Schema dell'algoritmo A*

- L'algoritmo fa uso di una funzione euristica f (n) = g (n) + h (n).
- Il costo vero del cammino risolutore ottimo (nel caso esista) passante per n è indicato come f *(n) = g *(n) + h *(n)
- La versione dell'algoritmo qui descritta non fa l'assunzione della monotonicità della funzione f.
- Lavora su grafo e non su albero.
- S rappresenta lo stato iniziale, GOALS contiene l'insieme degli stati finali (tipicamente uno solo). Inizialmente sia OPEN che CLOSED sono liste vuote

```
1.
      put S in OPEN
2.
      evaluate f(S)
3.
      found := false
      while ((OPEN \neq NIL) and (not found)) {
      // finché ho nodi da esplorare e non ho trovato l'obiettivo
             select n OPEN | f(n) is min // seleziona un nodo promettenre da esplorare
5.
6.
             remove n from OPEN
7.
             put n in CLOSED
8.
             if (n ∈ GOALS) // se è un goal ho finito
9.
                then found := true
                else { // se non è un goal
10.
11.
                   newnodes := expand(n) // costruisci i suoi successori
                   for each n<sub>i</sub> ∈ newnodes do { // per ciascun successore
12.
                       fprov(n_i) := g(n) + cost(operator(n, n_i)) + h(n_i) // valutalo
13.
14.
                       if (n_i \oplus OPEN \text{ and } n_i \oplus CLOSED) // se è un nodo nuovo
15.
                       then {
16.
                               put n<sub>i</sub>in OPEN // aggiungilo a quelli da considerare
17.
                               f(n_i) := fprov(n_i) // valutalo
                               father(n<sub>i</sub>) := n // segnati il genitore
18.
19.
20.
                       else // se il nodo era già stato attraversato
21.
                            if (fprov(n_i) < f(n_i)) // il percorso è migliore?
22.
                               then { // se sì aggiorna le info
23.
                                      f(n_i) := fprov(n_i)
24.
                                      father(n_i) := n
25.
                                      if (n_i \in CLOSED) // se il nodo era chiuso
26.
                                         then { // riaprilo
27.
                                                remove n<sub>i</sub> from CLOSED
```

```
28.
                                                     \boldsymbol{put} \ n_i \ \boldsymbol{in} \ \mathsf{OPEN}
                                                  } // then
29.
30.
                                       } // then
                       } // foreach
31.
                      } // else
32.
         }//while
33.
          if (found) then {
34.
                              solution := findpath(n)
35.
                              print (solution)
36.
37.
                    else print ("no solution exists")
38.
```