

function BEST-FIRST-SEARCH(*problem*, *f*) **returns** a solution node or *failure*

node \leftarrow NODE(STATE=*problem*.INITIAL)

frontier \leftarrow a priority queue ordered by *f*, with *node* as an element

reached \leftarrow a lookup table, with one entry with key *problem*.INITIAL and value *node*

while not IS-EMPTY(*frontier*) **do**

node \leftarrow POP(*frontier*)

if *problem*.IS-GOAL(*node*.STATE) **then return** *node*

for each *child* **in** EXPAND(*problem*, *node*) **do**

s \leftarrow *child*.STATE

if *s* is not in *reached* **or** *child*.PATH-COST < *reached*[*s*].PATH-COST **then**

reached[*s*] \leftarrow *child*

 add *child* to *frontier*

return *failure*

function EXPAND(*problem*, *node*) **yields** nodes

s \leftarrow *node*.STATE

for each *action* **in** *problem*.ACTIONS(*s*) **do**

s' \leftarrow *problem*.RESULT(*s*, *action*)

cost \leftarrow *node*.PATH-COST + *problem*.ACTION-COST(*s*, *action*, *s'*)

yield NODE(STATE=*s'*, PARENT=*node*, ACTION=*action*, PATH-COST=*cost*)