE A, ROUND 1



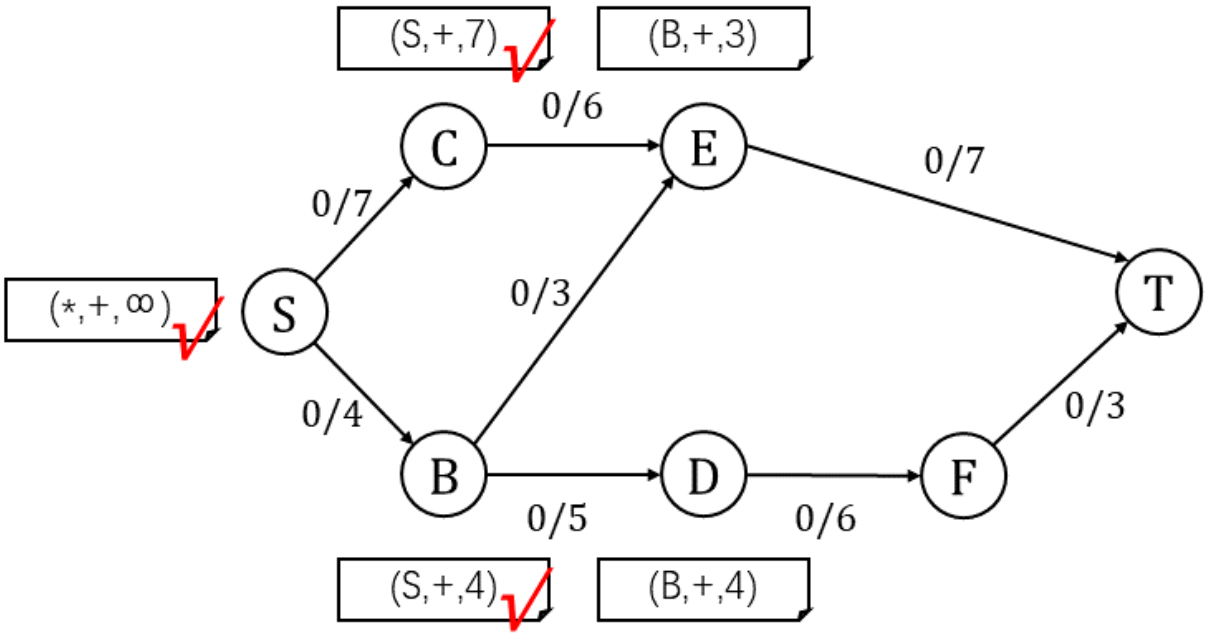
Example

ROUTINE A, ROUND 1



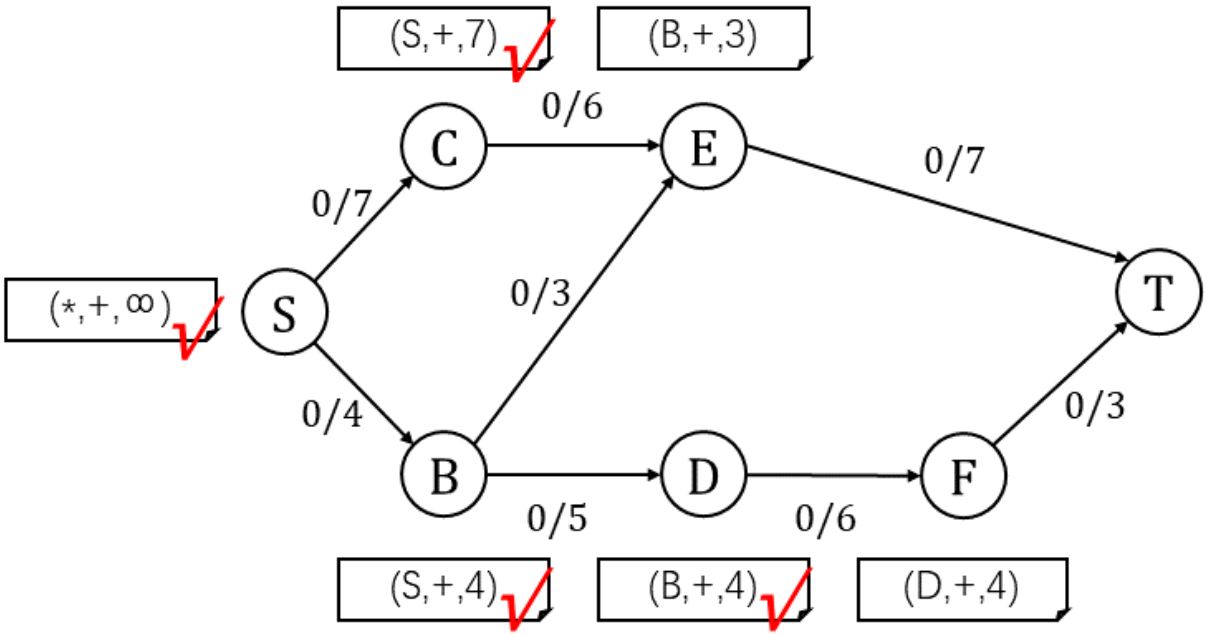
Example

ROUTINE A, ROUND 1



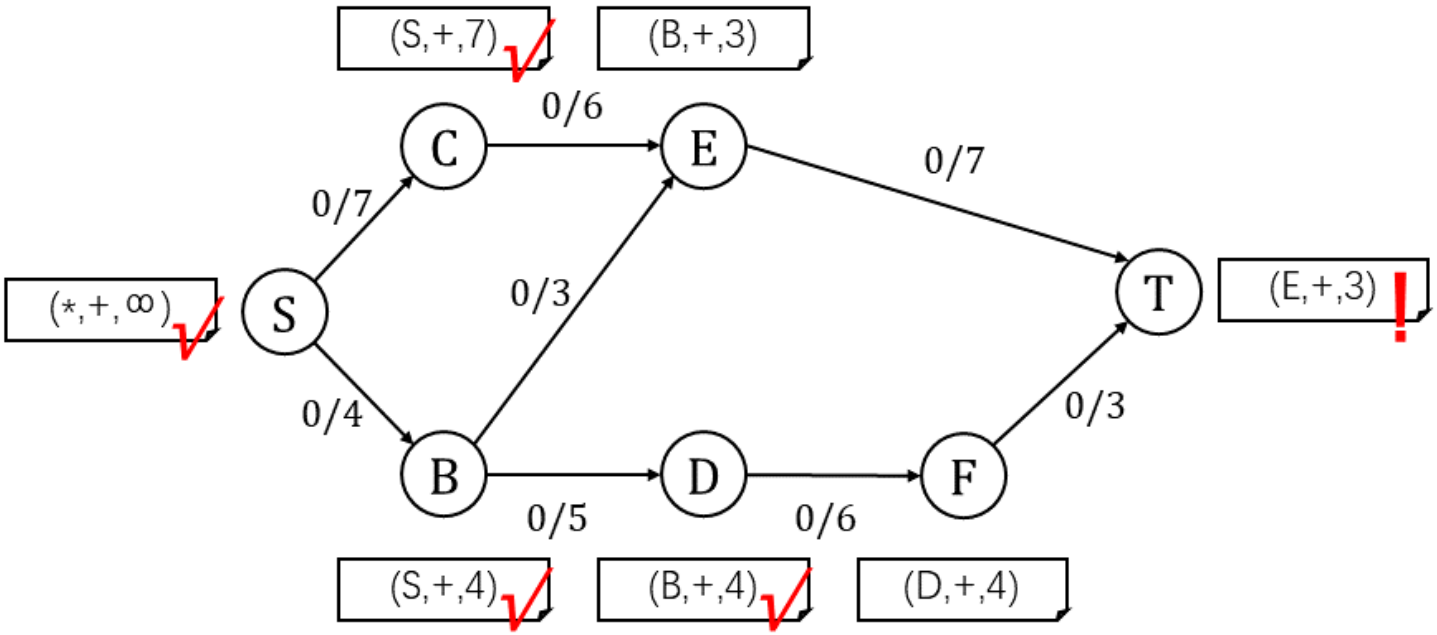
Example

ROUTINE A, ROUND 1



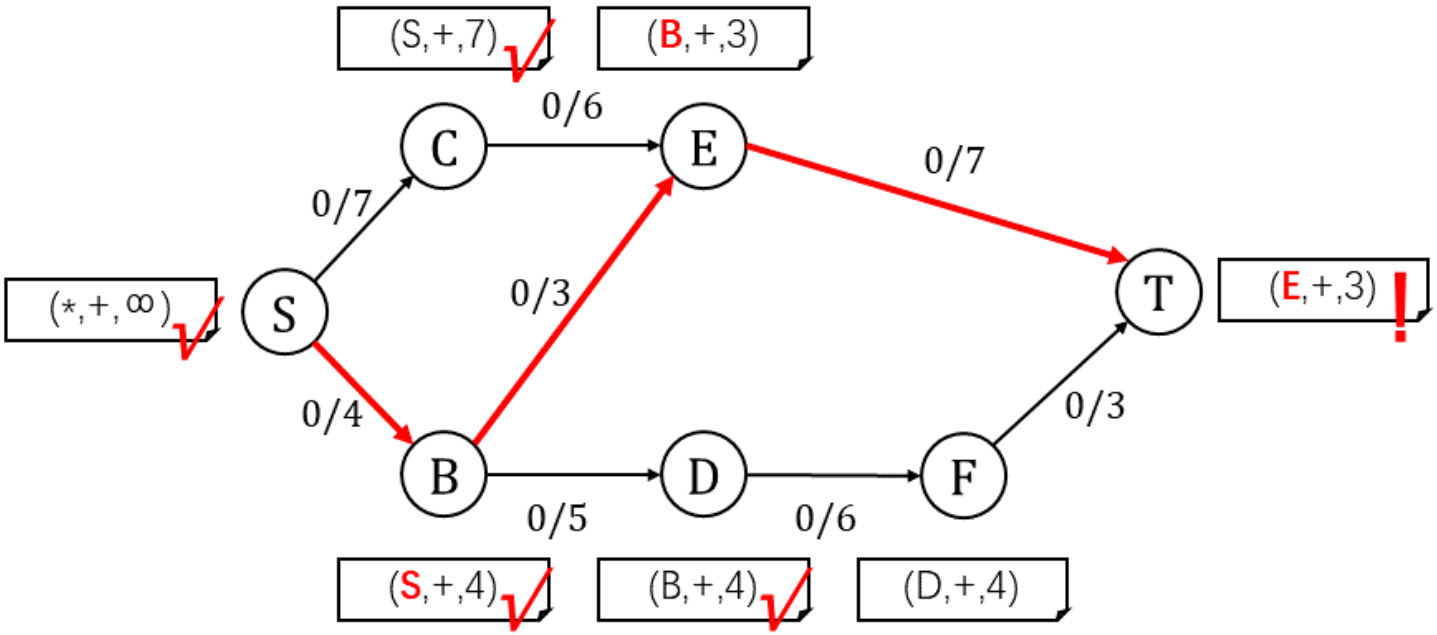
Example

ROUTINE A, ROUND 1



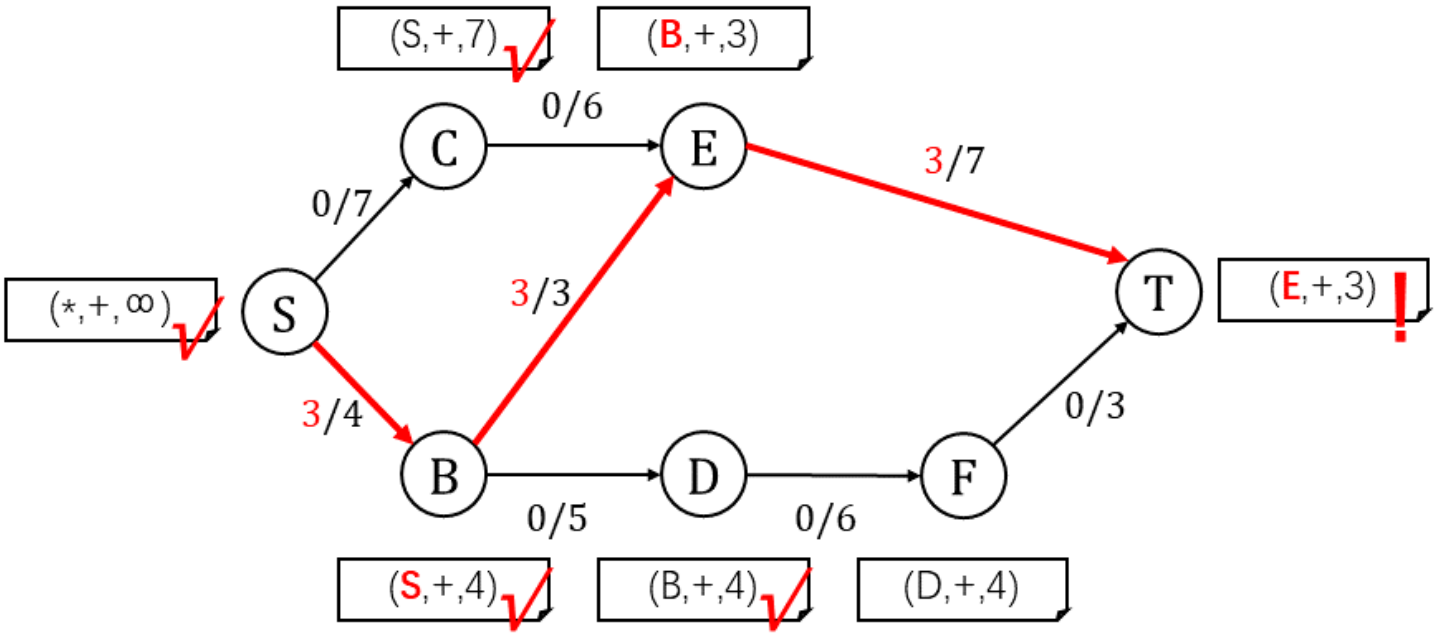
Example

ROUTINE A, ROUND 1



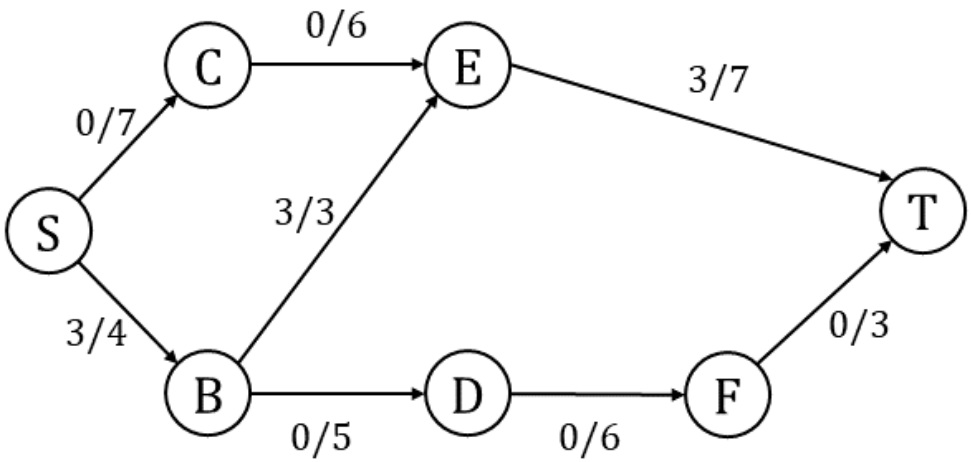
Example

ROUTINE B

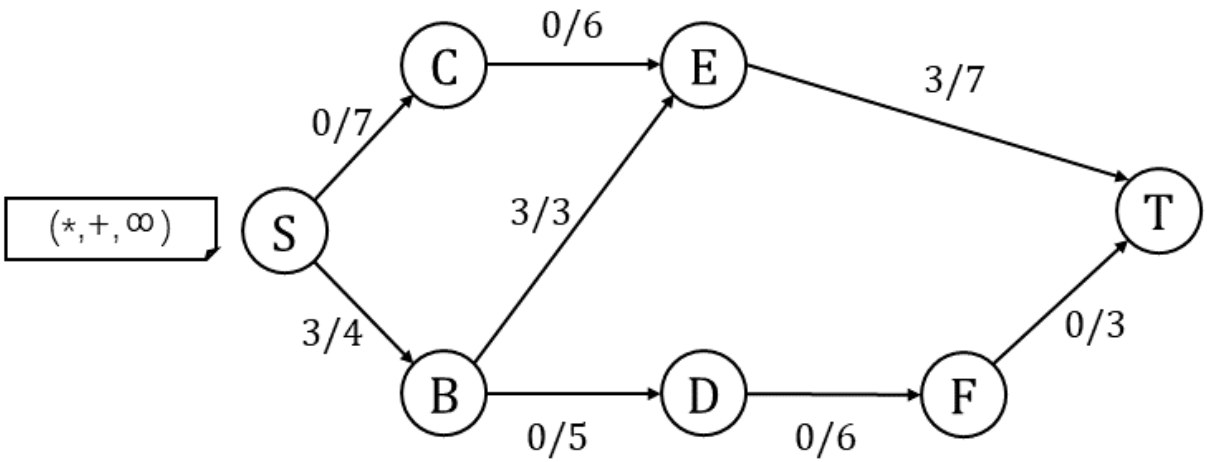


Example

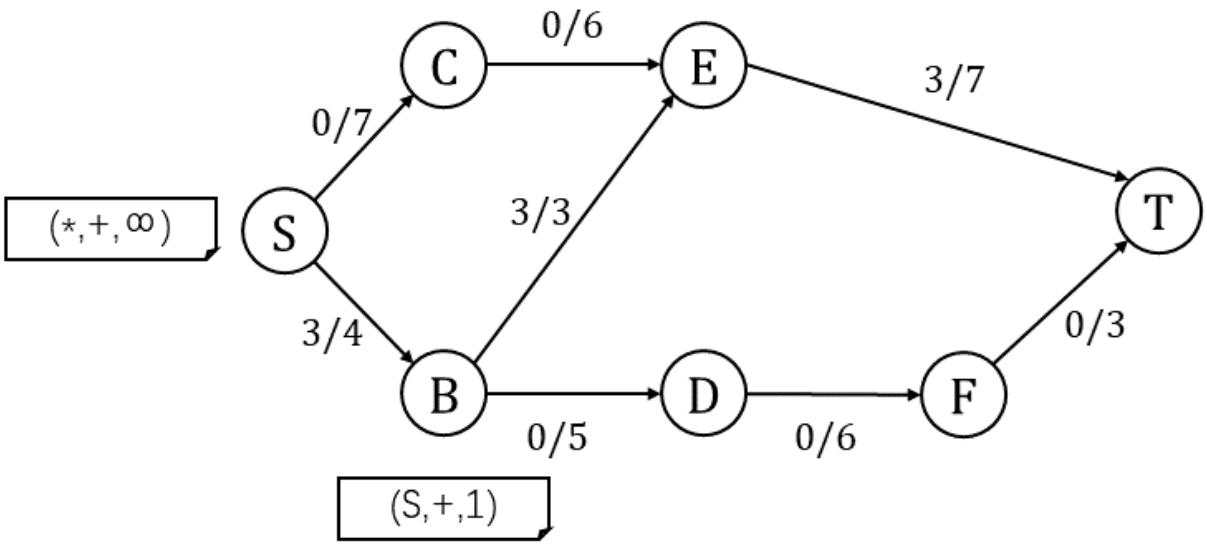
ROUTINE B



ROUTINE A, ROUND 2

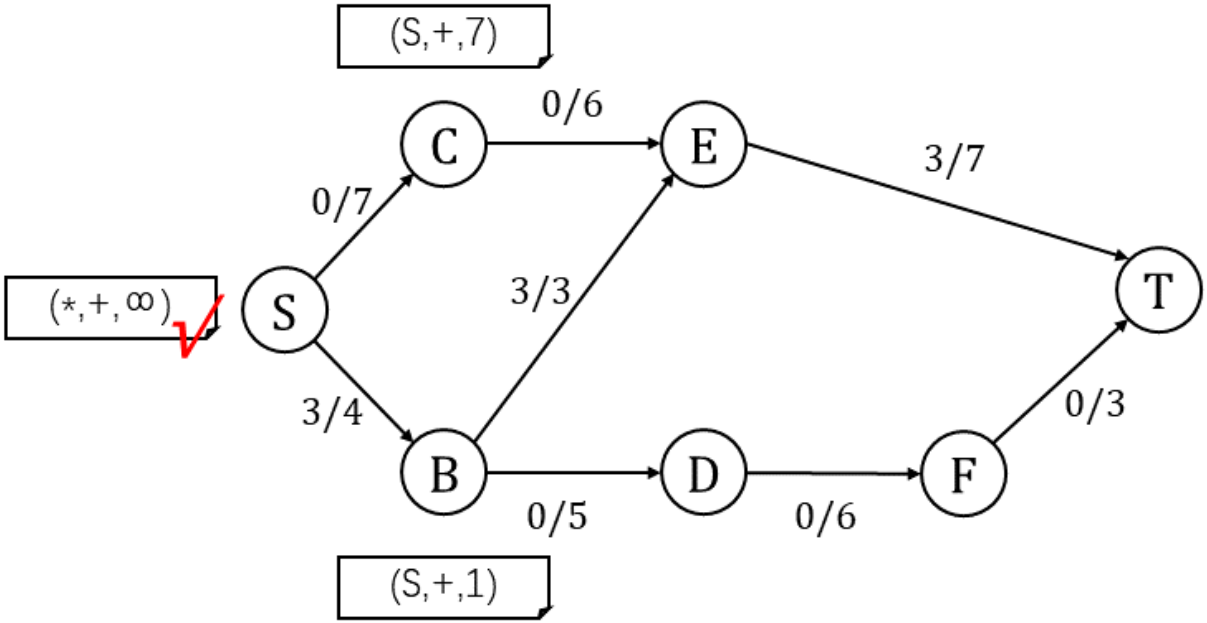


ROUTINE A, ROUND 2



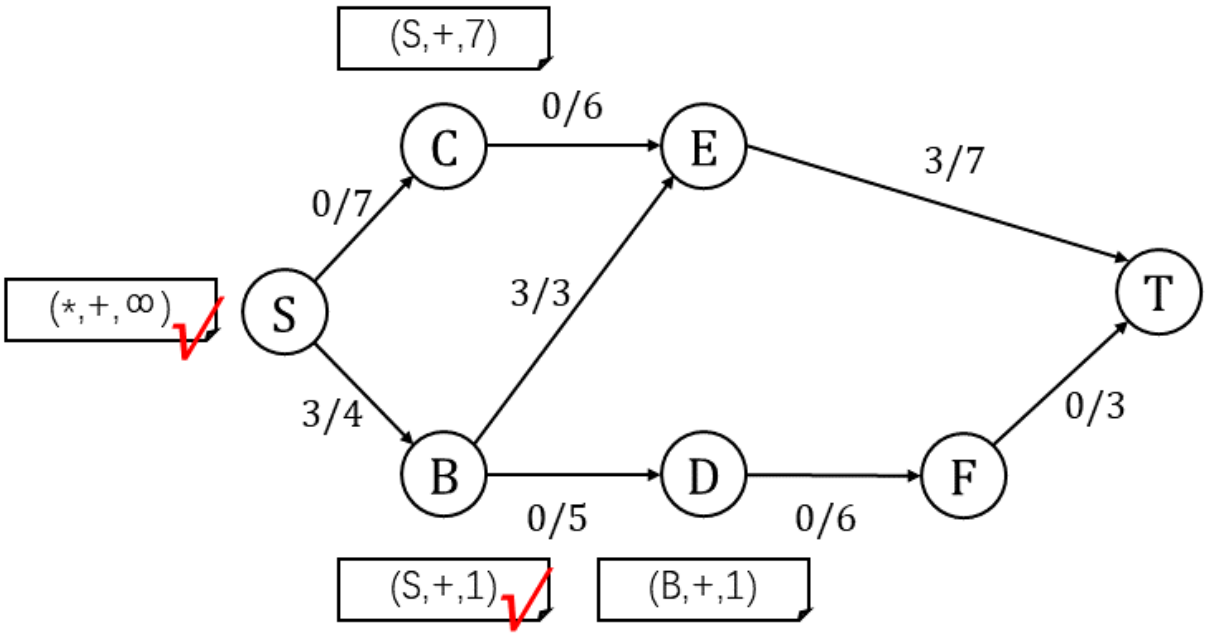
Example

ROUTINE A, ROUND 2



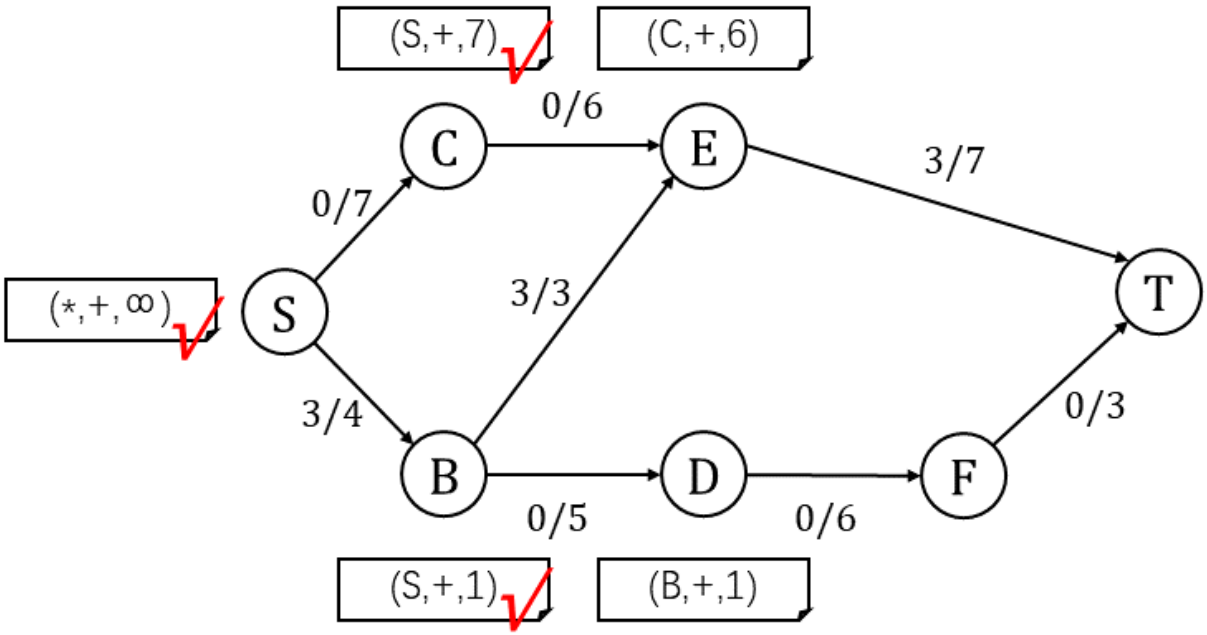
Example

ROUTINE A, ROUND 2



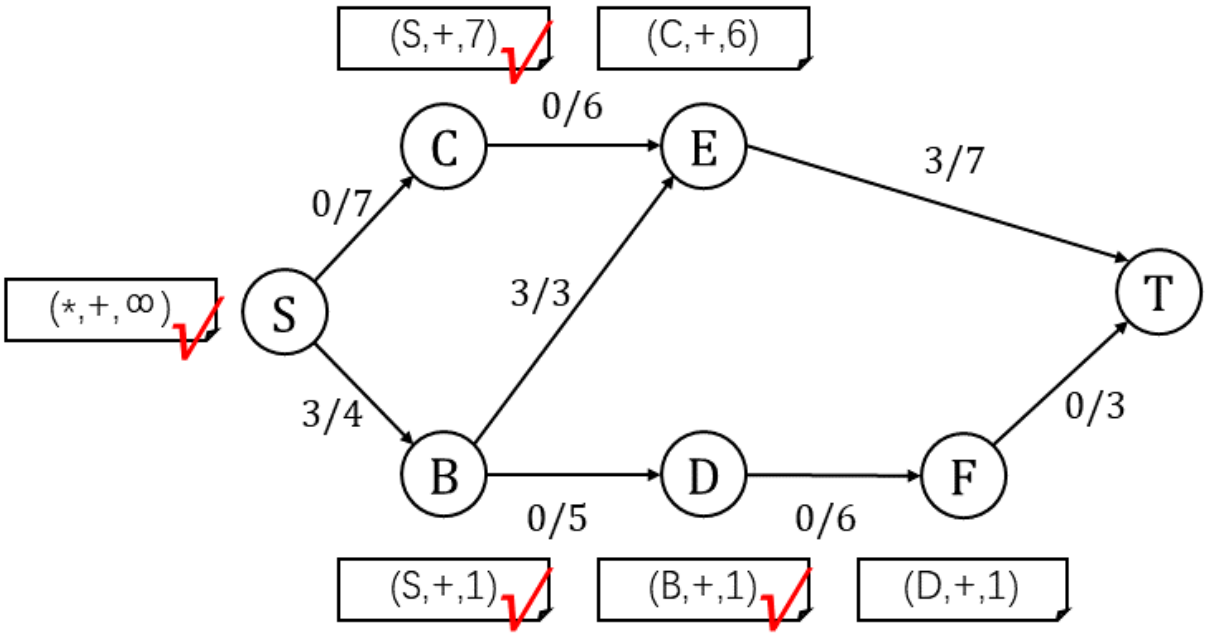
Example

ROUTINE A, ROUND 2



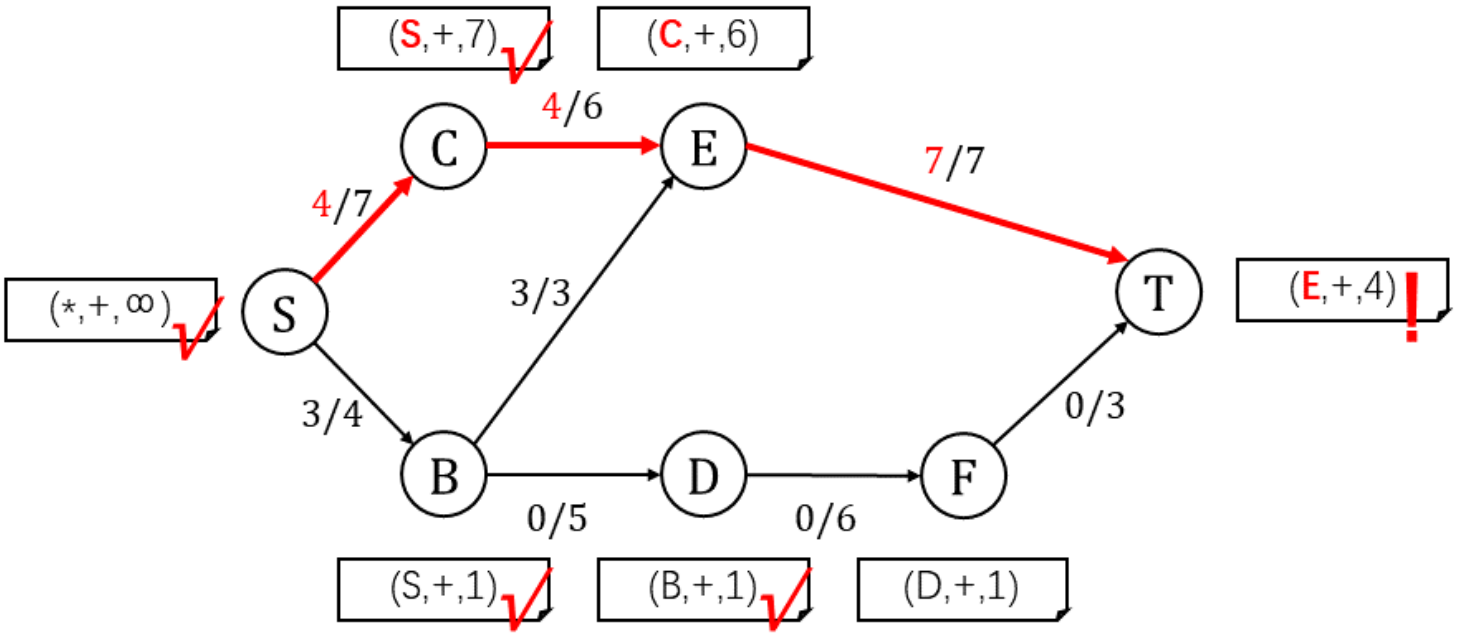
Example

ROUTINE A, ROUND 2



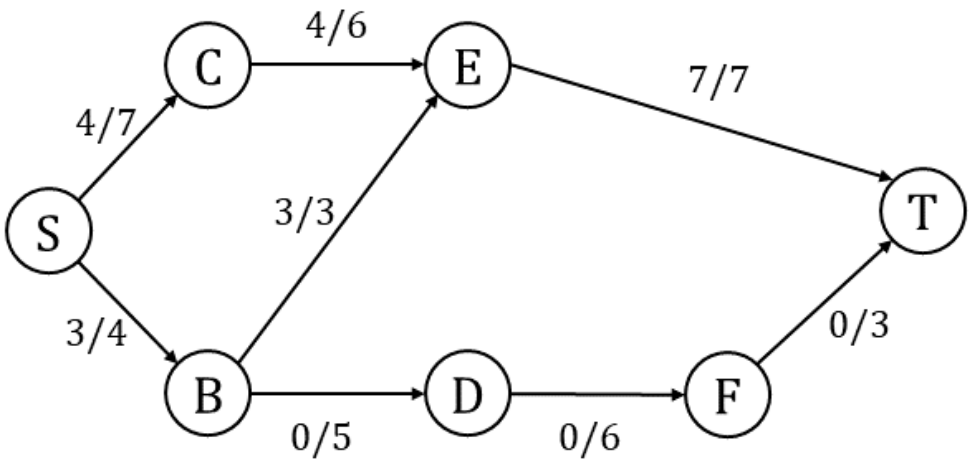
Example

ROUTINE B



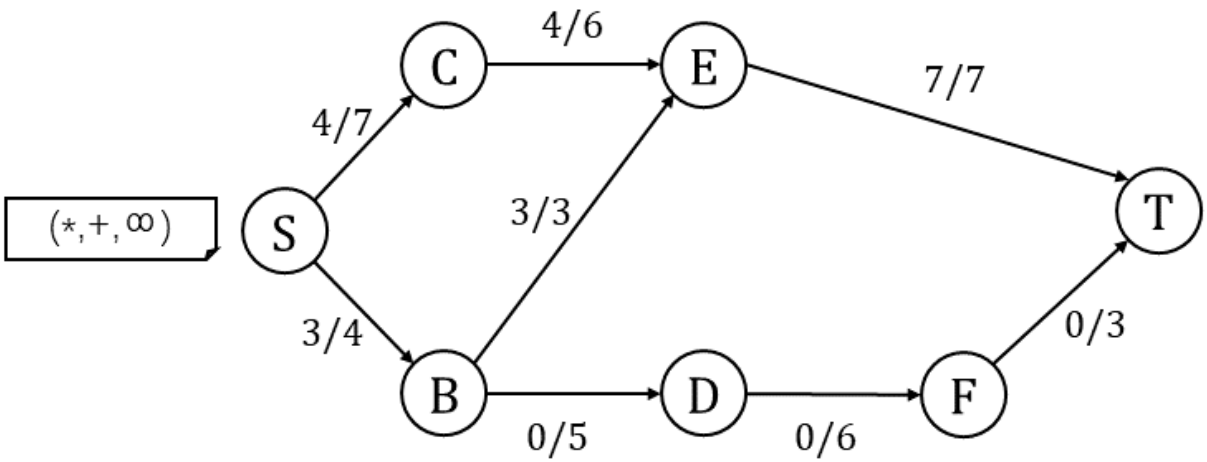
Example

ROUTINE B



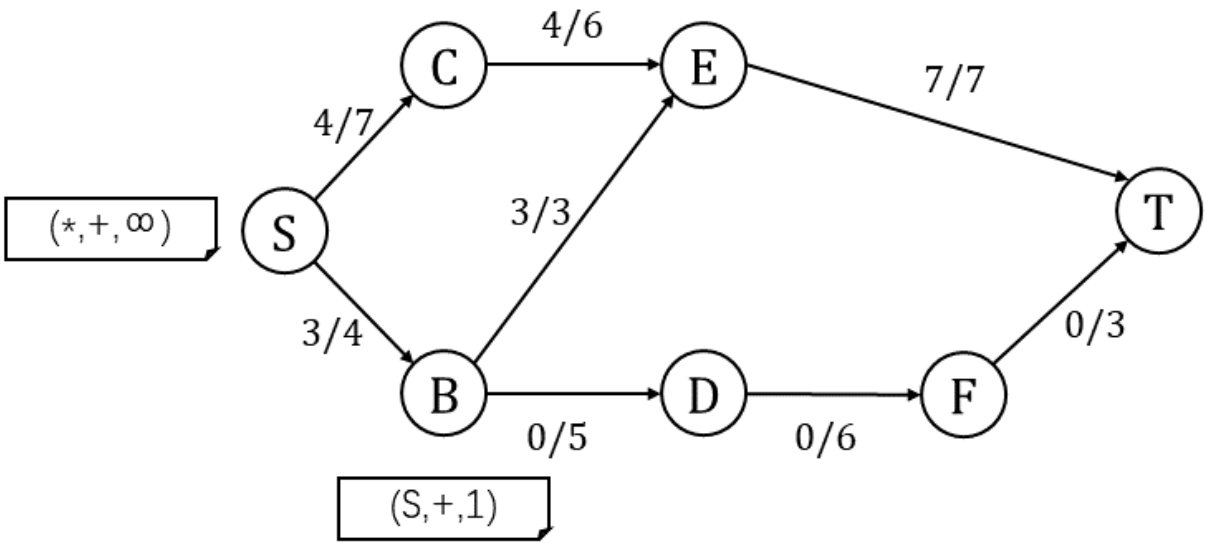
Example

ROUTINE A, ROUND 3



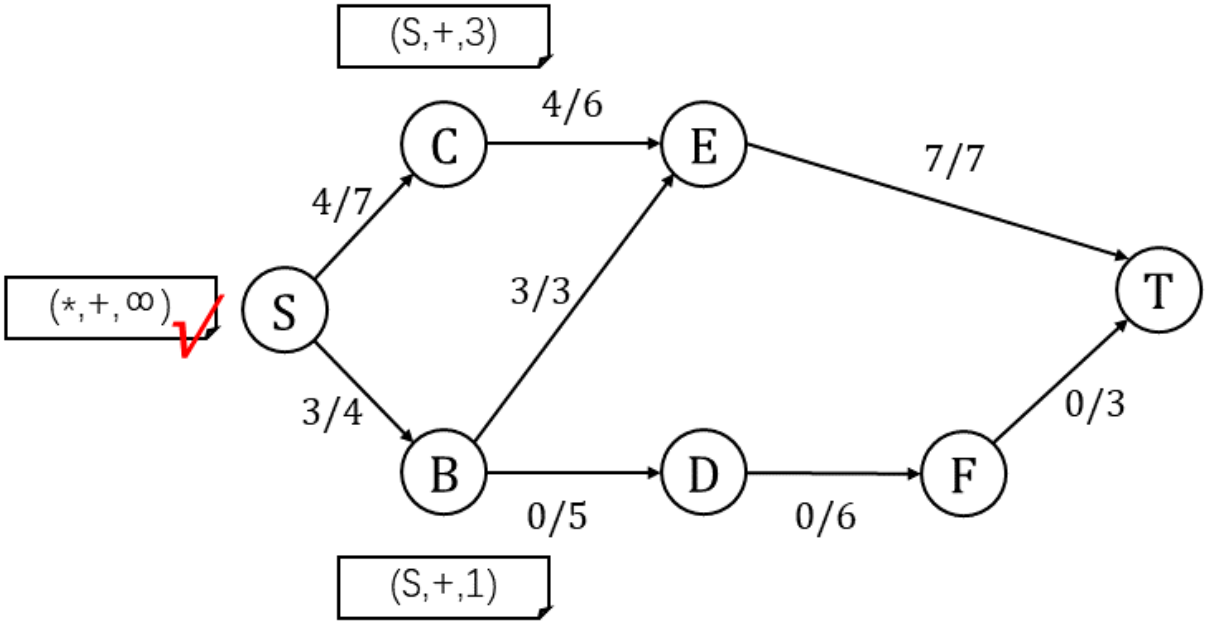
Example

ROUTINE A, ROUND 3



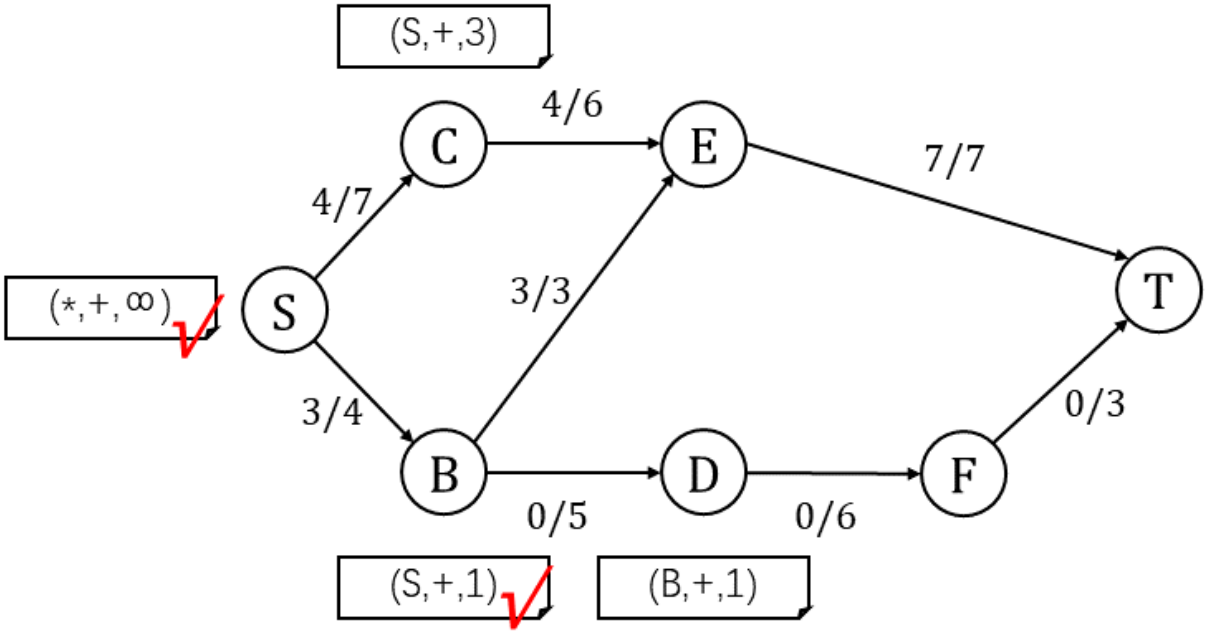
Example

ROUTINE A, ROUND 3



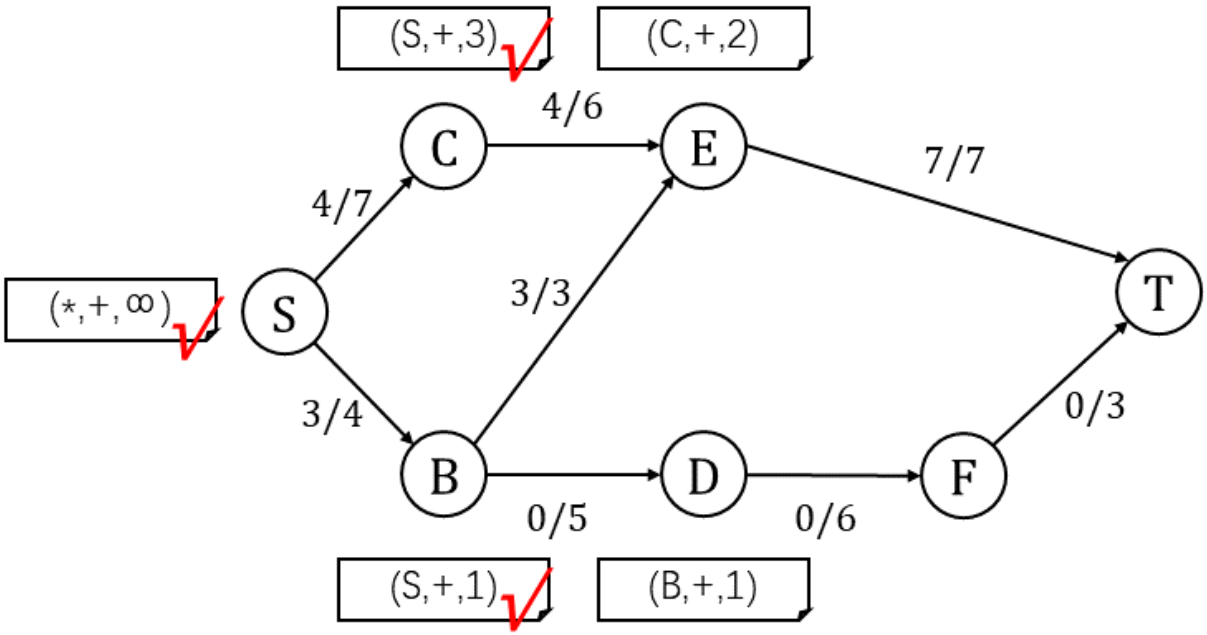
Example

ROUTINE A, ROUND 3



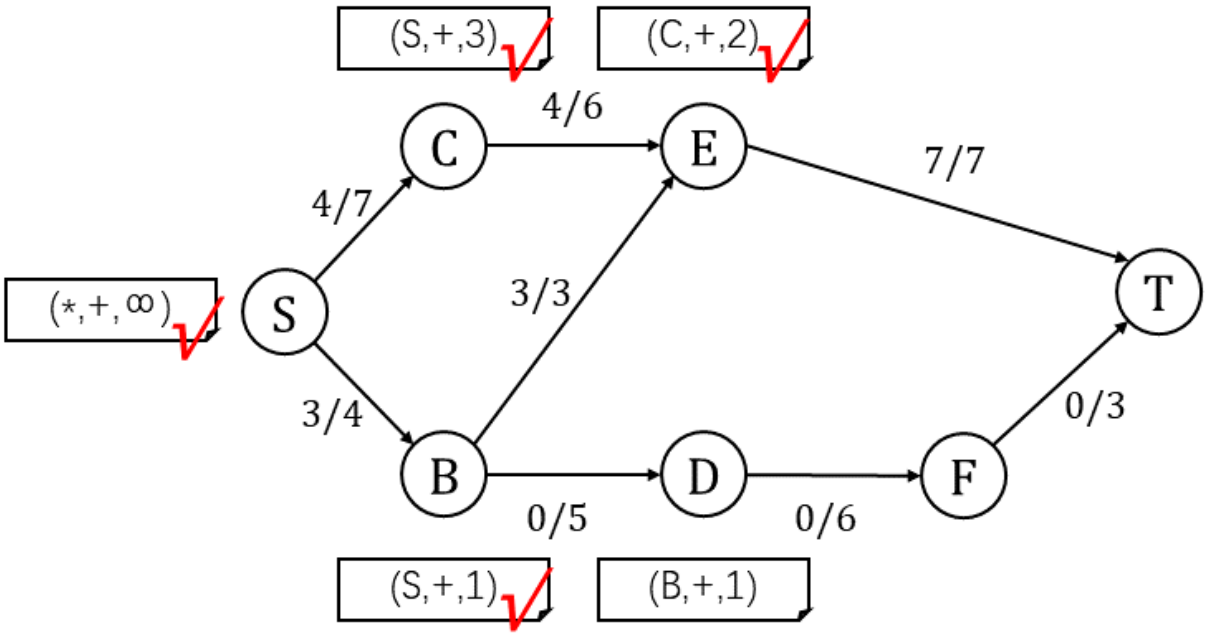
Example

ROUTINE A, ROUND 3



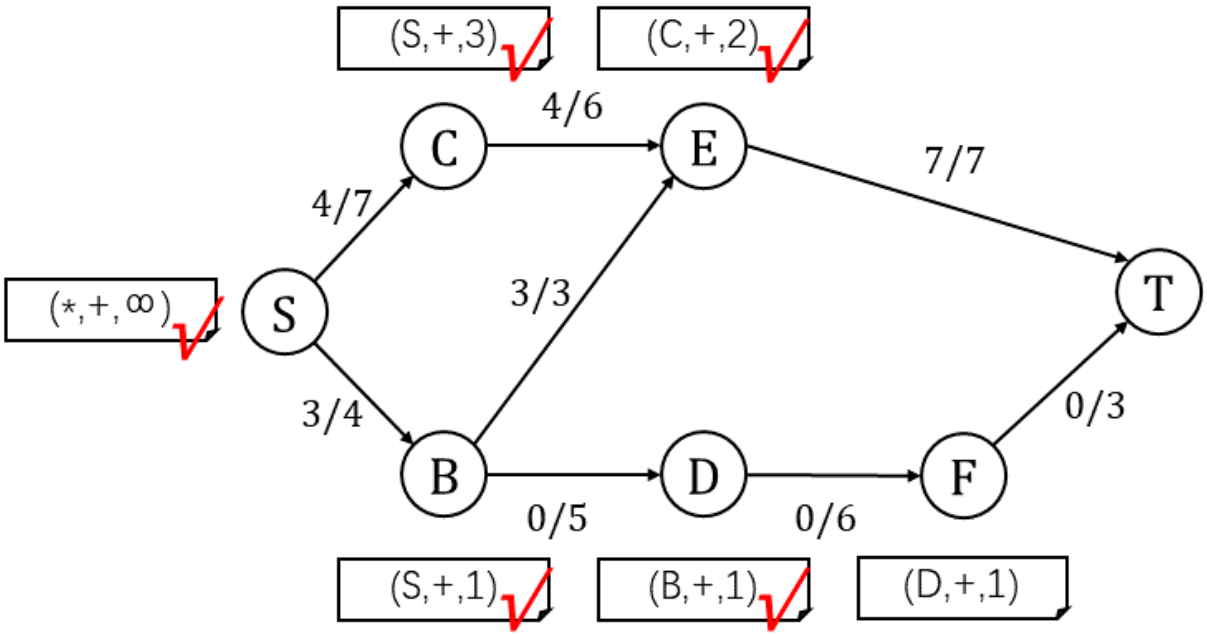
Example

ROUTINE A, ROUND 3



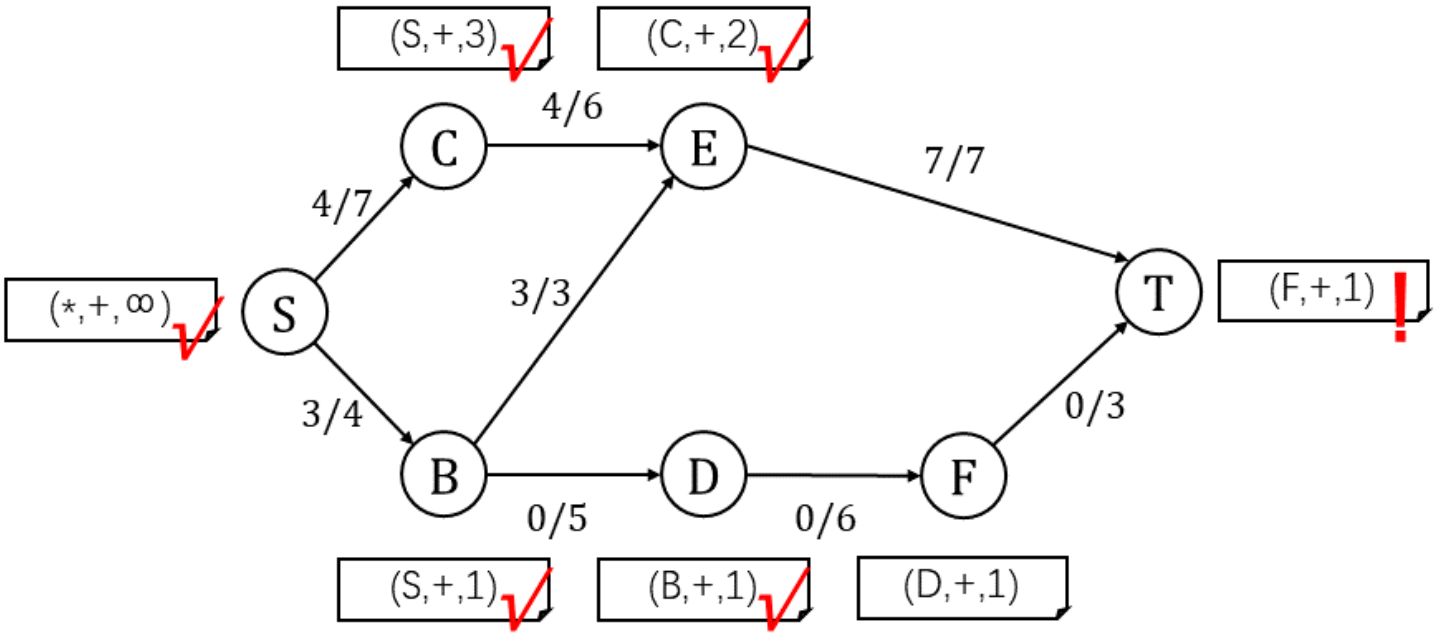
Example

ROUTINE A, ROUND 3



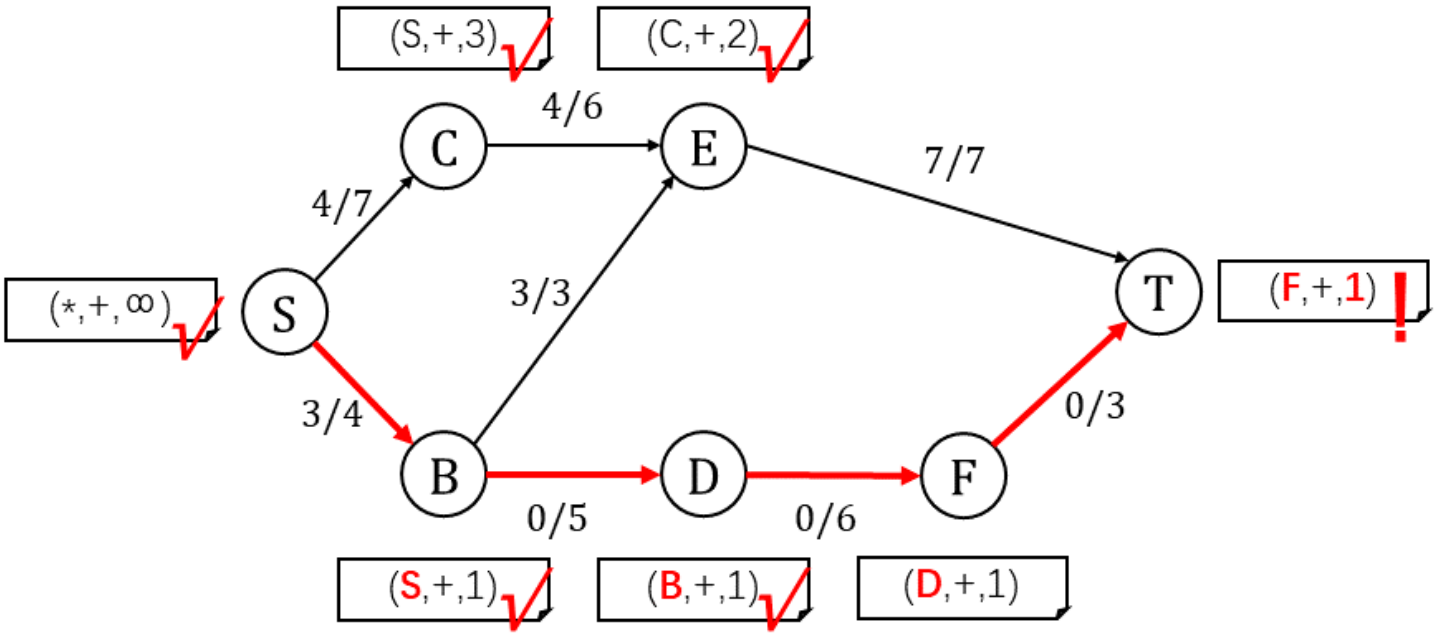
Example

ROUTINE A, ROUND 3



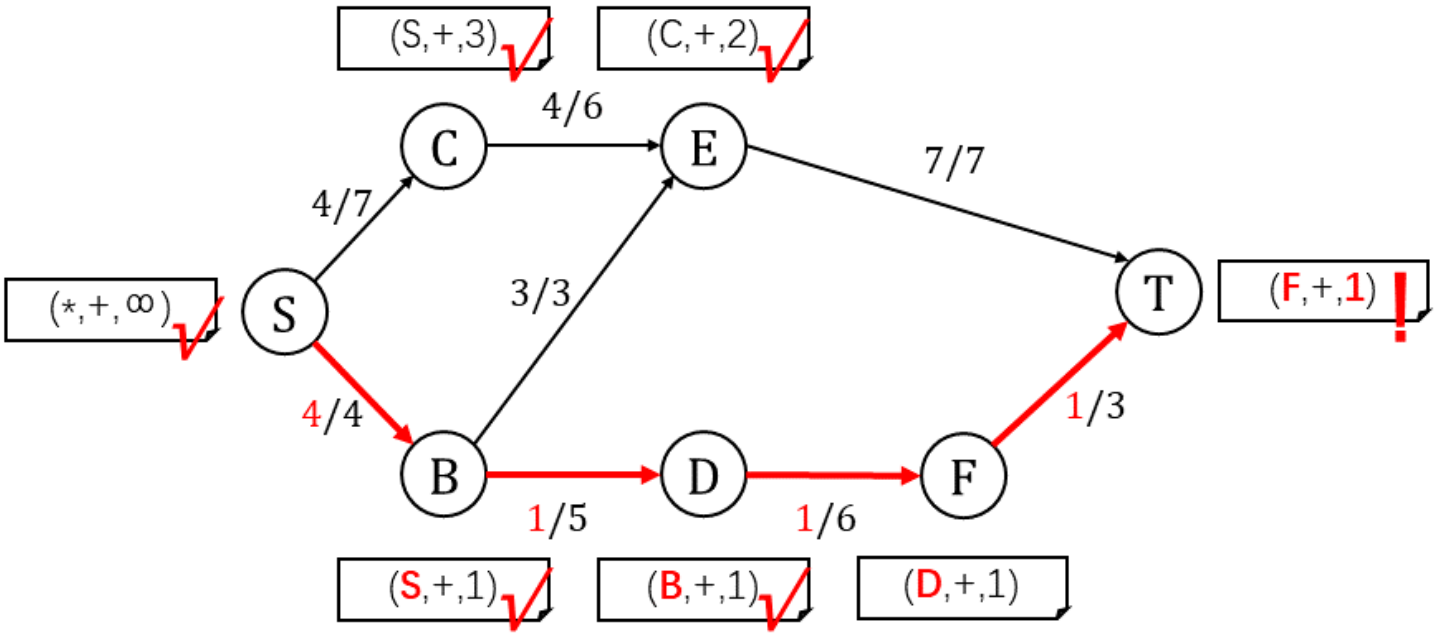
Example

ROUTINE A, ROUND 3



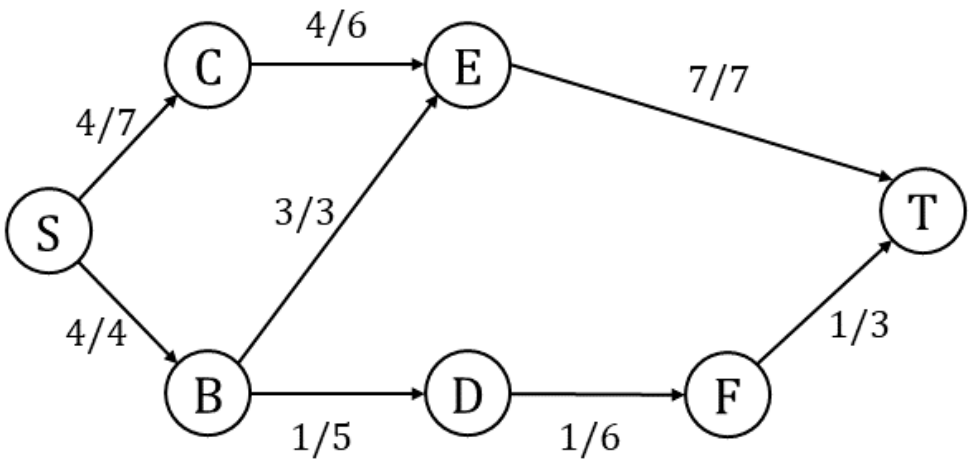
Example

ROUTINE B



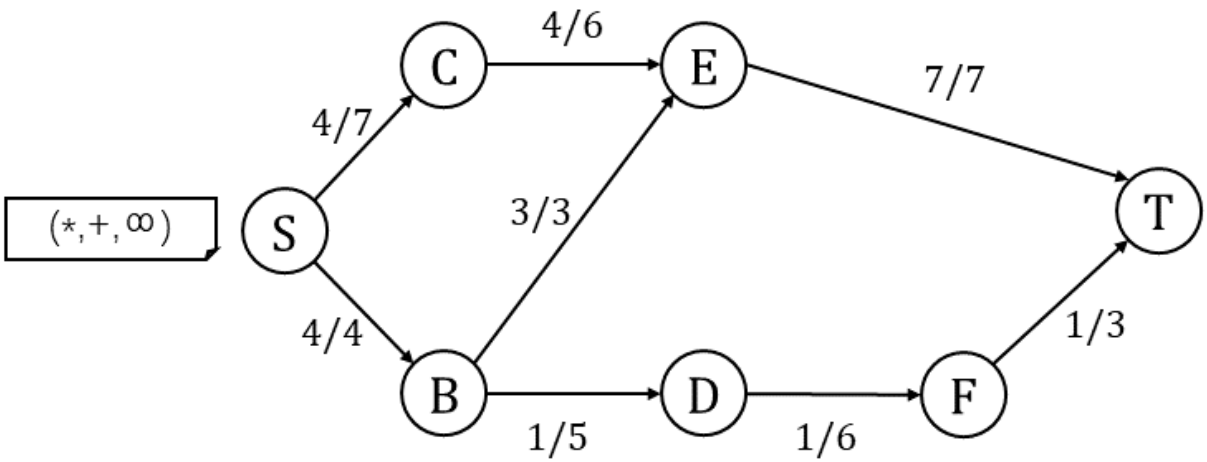
Example

ROUTINE A, ROUND 4



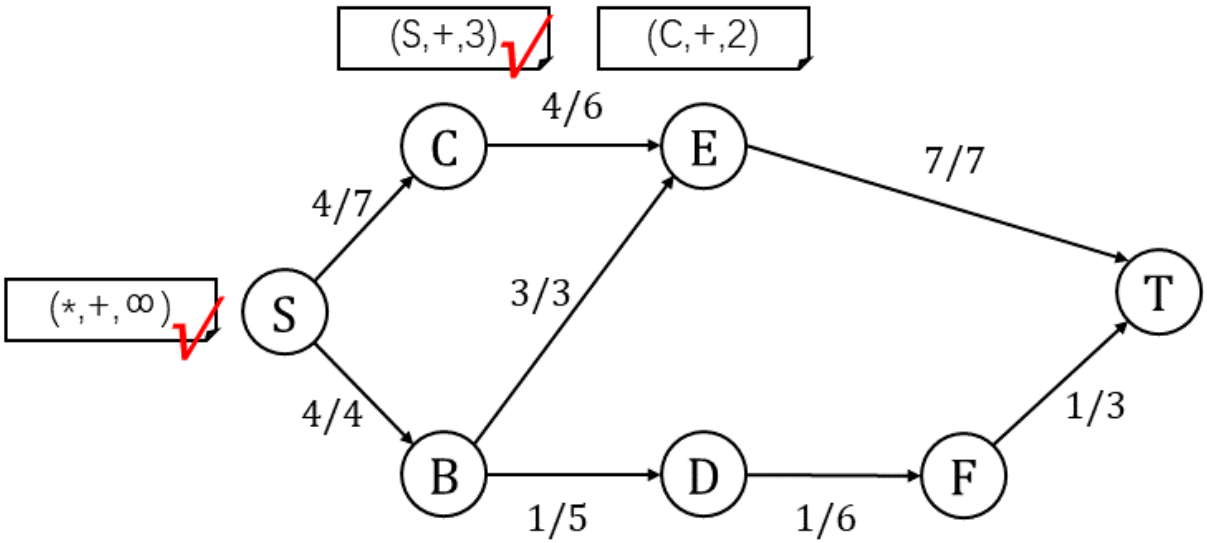
Example

ROUTINE A, ROUND 4



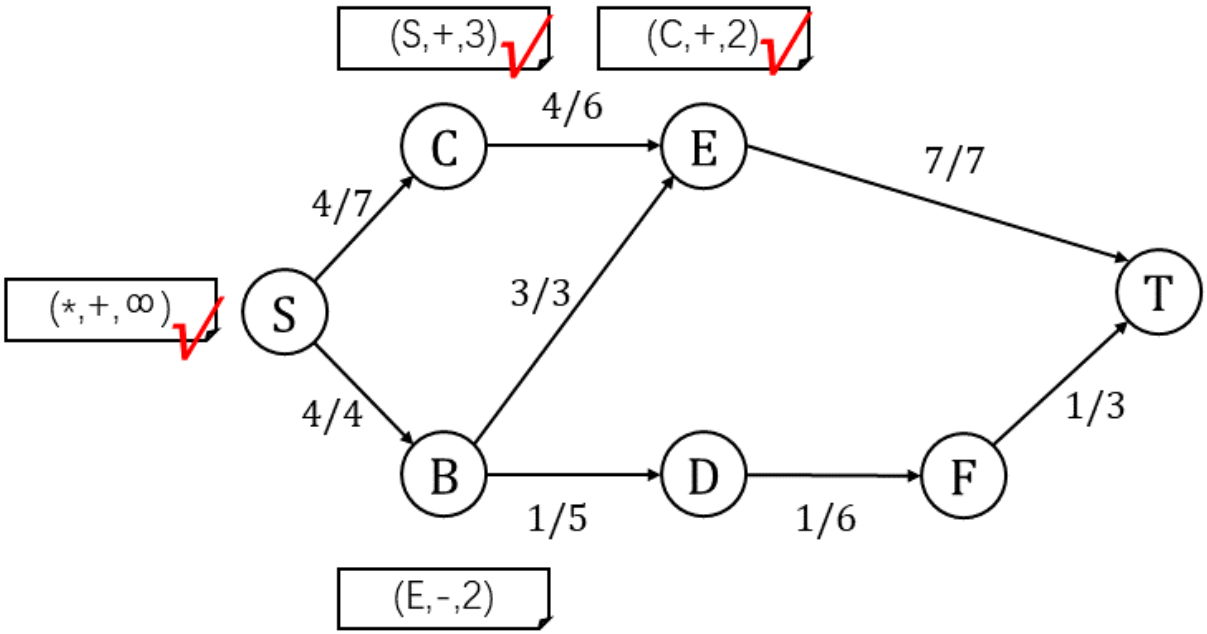
Example

ROUTINE A, ROUND 4



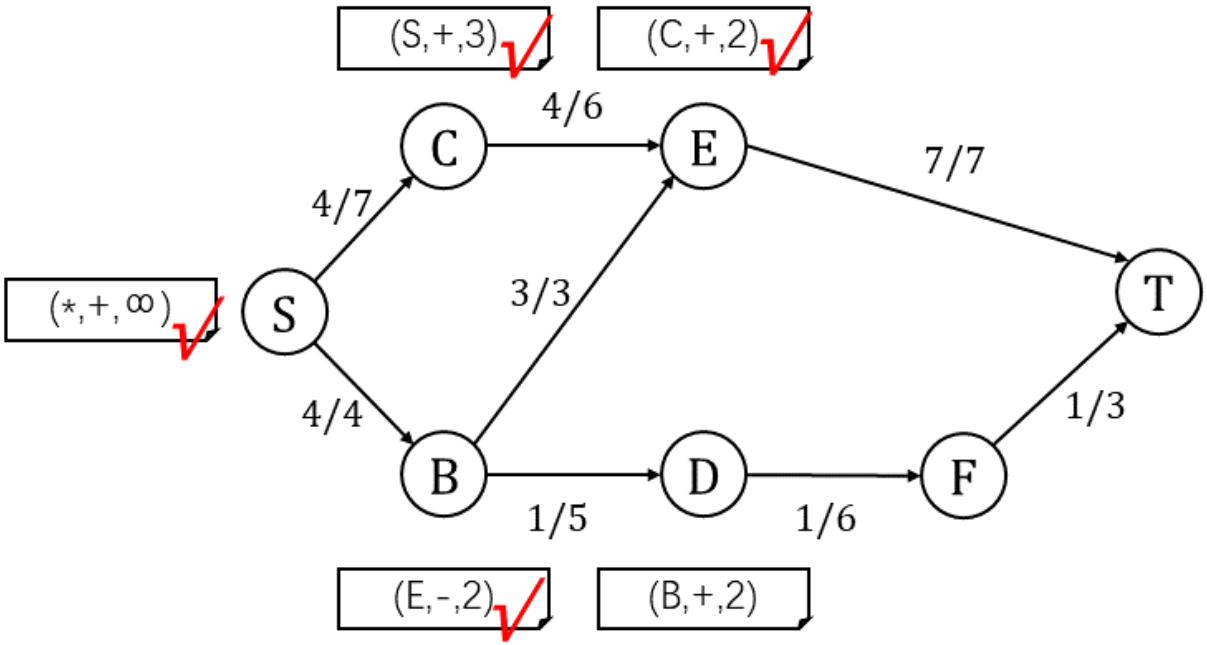
Example

ROUTINE A, ROUND 4



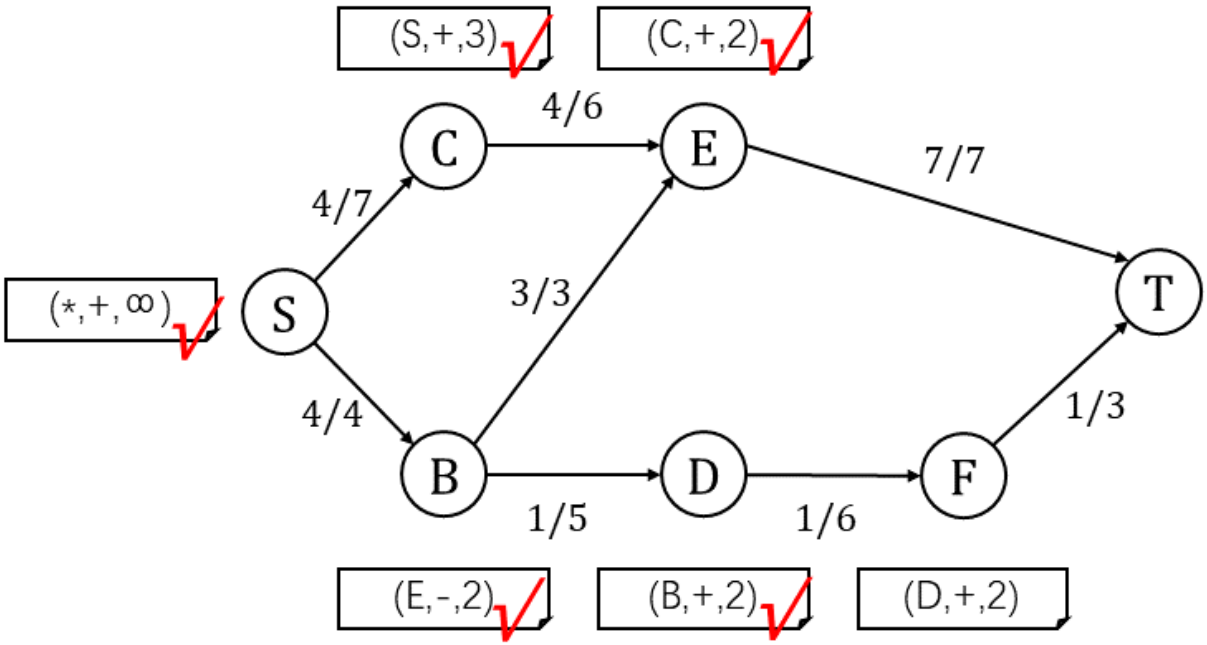
Example

ROUTINE A, ROUND 4



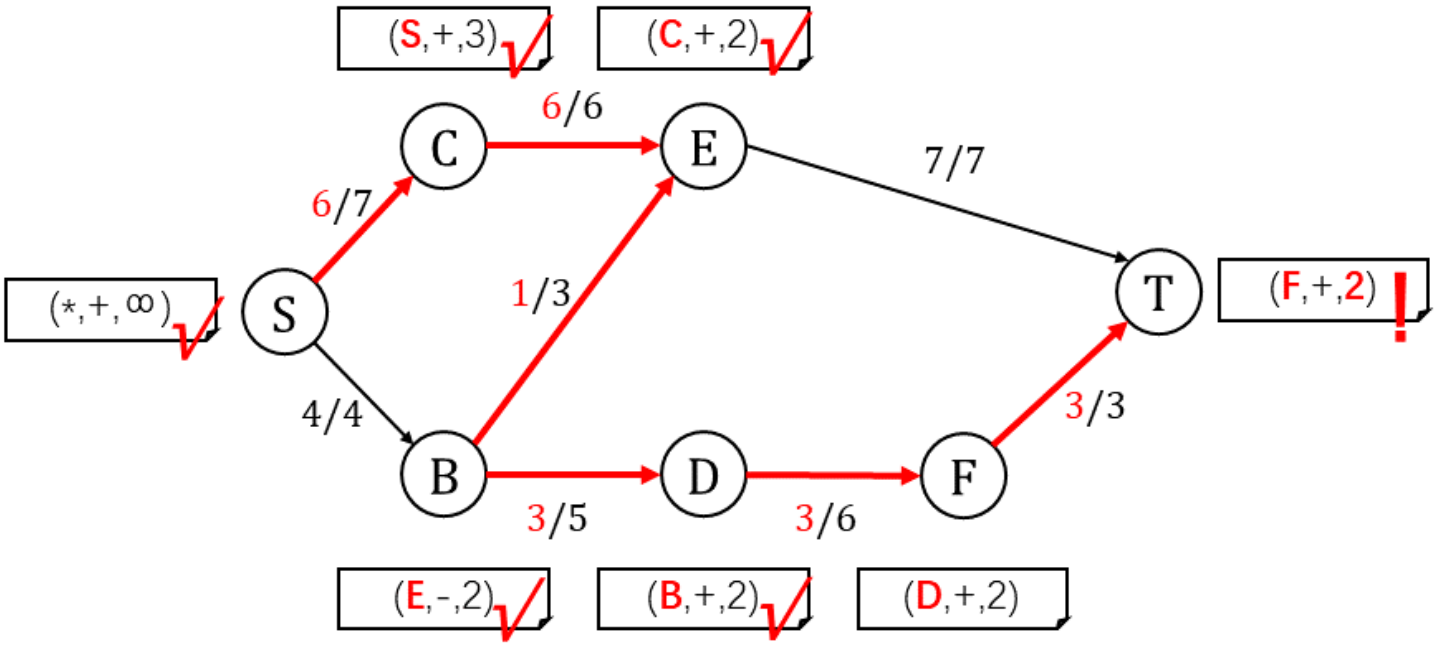
Example

ROUTINE A, ROUND 4



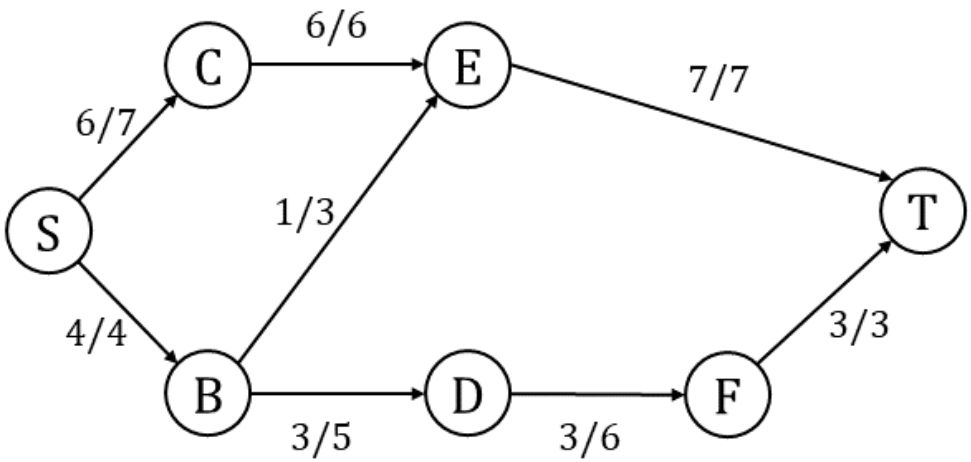
Example

ROUTINE B



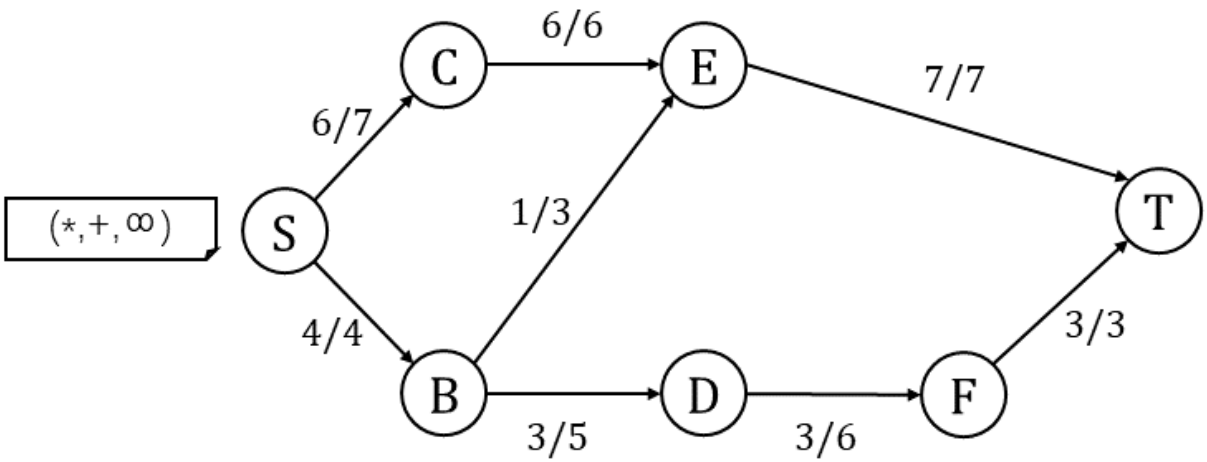
Example

ROUTINE B



Example

ROUTINE A, ROUND 5



Example

TERMINATION