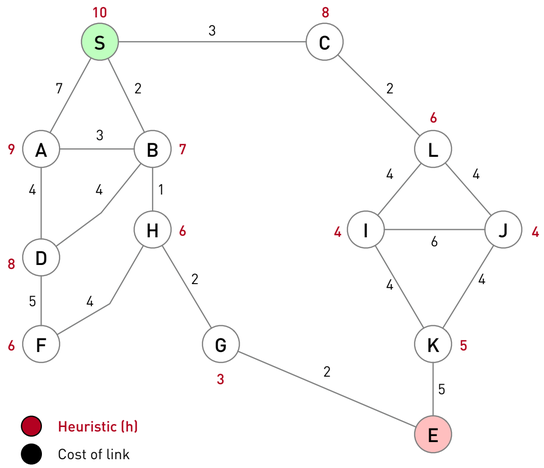
A* Algorithm

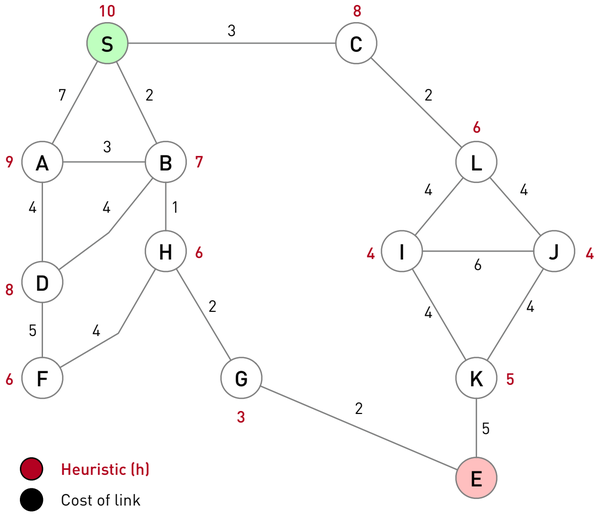
```
[1] g # search space, adjacency list but now including cost between nodes
[2] init # initial state
[3] goal # goal state
[4] q # list of tuples with five elements: terminal node, accumulated cost from init (f), heuristic
      # estimate to goal (h), total cost g = f+h and path to current node.
[5]   # initialise all tuples for every node, with empty path and cost set to infinite,
      # except for init that starts with cost heuristic(S) and [init] as its currently shortest path
    done = [] list of processed nodes # processed nodes end up here

[6] while q:
[7]     h = Head(q)
[8]     r = Rest(q)
[9]     if h[0] == goal:
[10]         exit # here we can exit but also alternatively just print path and continue
[11]    else:
[12]        e = Expand(h)
[13]    for node in e:
[14]        if accumulated cost plus the heuristic from init to e less than current cost in tuple for e:
[15]           Update tuple for e with new f cost, h cost, g = f+h and new path
[16]    done.append(h)
[17]    q = sort(e+r) #we combine all elements of the queue and sort from shortest to longest path
```




```
g = {
S: {(A,7),(B,2),(C,3)},
A: {(B,3),(D,4),(S,7)},
B: {(A,2),(D,4),(H,1),(S,2)},
C: {(L,3)},
…
}
```

A* Example



This is the head

Initialisation

S	0	10	10
[S]			

A	∞	0	∞
[]			

G	∞	0	∞
[]			

B	∞	0	∞
[]			

H	∞	0	∞
[]			

C	∞	0	∞
[]			

I	∞	0	∞
[]			

D	∞	0	∞
[]			

J	∞	0	∞
[]			

E	∞	0	∞
[]			

K	∞	0	∞
[]			

F	∞	0	∞
[]			

L	∞	0	∞
[]			

Iteration 1

S	0	10	10
[S]			

B	2	7	9
[S,B]			

G	∞	0	∞
[]			

C	3	8	11
[S,C]			

H	∞	0	∞
[]			

A	7	9	16
[S,A]			

I	∞	0	∞
[]			

D	∞	0	∞
[]			

J	∞	0	∞
[]			

E	∞	0	∞
[]			

K	∞	0	∞
[]			

F	∞	0	∞
[]			

L	∞	0	∞
[]			

Iteration 2

B	2	7	9
[S,B]			

H	3	6	9
[S,B,H]			

G	∞	0	∞
[]			

C	3	8	11
[S,C]			

D	6	8	14
[S,B,D]			

L	∞	0	∞
[]			

A	5	9	14
[S,B,A]			

I	∞	0	∞
[]			

E	∞	0	∞
[]			

J	∞	0	∞
[]			

F	∞	0	∞
[]			

K	∞	0	∞
[]			

Iteration 3

H	3	6	9
[S,B,H]			

G	5	3	8
[S,B,H,G]			

C	3	8	11
[S,C]			

D	6	8	14
[S,B,D]			

L	∞	0	∞
[]			

A	5	9	14
[S,B,A]			

I	∞	0	∞
[]			

F	7	6	14
[S,B,H,F]			

J	∞	0	∞
[]			

K	∞	0	∞
[]			

E	∞	0	∞
[]			

Iteration 4

G	5	3	8
[S,B,H,G]			

E	7	3	10
[S,B,H,G,E]			

C	3	8	11
[S,C]			

D	6	8	14
[S,B,D]			

A	5	9	14
[S,B,A]			

L	∞	0	∞
[]			

F	7	6	14
[S,B,H,F]			

I	∞	0	∞
[]			

K	∞	0	∞
[]			

J	∞	0	∞
[]			

Node | Cost so far (f) | Heuristic to goal (h) | Combined cost g = f+h

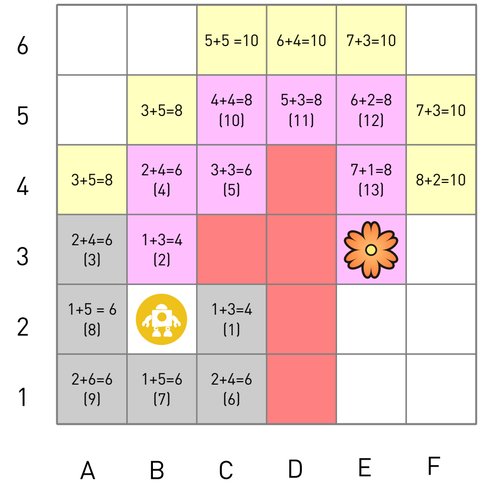
S	0	10	10
[S]			

Open path to node

During iteration 5, Node E is head of the Q
and we find the shortest path:
S,B,H,G,E

A Different Example

- **This is a grid world, with obstacles**
- **Robot starts at B2, must get flower at E3**
- **Robot cannot walk into obstacles (like D2)**
- **We will use the Manhattan distance as heuristic**
- **Robot can move straight to N, S, E, W (not diagonally)**



We can now expand any of the two paths with total value g=4

There are various paths with g=6. Prefer those with smallest heuristic values