A^* Algorithm

The A^* algorithm solves the single source minpath (least weight path) problem for a weighted directed graph. Let $w(x, y)$ be the weight of the arc, from x to y.

In the example below, the edges are not directional, but we simply assume that each edge represents two arcs, one in each direction.

A^* is an "intelligent" version of Dijkstra's algorithm, which finds the a minal path from the source S to all vertices. A^* is restricted to just one target vertex, T.

Heuristic

To work the algorithm, we must first obtain a value $h(x)$, shown in red in the figues, for each vertex x. For each x, $h(x)$ must be a positive number which is no greater than the least distance from x to T. Letting $h(x) = 0$ for all x is a valid choice: in that case the rounds of A^* duplicate the steps of Dijkstra's algorithm. The best choice is to let $h(x)$ be the true distance from x to T. That choice is clearly not obtainable in practice, since if we knew those values, we would already have a solution!

The heuristic should satisfy $h(x) - h(y) \geq w(x, y)$ for all vertices x and y. We say h is *monotone* or *consistent*.

Crow Flies

In an important practical case, where the distance from x to T follows a system of roads, a good choicd of heuristic could be the geodesic distance. Consistency is guaranteed in this case.

Steps of A^*

As in Dijkstra's algorithm, every vertex is either unprocessed, partially processed (OPEN), or fully processed (CLOSED) at each round. Inially, S is partially processed and all other vertices are unprocessed.

If x is partially processed, $f(x)$, shown in black in the figures, is the least cost of any path from S to x found so far. If x is fully processed, $f(x)$ is the

least cost of any path from S to x.

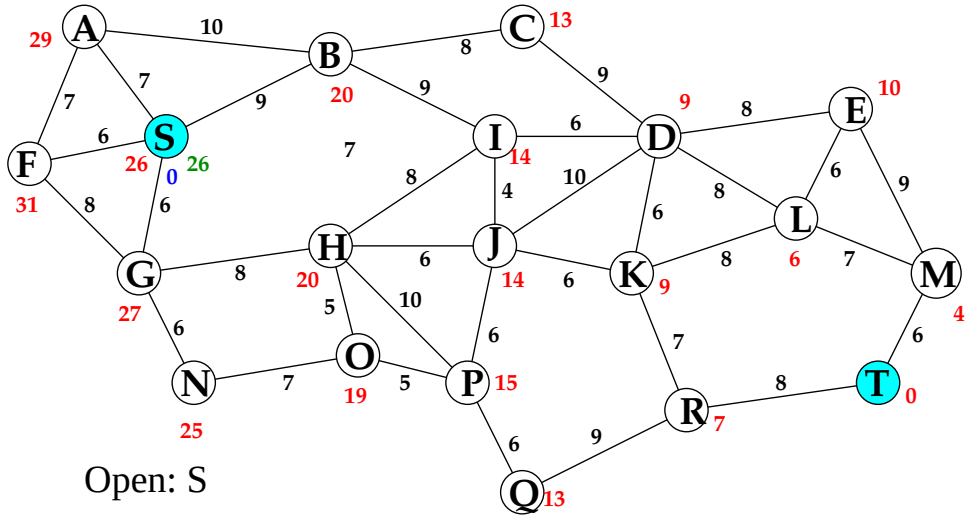
For fully and partially processed vertices, $g(x) = f(x)+h(x)$, shown in green in the figures.

At each round of the A^* algorithm, the following steps are executed.

1. The partially processed vertex x which has the smallest value of $g(x)$ is chosen.
2. For each out-neighbor y of x which is unprocessed, let $f(y) = f(x) + w(x, y)$ y becomes partially processed. the back pointer back(x) = y shown as a dashed magenta arrow in the figures, is defined.
3. For each out-neighbor z of x which is partially processed. compute temp $= f(z) + w(x, z)$. If temp $< f(x)$, redefine $f(x) =$ temp and redefine back$(x) = z$.
4. x is now fully processed.
5. If $x = T$, the algorithm halts. The least cost path, of weight T, may be found by following back pointers starting at T.

Example Calculation

We execute A^* on a weighted graph. Initially, S is the sole open vertices.



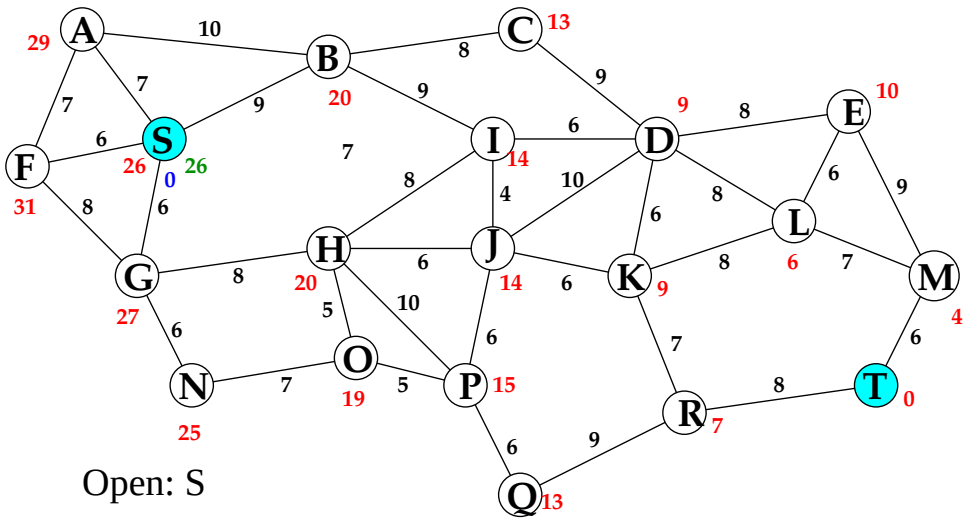
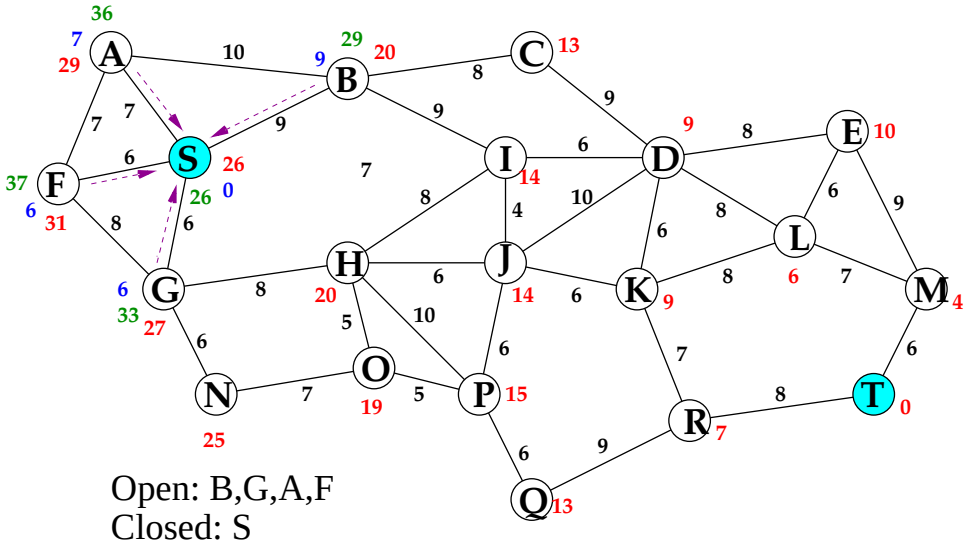


Figure B: S becomes fully processed (closed), and its neighbors inserted in the minqueue, whose items are shown in incresing order of g.



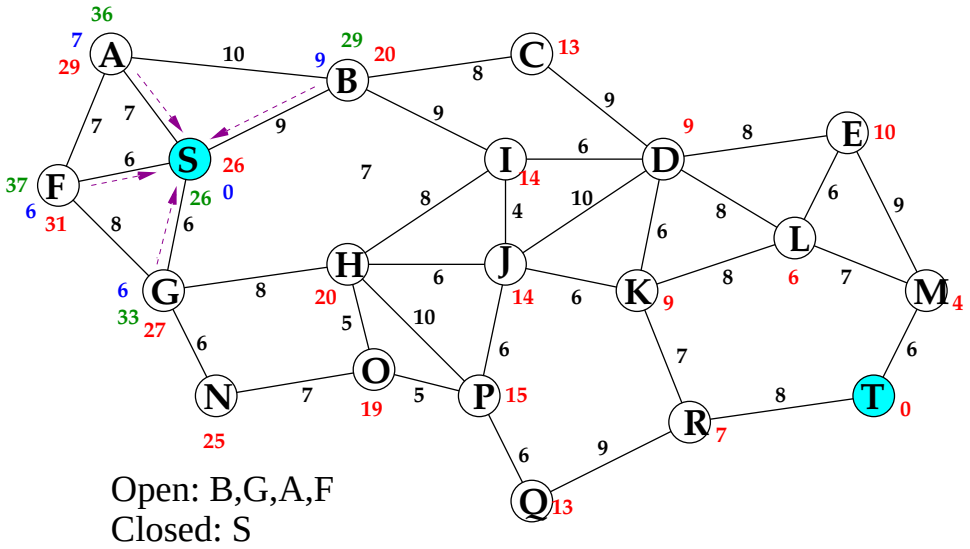
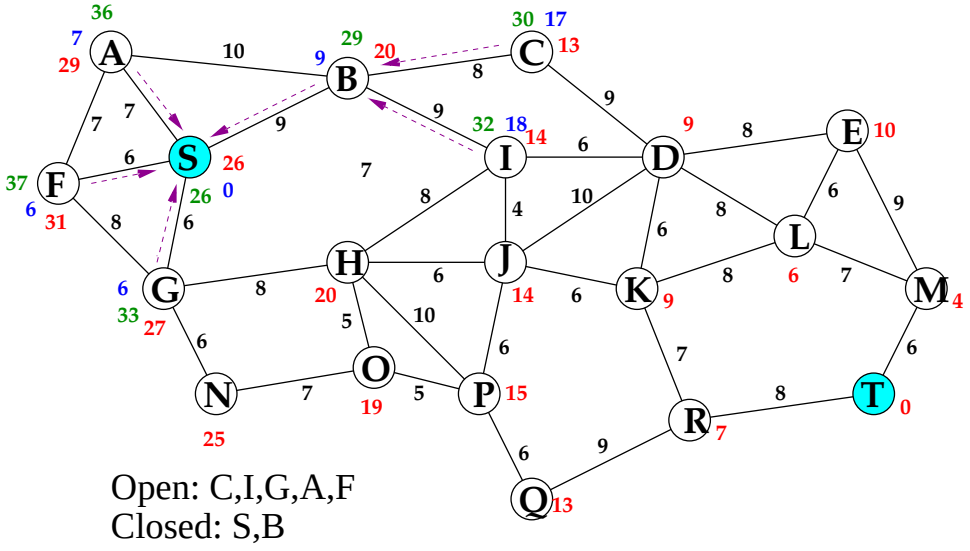


Figure C: B is fully processed, and its neighbors C and I are partially processed.



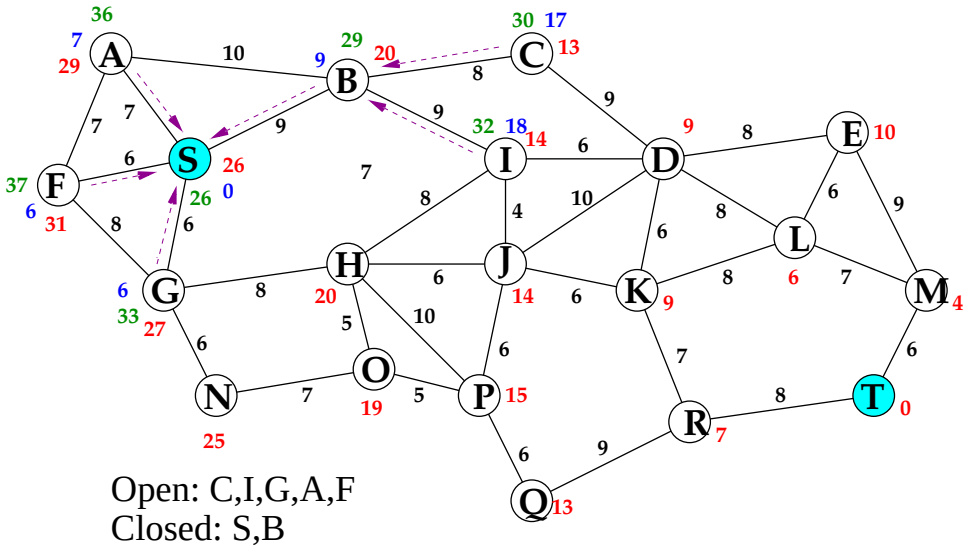
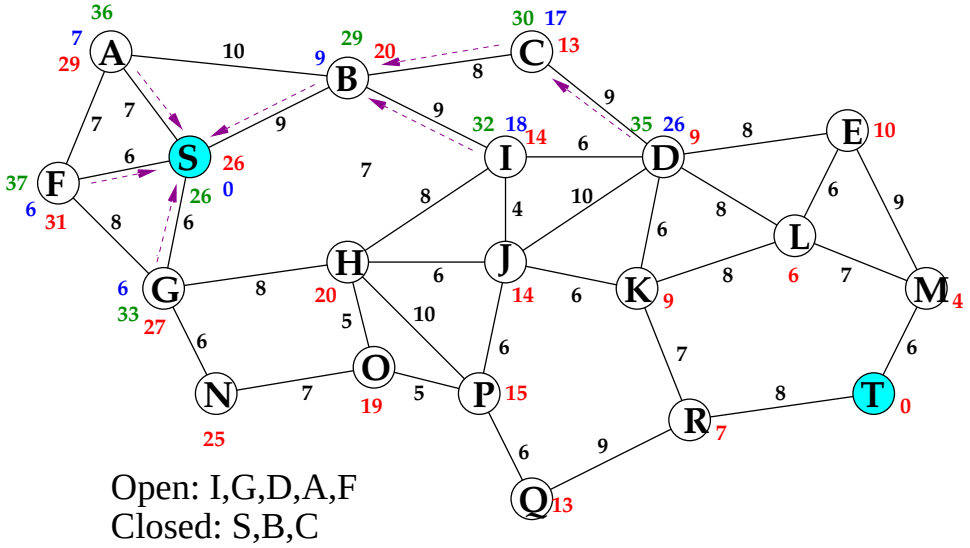


Figure D: C is fully processed, and its neighbors G and D are partially processed.



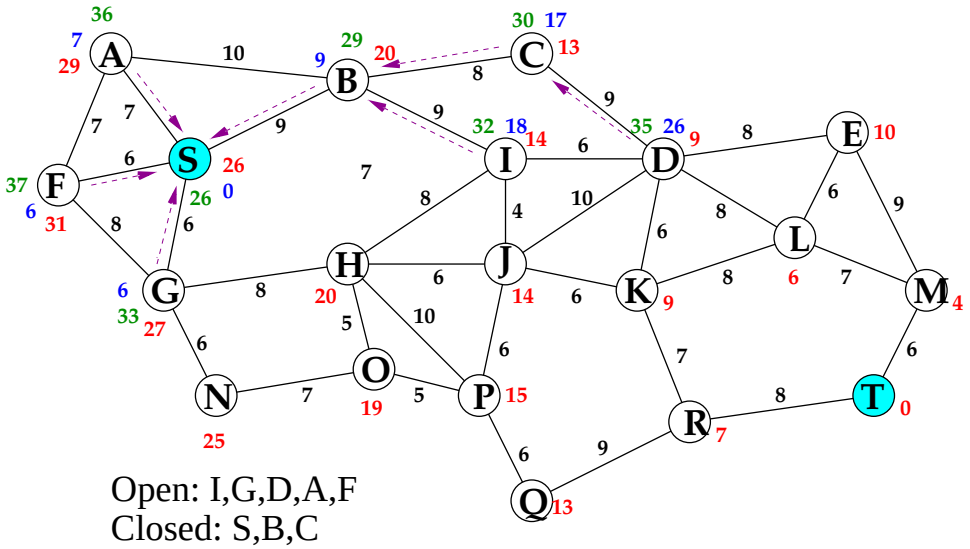
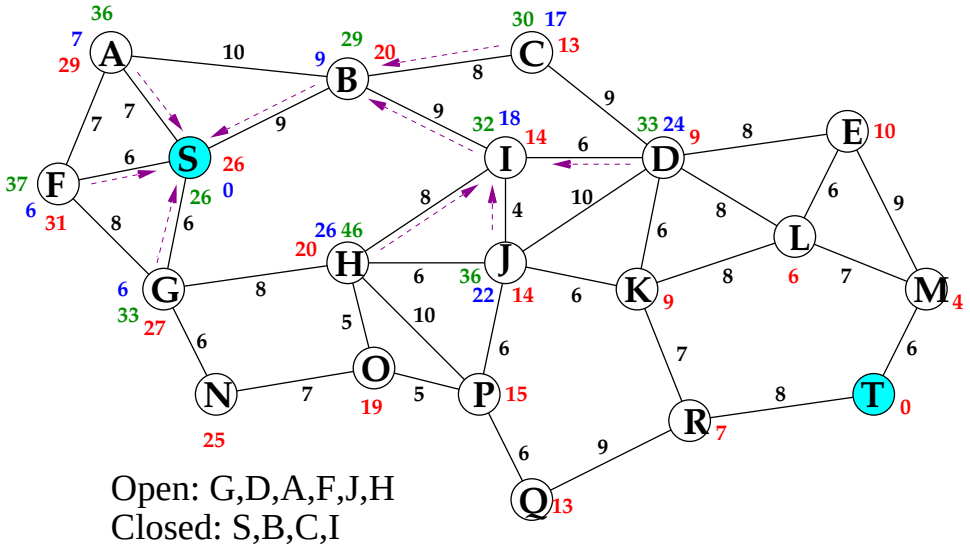


Figure E: I is fully processed, and its neighbors B and I are partially processed. D remains partially processed, but has a new backpointer, hence $f(D)$ and $g(D)$ decrease.



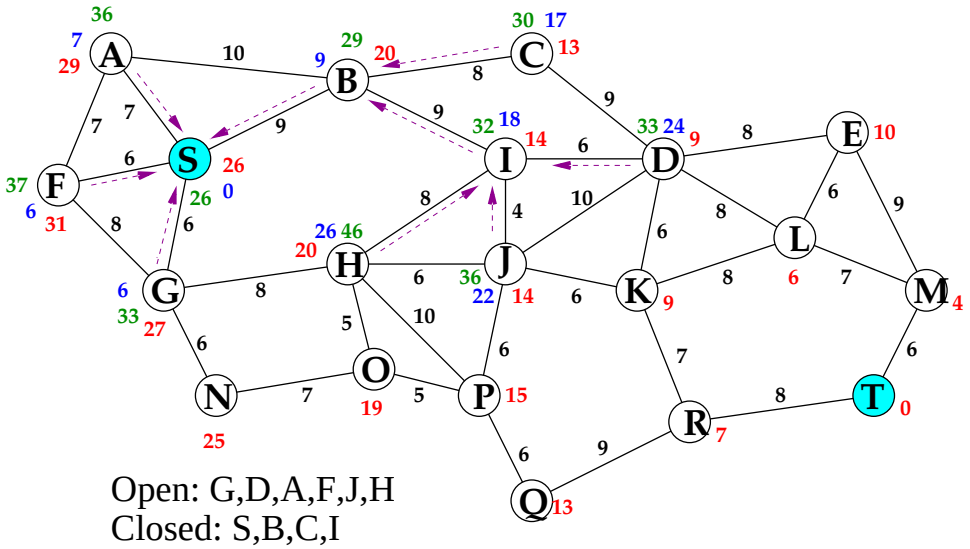
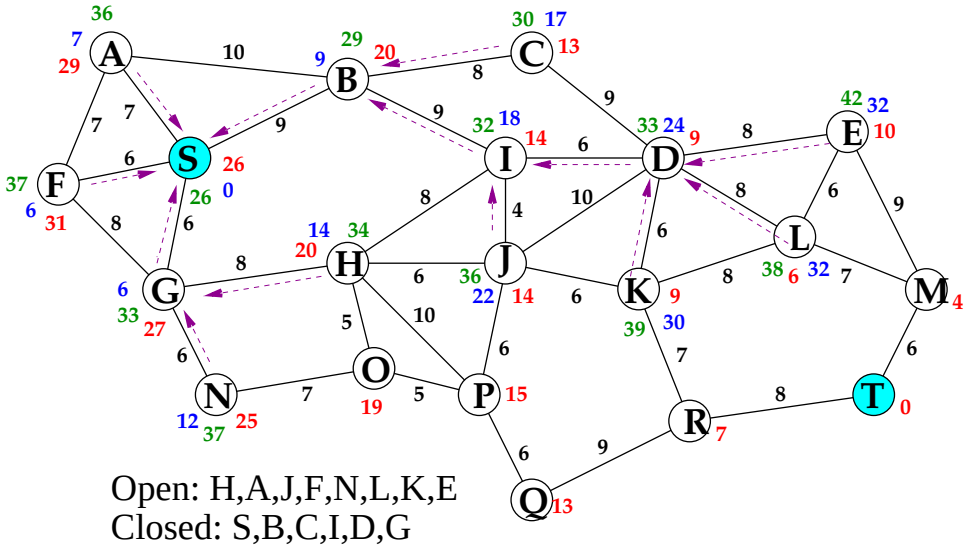


Figure F: $g(G) = g(D)$, hence they can be fully processed simultaneously. L, K, E, and N are partially processed, while $f(H)$ and $g(H)$ decrease. H jumps to the top of the minqueue.



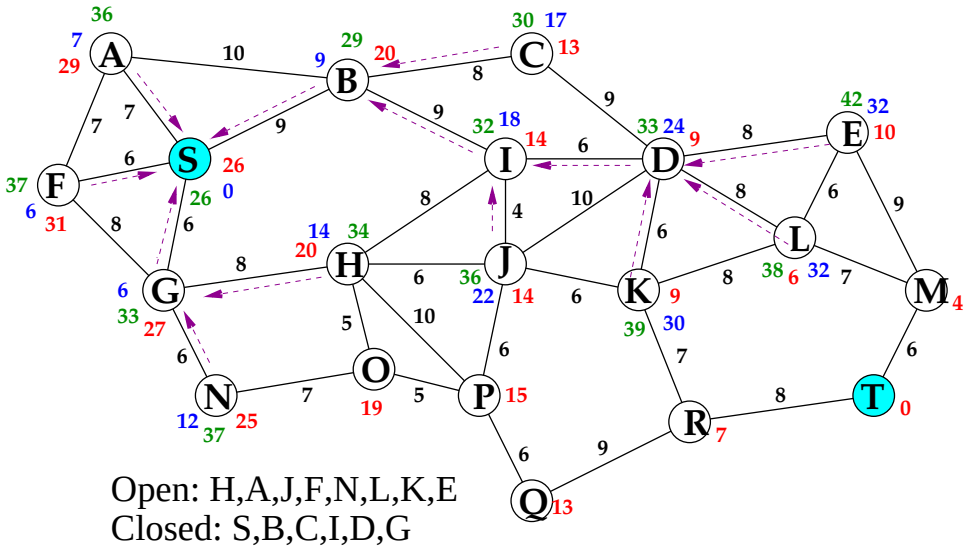
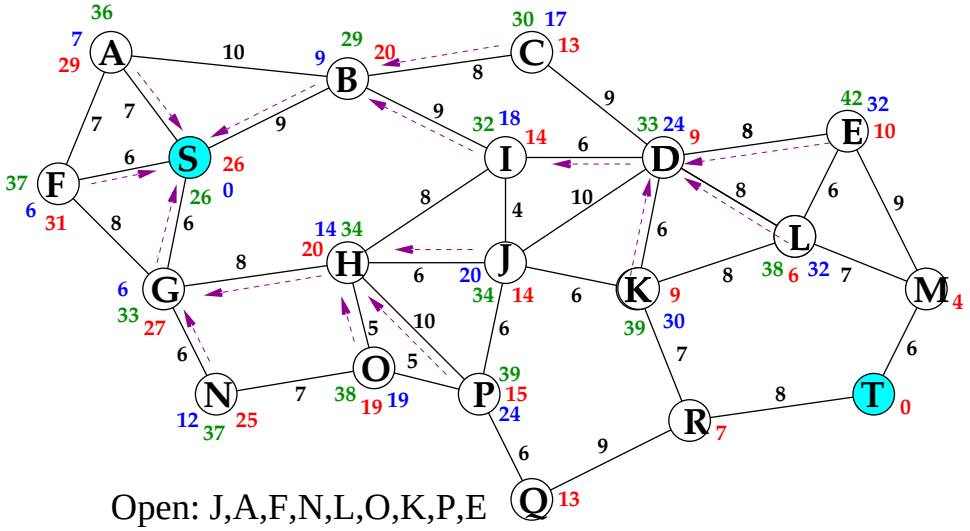


Figure G: H is processed. O and P are partially processed. while $f(J)$, $g(J)$ decrease, causing J to jump to the top of the minqueue.



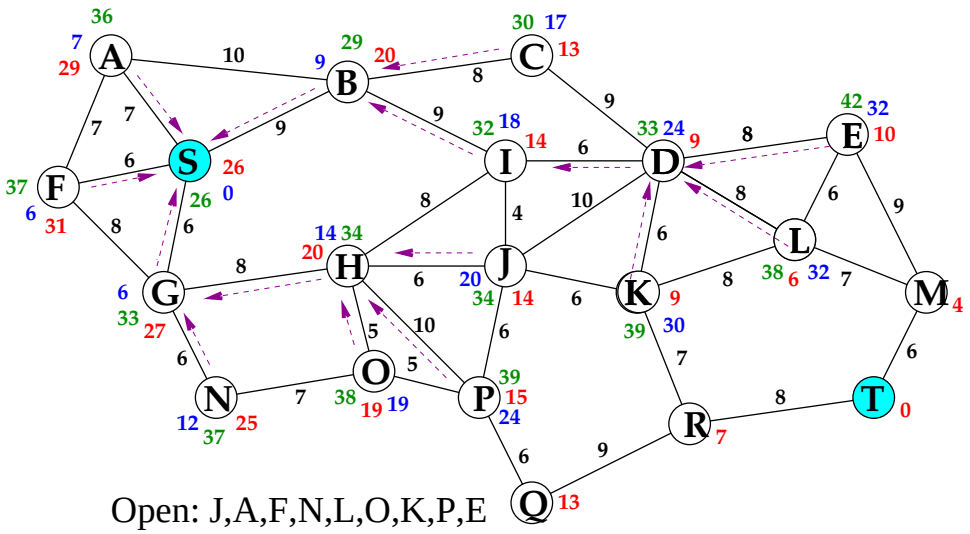
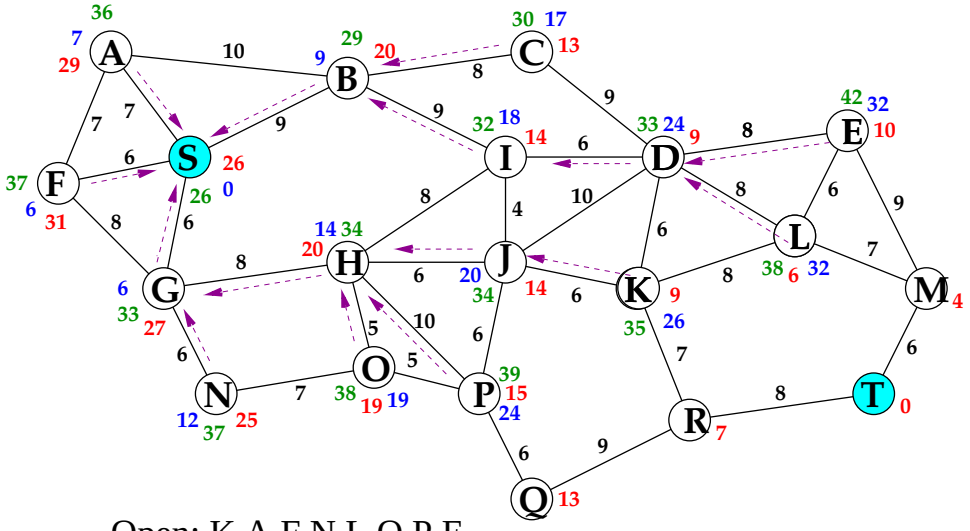
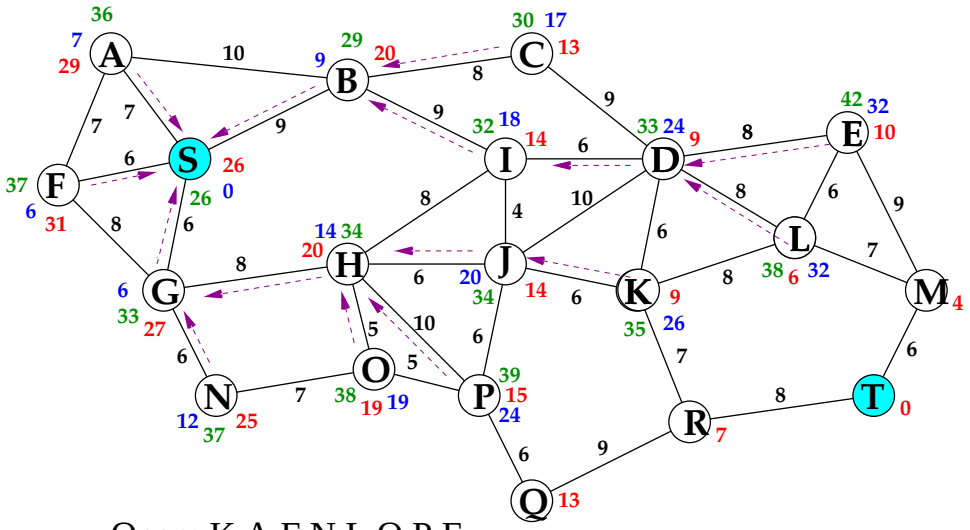


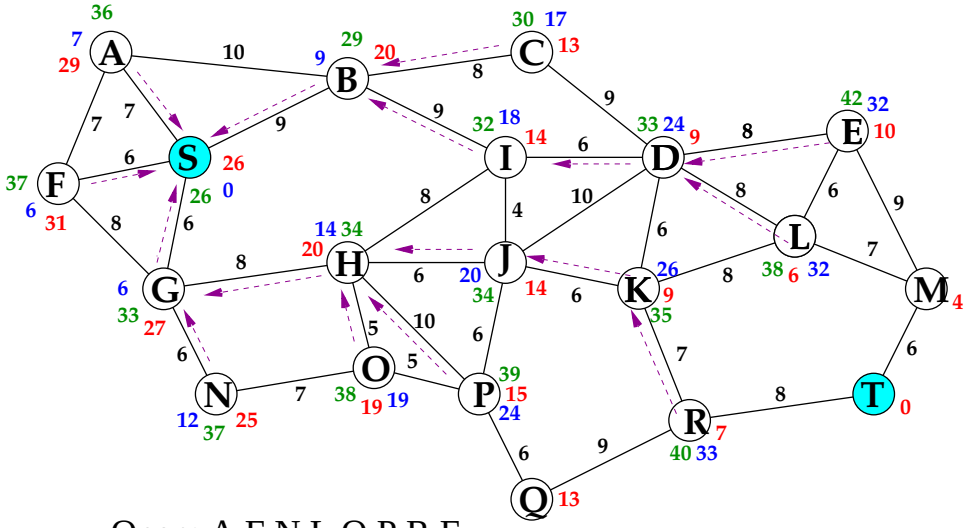
Figure H: J is fully processed. The figure does not change.



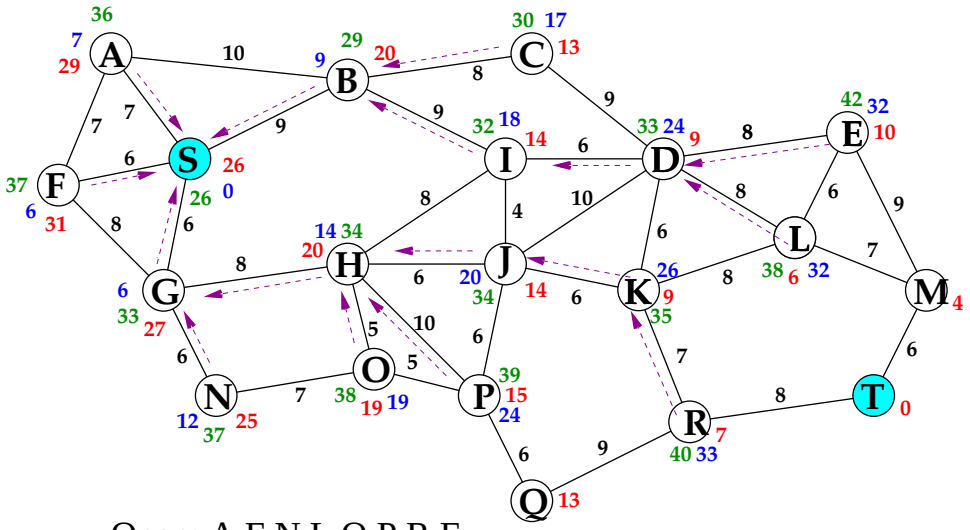


Open: K,A,F,N,L,O,P,E
Closed: S,B,C,I,D,G,H,J

Figure I: K is fully processed. R is partially processed.

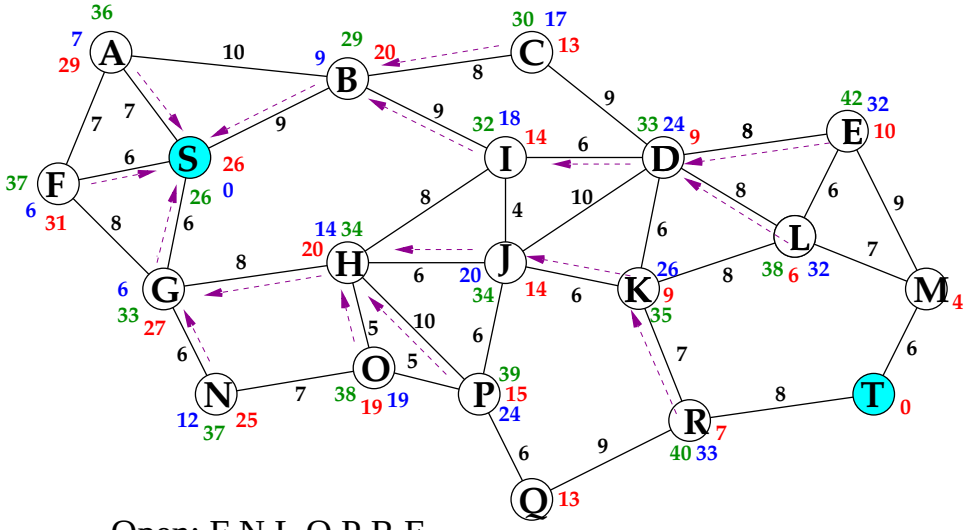


Open: A,F,N,L,O,P,R,E
Closed: S,B,C,I,D,G,H,J,K

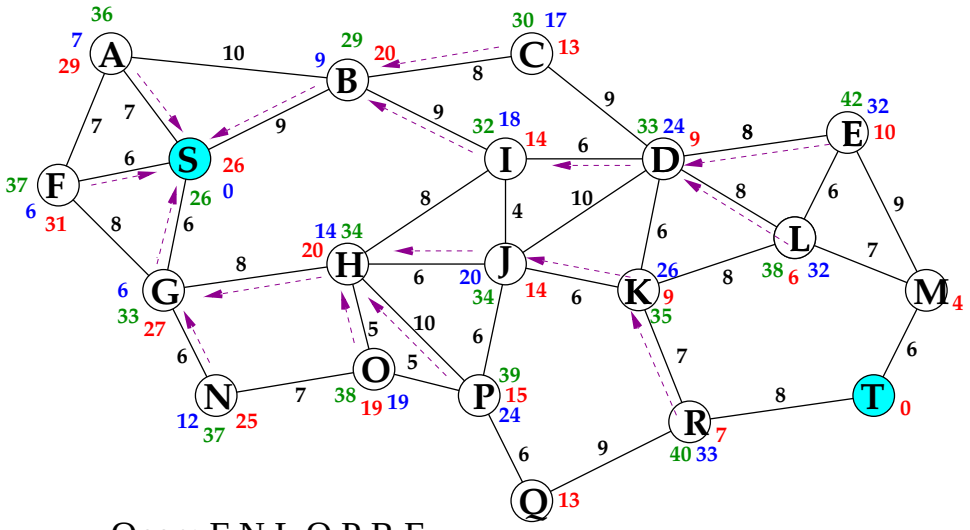


Open: A,F,N,L,O,P,R,E
Closed: S,B,C,I,D,G,H,J,K

Figure J: A is fully processed. No new vertices are visited. The figure does not change.

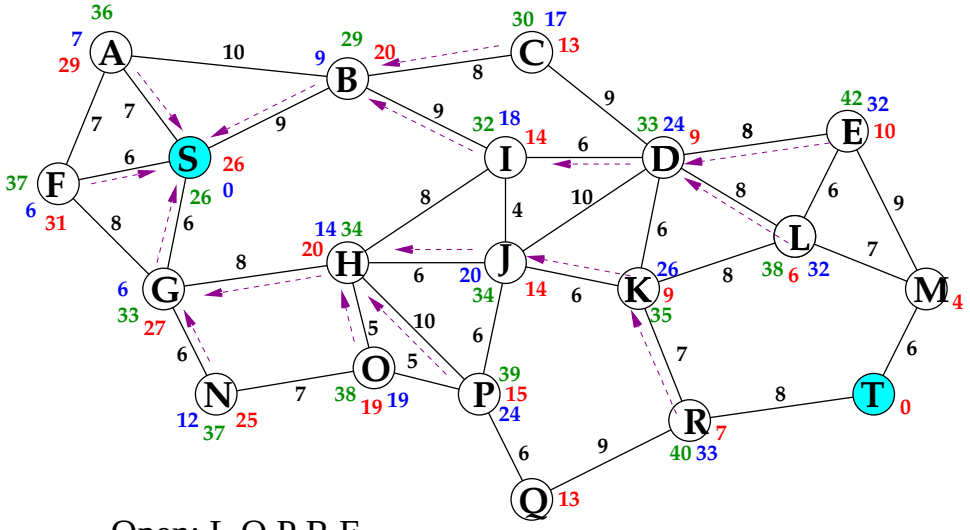


Open: F,N,L,O,P,R,E
Closed: S,B,C,I,D,G,H,J,K,A

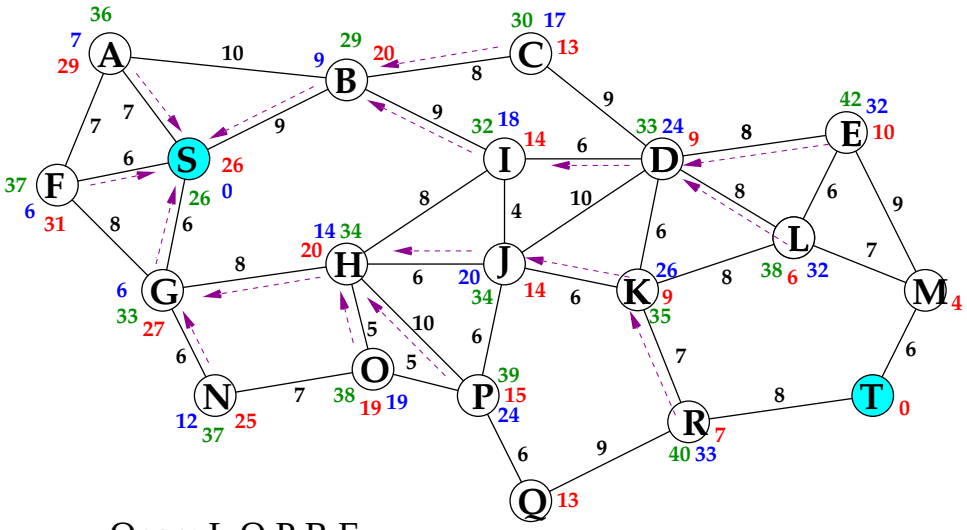


Open: F,N,L,O,P,R,E
Closed: S,B,C,I,D,G,H,J,K,A

Figure K: F and N are processed simultaneously, since $g(F) = g(N)$. No new vertices are visited. The figure does not change.

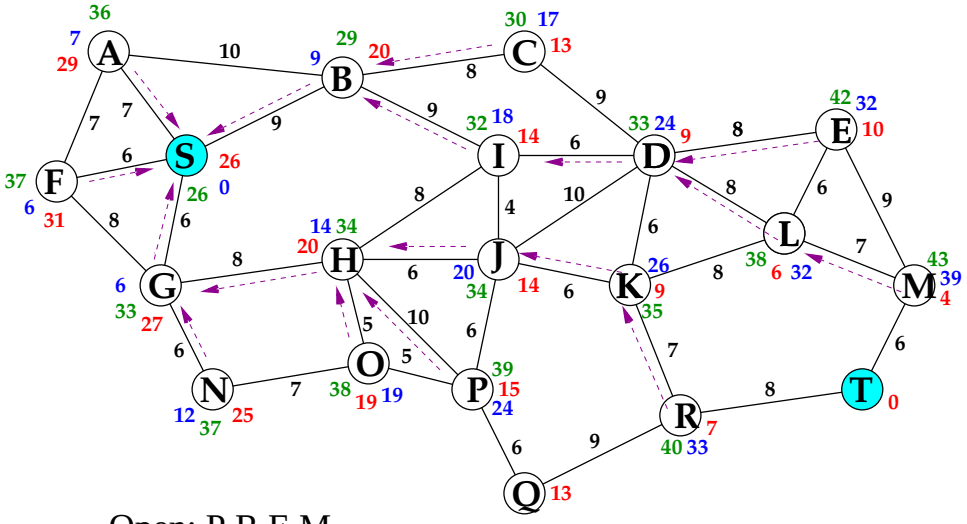


Open: L,O,P,R,E
Closed: S,B,C,I,D,G,H,J,K,A,F,N

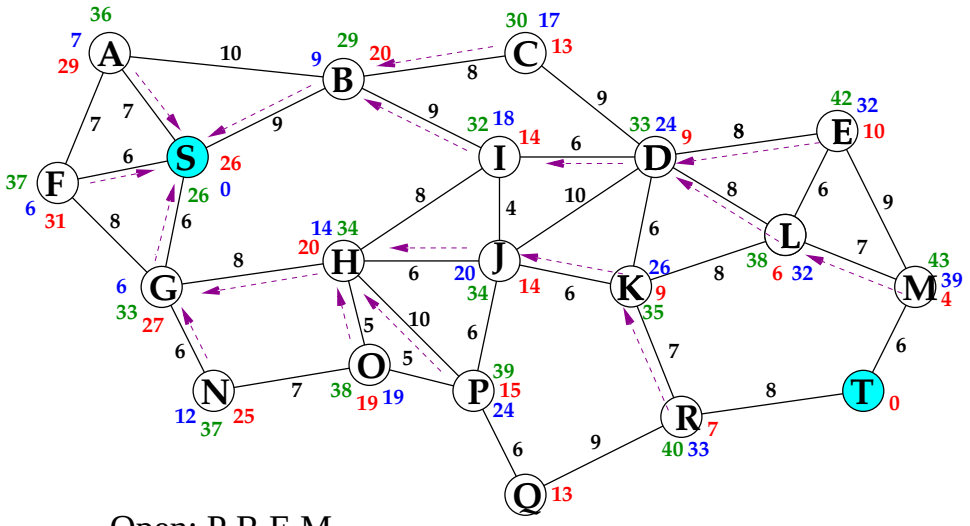


Open: L,O,P,R,E
Closed: S,B,C,I,D,G,H,J,K,A,F,N

Figure L: M is partially processed.



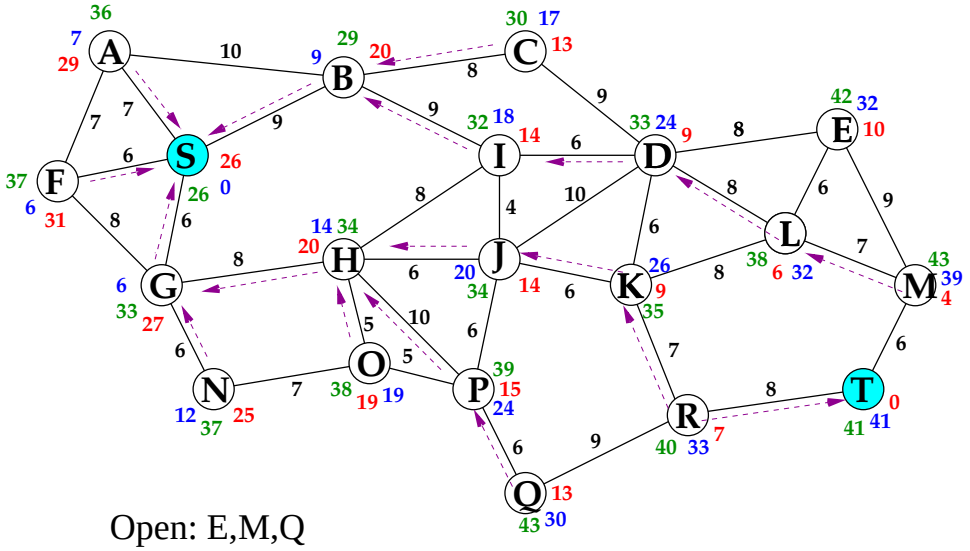
Open: P,R,E,M
Closed: S,B,C,I,D,G,H,J,K,A,F,N



Open: P,R,E,M

Closed: S,B,C,I,D,G,H,J,K,A,F,N

Figure M: P and R are processed, while Q and T are partially processed.



Open: E,M,Q

Closed: S,B,C,I,D,G,H,J,K,A,F,N,P,R

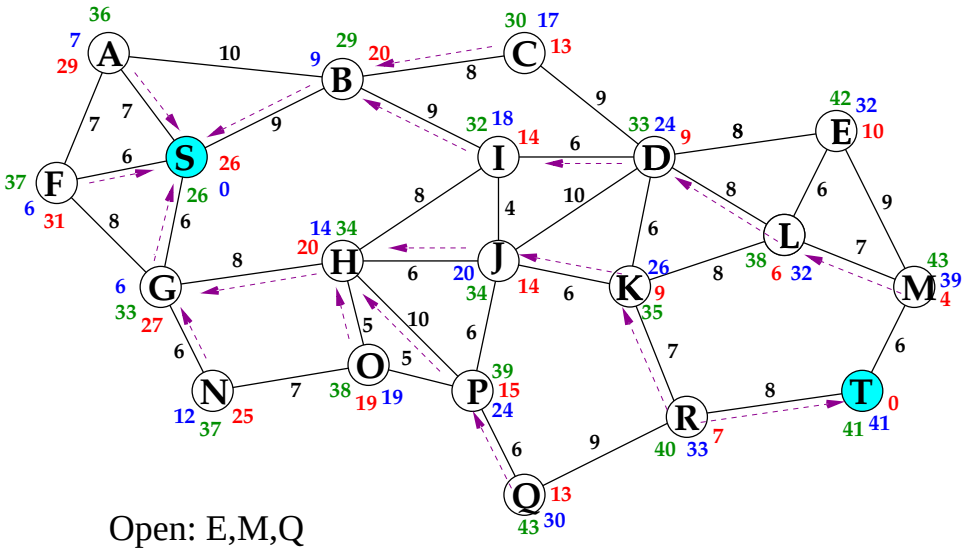


Figure N: T is fully processed, which terminates the A^* algorithm. The figure does not change. The shortest path from S to T is indicated.