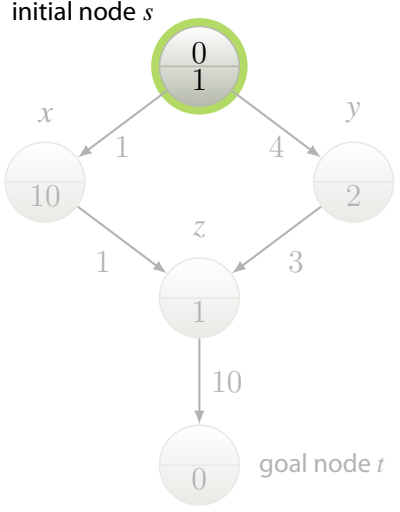
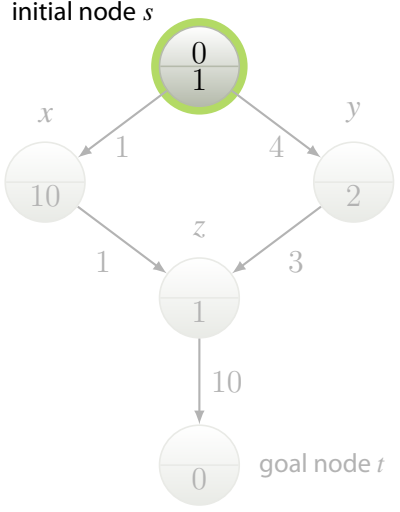
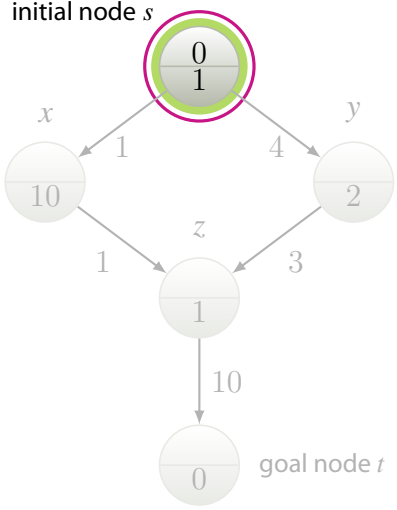


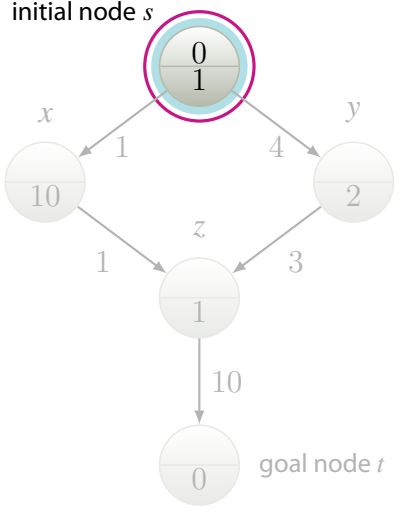

```
1  function A*(s)
2      OPEN ← new PriorityQueue_f
3      g[s] ← 0; f[s] ← h(s)
4      Insert_f(OPEN, s)
5      CLOSED ← ∅
6      loop do
7          if IsEmpty (OPEN) then return "failure"
8          v ← DeleteMin_f(OPEN)
9          CLOSED ← CLOSED ∪ {v}
10         if IsGoal(v) then return Solution(v, s)
11         Expand(v)
12 procedure Expand(v)
13     foreach u ∈ Succ(v) do
14         if u ∉ OPEN ∪ CLOSED then
15             g[u] ← g[v] + c(v, u)
16             f[u] ← g[u] + h(u)
17             Parent[u] ← v
18             Insert_f(OPEN, u)
19         else if u ∈ OPEN then
20             if g[v] + c(v, u) < g[v] then
21                 g[u] ← g[v] + c(v, u)
22                 f[u] ← g[u] + h(u)
23                 Parent[u] ← v
24         else                                  # if u ∈ CLOSED
25             if g[v] + c(v, u) < g[u] then
26                 g[u] ← g[v] + c(v, u)
27                 f[u] ← g[u] + h(u)
28                 Parent[u] ← v
29                 CLOSED ← CLOSED\{u}
30                 Insert_f(OPEN, u)
```



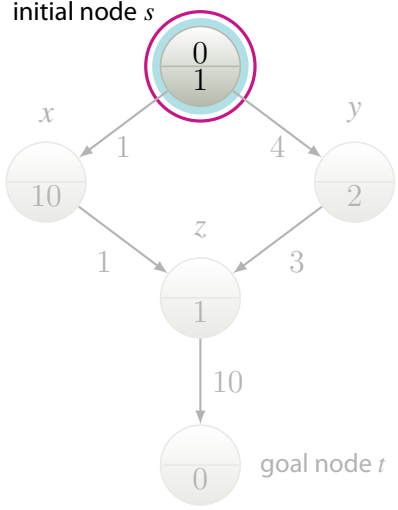

```
1  function A*(s)
2      OPEN ← new PriorityQueue_f
3      g[s] ← 0; f[s] ← h(s)
4      Insert_f(OPEN, s)
5      CLOSED ← ∅
6      loop do
7          if IsEmpty (OPEN) then return "failure"
8          v ← DeleteMin_f(OPEN)
9          CLOSED ← CLOSED ∪ {v}
10         if IsGoal(v) then return Solution(v, s)
11         Expand(v)
12 procedure Expand(v)
13     foreach u ∈ Succ(v) do
14         if u ∉ OPEN ∪ CLOSED then
15             g[u] ← g[v] + c(v, u)
16             f[u] ← g[u] + h(u)
17             Parent[u] ← v
18             Insert_f(OPEN, u)
19         else if u ∈ OPEN then
20             if g[v] + c(v, u) < g[v] then
21                 g[u] ← g[v] + c(v, u)
22                 f[u] ← g[u] + h(u)
23                 Parent[u] ← v
24         else                                  # if u ∈ CLOSED
25             if g[v] + c(v, u) < g[u] then
26                 g[u] ← g[v] + c(v, u)
27                 f[u] ← g[u] + h(u)
28                 Parent[u] ← v
29                 CLOSED ← CLOSED\{u}
30                 Insert_f(OPEN, u)
```



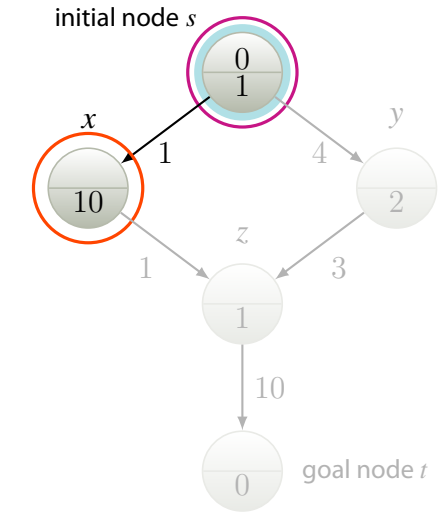

```
1  function A*(s)
2      OPEN ← new PriorityQueue_f
3      g[s] ← 0; f[s] ← h(s)
4      Insert_f(OPEN, s)
5      CLOSED ← ∅
6      loop do
7          if IsEmpty (OPEN) then return "failure"
8          v ← DeleteMin_f(OPEN)
9          CLOSED ← CLOSED ∪ {v}
10         if IsGoal(v) then return Solution(v, s)
11         Expand(v)
12 procedure Expand(v)
13     foreach u ∈ Succ(v) do
14         if u ∉ OPEN ∪ CLOSED then
15             g[u] ← g[v] + c(v, u)
16             f[u] ← g[u] + h(u)
17             Parent[u] ← v
18             Insert_f(OPEN, u)
19         else if u ∈ OPEN then
20             if g[v] + c(v, u) < g[v] then
21                 g[u] ← g[v] + c(v, u)
22                 f[u] ← g[u] + h(u)
23                 Parent[u] ← v
24         else                                  # if u ∈ CLOSED
25             if g[v] + c(v, u) < g[u] then
26                 g[u] ← g[v] + c(v, u)
27                 f[u] ← g[u] + h(u)
28                 Parent[u] ← v
29                 CLOSED ← CLOSED\{u}
30                 Insert_f(OPEN, u)
```

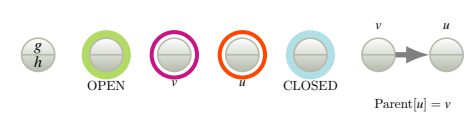



```
1  function A*(s)
2      OPEN ← new PriorityQueue_f
3      g[s] ← 0; f[s] ← h(s)
4      Insert_f(OPEN, s)
5      CLOSED ← ∅
6      loop do
7          if IsEmpty(OPEN) then return "failure"
8          v ← DeleteMin_f(OPEN)
9          CLOSED ← CLOSED ∪ {v}
10         if IsGoal(v) then return Solution(v, s)
11         Expand(v)
12 procedure Expand(v)
13     foreach u ∈ Succ(v) do
14         if u ∉ OPEN ∪ CLOSED then
15             g[u] ← g[v] + c(v, u)
16             f[u] ← g[u] + h(u)
17             Parent[u] ← v
18             Insert_f(OPEN, u)
19         else if u ∈ OPEN then
20             if g[v] + c(v, u) < g[v] then
21                 g[u] ← g[v] + c(v, u)
22                 f[u] ← g[u] + h(u)
23                 Parent[u] ← v
24         else                                  # if u ∈ CLOSED
25             if g[v] + c(v, u) < g[u] then
26                 g[u] ← g[v] + c(v, u)
27                 f[u] ← g[u] + h(u)
28                 Parent[u] ← v
29                 CLOSED ← CLOSED\{u}
30                 Insert_f(OPEN, u)
```

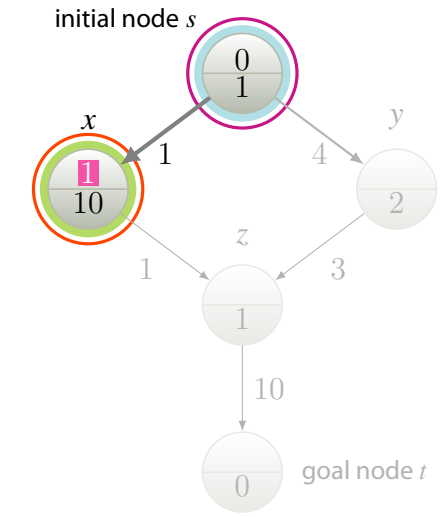


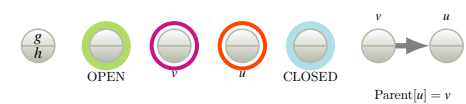

```
1  function A*(s)
2      OPEN ← new PriorityQueue_f
3      g[s] ← 0; f[s] ← h(s)
4      Insert_f(OPEN, s)
5      CLOSED ← ∅
6      loop do
7          if IsEmpty(OPEN) then return "failure"
8          v ← DeleteMin_f(OPEN)
9          CLOSED ← CLOSED ∪ {v}
10         if IsGoal(v) then return Solution(v, s)
11         Expand(v)
12 procedure Expand(v)
13     foreach u ∈ Succ(v) do
14         if u ∉ OPEN ∪ CLOSED then
15             g[u] ← g[v] + c(v, u)
16             f[u] ← g[u] + h(u)
17             Parent[u] ← v
18             Insert_f(OPEN, u)
19         else if u ∈ OPEN then
20             if g[v] + c(v, u) < g[v] then
21                 g[u] ← g[v] + c(v, u)
22                 f[u] ← g[u] + h(u)
23                 Parent[u] ← v
24         else                                  # if u ∈ CLOSED
25             if g[v] + c(v, u) < g[u] then
26                 g[u] ← g[v] + c(v, u)
27                 f[u] ← g[u] + h(u)
28                 Parent[u] ← v
29                 CLOSED ← CLOSED\{u}
30                 Insert_f(OPEN, u)
```



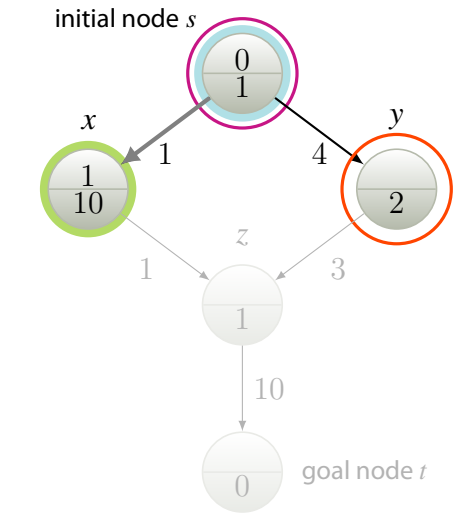


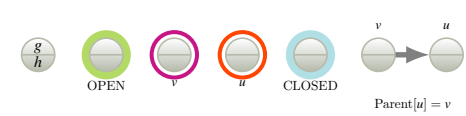

```
1  function A*(s)
2      OPEN ← new PriorityQueue_f
3      g[s] ← 0; f[s] ← h(s)
4      Insert_f(OPEN, s)
5      CLOSED ← ∅
6      loop do
7          if IsEmpty(OPEN) then return "failure"
8          v ← DeleteMin_f(OPEN)
9          CLOSED ← CLOSED ∪ {v}
10         if IsGoal(v) then return Solution(v, s)
11         Expand(v)
12 procedure Expand(v)
13     foreach u ∈ Succ(v) do
14         if u ∉ OPEN ∪ CLOSED then
15             g[u] ← g[v] + c(v, u)
16             f[u] ← g[u] + h(u)
17             Parent[u] ← v
18             Insert_f(OPEN, u)
19         else if u ∈ OPEN then
20             if g[v] + c(v, u) < g[v] then
21                 g[u] ← g[v] + c(v, u)
22                 f[u] ← g[u] + h(u)
23                 Parent[u] ← v
24         else                                  # if u ∈ CLOSED
25             if g[v] + c(v, u) < g[u] then
26                 g[u] ← g[v] + c(v, u)
27                 f[u] ← g[u] + h(u)
28                 Parent[u] ← v
29                 CLOSED ← CLOSED\{u}
30                 Insert_f(OPEN, u)
```



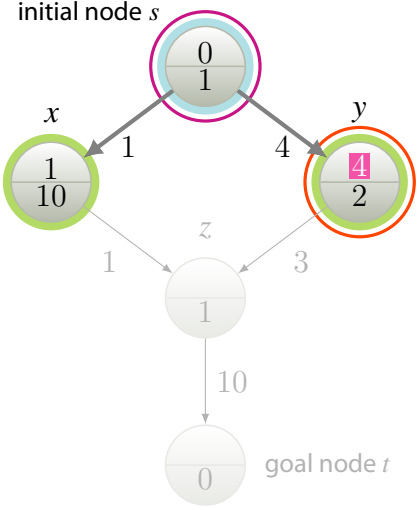


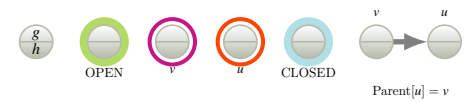

```
1  function A*(s)
2      OPEN ← new PriorityQueue_f
3      g[s] ← 0; f[s] ← h(s)
4      Insert_f(OPEN, s)
5      CLOSED ← ∅
6      loop do
7          if IsEmpty (OPEN) then return "failure"
8          v ← DeleteMin_f(OPEN)
9          CLOSED ← CLOSED ∪ {v}
10         if IsGoal(v) then return Solution(v, s)
11         Expand(v)
12 procedure Expand(v)
13     foreach u ∈ Succ(v) do
14         if u ∉ OPEN ∪ CLOSED then
15             g[u] ← g[v] + c(v, u)
16             f[u] ← g[u] + h(u)
17             Parent[u] ← v
18             Insert_f(OPEN, u)
19         else if u ∈ OPEN then
20             if g[v] + c(v, u) < g[v] then
21                 g[u] ← g[v] + c(v, u)
22                 f[u] ← g[u] + h(u)
23                 Parent[u] ← v
24         else                                  # if u ∈ CLOSED
25             if g[v] + c(v, u) < g[u] then
26                 g[u] ← g[v] + c(v, u)
27                 f[u] ← g[u] + h(u)
28                 Parent[u] ← v
29                 CLOSED ← CLOSED\{u}
30                 Insert_f(OPEN, u)
```



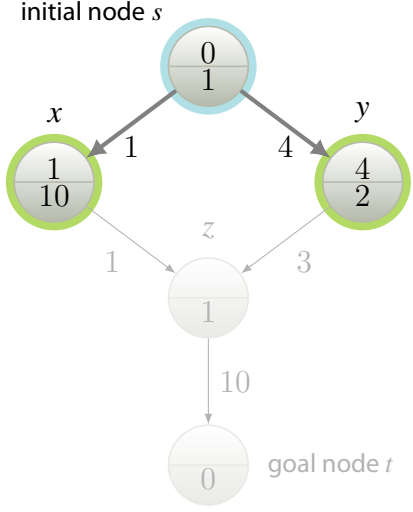



```
1  function A*(s)
2      OPEN ← new PriorityQueue_f
3      g[s] ← 0; f[s] ← h(s)
4      Insert_f(OPEN, s)
5      CLOSED ← ∅
6      loop do
7          if IsEmpty(OPEN) then return "failure"
8          v ← DeleteMin_f(OPEN)
9          CLOSED ← CLOSED ∪ {v}
10         if IsGoal(v) then return Solution(v, s)
11         Expand(v)
12 procedure Expand(v)
13     foreach u ∈ Succ(v) do
14         if u ∉ OPEN ∪ CLOSED then
15             g[u] ← g[v] + c(v, u)
16             f[u] ← g[u] + h(u)
17             Parent[u] ← v
18             Insert_f(OPEN, u)
19         else if u ∈ OPEN then
20             if g[v] + c(v, u) < g[v] then
21                 g[u] ← g[v] + c(v, u)
22                 f[u] ← g[u] + h(u)
23                 Parent[u] ← v
24         else                                  # if u ∈ CLOSED
25             if g[v] + c(v, u) < g[u] then
26                 g[u] ← g[v] + c(v, u)
27                 f[u] ← g[u] + h(u)
28                 Parent[u] ← v
29                 CLOSED ← CLOSED\{u}
30                 Insert_f(OPEN, u)
```

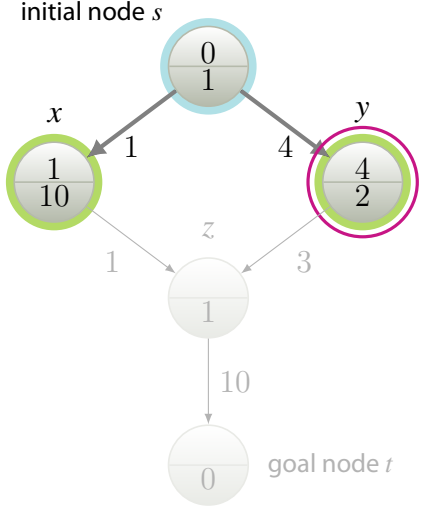




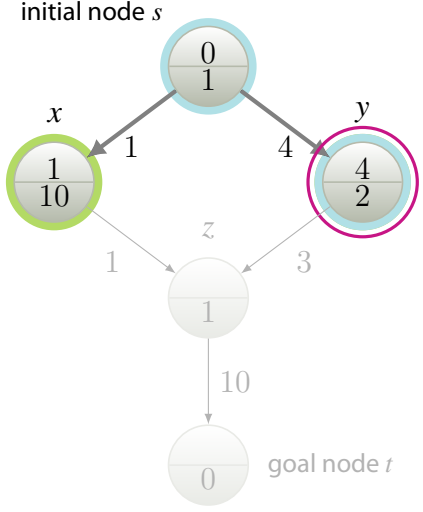

```
1  function A*(s)
2      OPEN ← new PriorityQueue_f
3      g[s] ← 0; f[s] ← h(s)
4      Insert_f(OPEN, s)
5      CLOSED ← ∅
6      loop do
7          if IsEmpty (OPEN) then return "failure"
8          v ← DeleteMin_f(OPEN)
9          CLOSED ← CLOSED ∪ {v}
10         if IsGoal(v) then return Solution(v, s)
11         Expand(v)
12 procedure Expand(v)
13     foreach u ∈ Succ(v) do
14         if u ∉ OPEN ∪ CLOSED then
15             g[u] ← g[v] + c(v, u)
16             f[u] ← g[u] + h(u)
17             Parent[u] ← v
18             Insert_f(OPEN, u)
19         else if u ∈ OPEN then
20             if g[v] + c(v, u) < g[v] then
21                 g[u] ← g[v] + c(v, u)
22                 f[u] ← g[u] + h(u)
23                 Parent[u] ← v
24         else                                  # if u ∈ CLOSED
25             if g[v] + c(v, u) < g[u] then
26                 g[u] ← g[v] + c(v, u)
27                 f[u] ← g[u] + h(u)
28                 Parent[u] ← v
29                 CLOSED ← CLOSED\{u}
30                 Insert_f(OPEN, u)
```



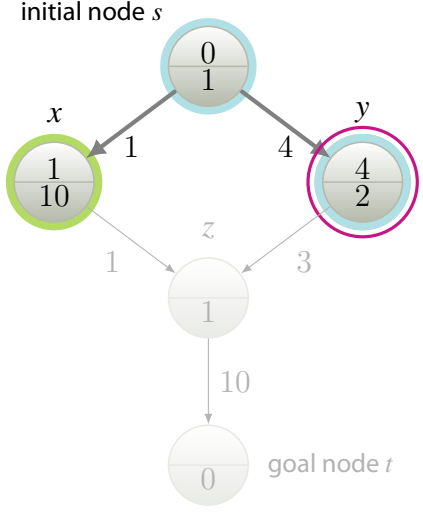

```
1  function A*(s)
2      OPEN ← new PriorityQueue_f
3      g[s] ← 0; f[s] ← h(s)
4      Insert_f(OPEN, s)
5      CLOSED ← ∅
6      loop do
7          if IsEmpty (OPEN) then return "failure"
8          v ← DeleteMin_f(OPEN)
9          CLOSED ← CLOSED ∪ {v}
10         if IsGoal(v) then return Solution(v, s)
11         Expand(v)
12 procedure Expand(v)
13     foreach u ∈ Succ(v) do
14         if u ∉ OPEN ∪ CLOSED then
15             g[u] ← g[v] + c(v, u)
16             f[u] ← g[u] + h(u)
17             Parent[u] ← v
18             Insert_f(OPEN, u)
19         else if u ∈ OPEN then
20             if g[v] + c(v, u) < g[v] then
21                 g[u] ← g[v] + c(v, u)
22                 f[u] ← g[u] + h(u)
23                 Parent[u] ← v
24         else                                  # if u ∈ CLOSED
25             if g[v] + c(v, u) < g[u] then
26                 g[u] ← g[v] + c(v, u)
27                 f[u] ← g[u] + h(u)
28                 Parent[u] ← v
29                 CLOSED ← CLOSED\{u}
30                 Insert_f(OPEN, u)
```



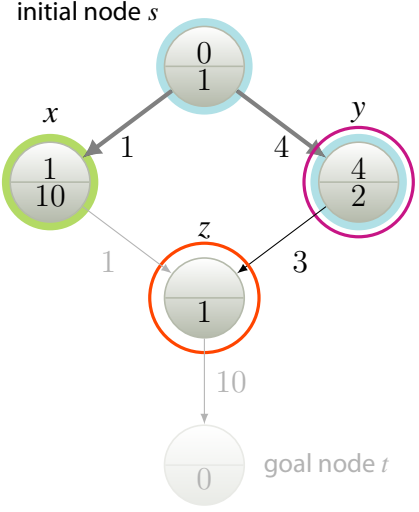

```
1  function A*(s)
2      OPEN ← new PriorityQueue_f
3      g[s] ← 0; f[s] ← h(s)
4      Insert_f(OPEN, s)
5      CLOSED ← ∅
6      loop do
7          if IsEmpty (OPEN) then return "failure"
8          v ← DeleteMin_f(OPEN)
9          CLOSED ← CLOSED ∪ {v}
10         if IsGoal(v) then return Solution(v, s)
11         Expand(v)
12 procedure Expand(v)
13     foreach u ∈ Succ(v) do
14         if u ∉ OPEN ∪ CLOSED then
15             g[u] ← g[v] + c(v, u)
16             f[u] ← g[u] + h(u)
17             Parent[u] ← v
18             Insert_f(OPEN, u)
19         else if u ∈ OPEN then
20             if g[v] + c(v, u) < g[v] then
21                 g[u] ← g[v] + c(v, u)
22                 f[u] ← g[u] + h(u)
23                 Parent[u] ← v
24         else                                  # if u ∈ CLOSED
25             if g[v] + c(v, u) < g[u] then
26                 g[u] ← g[v] + c(v, u)
27                 f[u] ← g[u] + h(u)
28                 Parent[u] ← v
29                 CLOSED ← CLOSED\{u}
30                 Insert_f(OPEN, u)
```



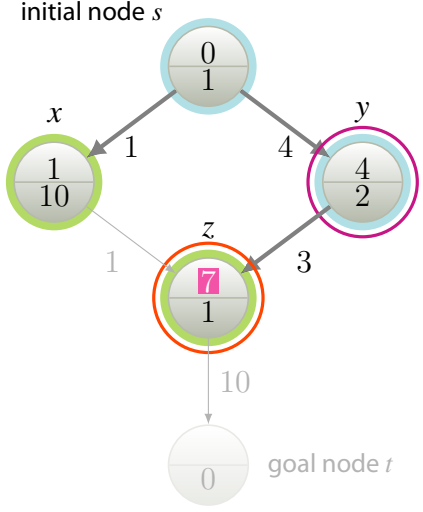

```
1  function A*(s)
2      OPEN ← new PriorityQueue_f
3      g[s] ← 0; f[s] ← h(s)
4      Insert_f(OPEN, s)
5      CLOSED ← ∅
6      loop do
7          if IsEmpty (OPEN) then return "failure"
8          v ← DeleteMin_f(OPEN)
9          CLOSED ← CLOSED ∪ {v}
10         if IsGoal(v) then return Solution(v, s)
11         Expand(v)
12 procedure Expand(v)
13     foreach u ∈ Succ(v) do
14         if u ∉ OPEN ∪ CLOSED then
15             g[u] ← g[v] + c(v, u)
16             f[u] ← g[u] + h(u)
17             Parent[u] ← v
18             Insert_f(OPEN, u)
19         else if u ∈ OPEN then
20             if g[v] + c(v, u) < g[v] then
21                 g[u] ← g[v] + c(v, u)
22                 f[u] ← g[u] + h(u)
23                 Parent[u] ← v
24         else                                  # if u ∈ CLOSED
25             if g[v] + c(v, u) < g[u] then
26                 g[u] ← g[v] + c(v, u)
27                 f[u] ← g[u] + h(u)
28                 Parent[u] ← v
29                 CLOSED ← CLOSED\{u}
30                 Insert_f(OPEN, u)
```



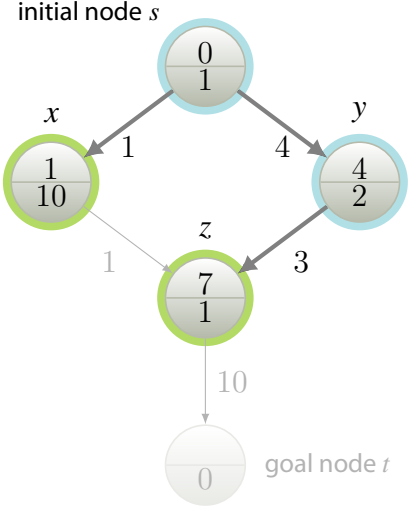

```
1  function A*(s)
2      OPEN ← new PriorityQueue_f
3      g[s] ← 0; f[s] ← h(s)
4      Insert_f(OPEN, s)
5      CLOSED ← ∅
6      loop do
7          if IsEmpty(OPEN) then return "failure"
8          v ← DeleteMin_f(OPEN)
9          CLOSED ← CLOSED ∪ {v}
10         if IsGoal(v) then return Solution(v, s)
11         Expand(v)
12 procedure Expand(v)
13     foreach u ∈ Succ(v) do
14         if u ∉ OPEN ∪ CLOSED then
15             g[u] ← g[v] + c(v, u)
16             f[u] ← g[u] + h(u)
17             Parent[u] ← v
18             Insert_f(OPEN, u)
19         else if u ∈ OPEN then
20             if g[v] + c(v, u) < g[v] then
21                 g[u] ← g[v] + c(v, u)
22                 f[u] ← g[u] + h(u)
23                 Parent[u] ← v
24         else                                  # if u ∈ CLOSED
25             if g[v] + c(v, u) < g[u] then
26                 g[u] ← g[v] + c(v, u)
27                 f[u] ← g[u] + h(u)
28                 Parent[u] ← v
29                 CLOSED ← CLOSED\{u}
30                 Insert_f(OPEN, u)
```



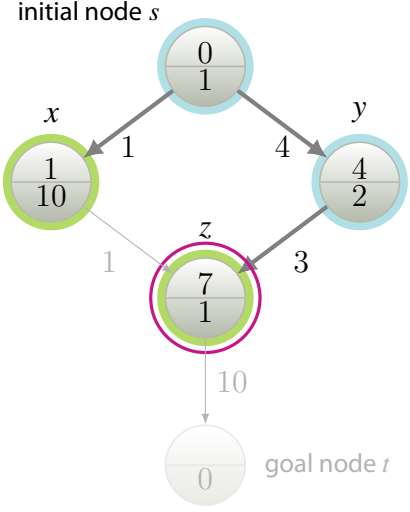

```
1  function A*(s)
2      OPEN ← new PriorityQueue_f
3      g[s] ← 0; f[s] ← h(s)
4      Insert_f(OPEN, s)
5      CLOSED ← ∅
6      loop do
7          if IsEmpty (OPEN) then return "failure"
8          v ← DeleteMin_f(OPEN)
9          CLOSED ← CLOSED ∪ {v}
10         if IsGoal(v) then return Solution(v, s)
11         Expand(v)
12 procedure Expand(v)
13     foreach u ∈ Succ(v) do
14         if u ∉ OPEN ∪ CLOSED then
15             g[u] ← g[v] + c(v, u)
16             f[u] ← g[u] + h(u)
17             Parent[u] ← v
18             Insert_f(OPEN, u)
19         else if u ∈ OPEN then
20             if g[v] + c(v, u) < g[v] then
21                 g[u] ← g[v] + c(v, u)
22                 f[u] ← g[u] + h(u)
23                 Parent[u] ← v
24         else                                  # if u ∈ CLOSED
25             if g[v] + c(v, u) < g[u] then
26                 g[u] ← g[v] + c(v, u)
27                 f[u] ← g[u] + h(u)
28                 Parent[u] ← v
29                 CLOSED ← CLOSED\{u}
30                 Insert_f(OPEN, u)
```



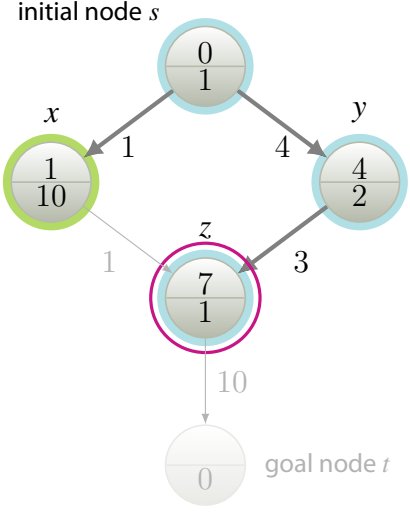

```
1  function A*(s)
2      OPEN ← new PriorityQueue_f
3      g[s] ← 0; f[s] ← h(s)
4      Insert_f(OPEN, s)
5      CLOSED ← ∅
6      loop do
7          if IsEmpty(OPEN) then return "failure"
8          v ← DeleteMin_f(OPEN)
9          CLOSED ← CLOSED ∪ {v}
10         if IsGoal(v) then return Solution(v, s)
11         Expand(v)
12 procedure Expand(v)
13     foreach u ∈ Succ(v) do
14         if u ∉ OPEN ∪ CLOSED then
15             g[u] ← g[v] + c(v, u)
16             f[u] ← g[u] + h(u)
17             Parent[u] ← v
18             Insert_f(OPEN, u)
19         else if u ∈ OPEN then
20             if g[v] + c(v, u) < g[v] then
21                 g[u] ← g[v] + c(v, u)
22                 f[u] ← g[u] + h(u)
23                 Parent[u] ← v
24         else                                  # if u ∈ CLOSED
25             if g[v] + c(v, u) < g[u] then
26                 g[u] ← g[v] + c(v, u)
27                 f[u] ← g[u] + h(u)
28                 Parent[u] ← v
29                 CLOSED ← CLOSED\{u}
30                 Insert_f(OPEN, u)
```



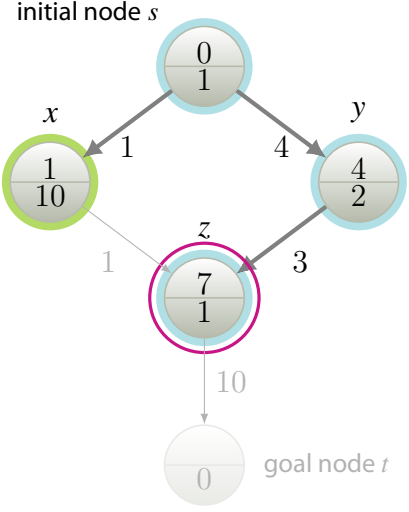

```
1  function A*(s)
2      OPEN ← new PriorityQueue_f
3      g[s] ← 0; f[s] ← h(s)
4      Insert_f(OPEN, s)
5      CLOSED ← ∅
6      loop do
7          if IsEmpty (OPEN) then return "failure"
8          v ← DeleteMin_f(OPEN)
9          CLOSED ← CLOSED ∪ {v}
10         if IsGoal(v) then return Solution(v, s)
11         Expand(v)
12 procedure Expand(v)
13     foreach u ∈ Succ(v) do
14         if u ∉ OPEN ∪ CLOSED then
15             g[u] ← g[v] + c(v, u)
16             f[u] ← g[u] + h(u)
17             Parent[u] ← v
18             Insert_f(OPEN, u)
19         else if u ∈ OPEN then
20             if g[v] + c(v, u) < g[v] then
21                 g[u] ← g[v] + c(v, u)
22                 f[u] ← g[u] + h(u)
23                 Parent[u] ← v
24         else                                  # if u ∈ CLOSED
25             if g[v] + c(v, u) < g[u] then
26                 g[u] ← g[v] + c(v, u)
27                 f[u] ← g[u] + h(u)
28                 Parent[u] ← v
29                 CLOSED ← CLOSED\{u}
30                 Insert_f(OPEN, u)
```



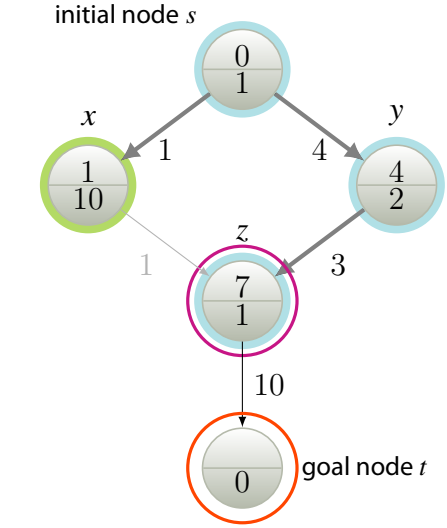

```
1  function A*(s)
2      OPEN ← new PriorityQueue_f
3      g[s] ← 0; f[s] ← h(s)
4      Insert_f(OPEN, s)
5      CLOSED ← ∅
6      loop do
7          if IsEmpty (OPEN) then return "failure"
8          v ← DeleteMin_f(OPEN)
9          CLOSED ← CLOSED ∪ {v}
10         if IsGoal(v) then return Solution(v, s)
11         Expand(v)
12 procedure Expand(v)
13     foreach u ∈ Succ(v) do
14         if u ∉ OPEN ∪ CLOSED then
15             g[u] ← g[v] + c(v, u)
16             f[u] ← g[u] + h(u)
17             Parent[u] ← v
18             Insert_f(OPEN, u)
19         else if u ∈ OPEN then
20             if g[v] + c(v, u) < g[v] then
21                 g[u] ← g[v] + c(v, u)
22                 f[u] ← g[u] + h(u)
23                 Parent[u] ← v
24         else                                  # if u ∈ CLOSED
25             if g[v] + c(v, u) < g[u] then
26                 g[u] ← g[v] + c(v, u)
27                 f[u] ← g[u] + h(u)
28                 Parent[u] ← v
29                 CLOSED ← CLOSED\{u}
30                 Insert_f(OPEN, u)
```

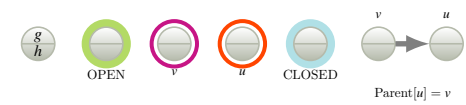



```
1  function A*(s)
2      OPEN ← new PriorityQueue_f
3      g[s] ← 0; f[s] ← h(s)
4      Insert_f(OPEN, s)
5      CLOSED ← ∅
6      loop do
7          if IsEmpty(OPEN) then return "failure"
8          v ← DeleteMin_f(OPEN)
9          CLOSED ← CLOSED ∪ {v}
10         if IsGoal(v) then return Solution(v, s)
11         Expand(v)
12 procedure Expand(v)
13     foreach u ∈ Succ(v) do
14         if u ∉ OPEN ∪ CLOSED then
15             g[u] ← g[v] + c(v, u)
16             f[u] ← g[u] + h(u)
17             Parent[u] ← v
18             Insert_f(OPEN, u)
19         else if u ∈ OPEN then
20             if g[v] + c(v, u) < g[v] then
21                 g[u] ← g[v] + c(v, u)
22                 f[u] ← g[u] + h(u)
23                 Parent[u] ← v
24         else                                  # if u ∈ CLOSED
25             if g[v] + c(v, u) < g[u] then
26                 g[u] ← g[v] + c(v, u)
27                 f[u] ← g[u] + h(u)
28                 Parent[u] ← v
29                 CLOSED ← CLOSED\{u}
30                 Insert_f(OPEN, u)
```

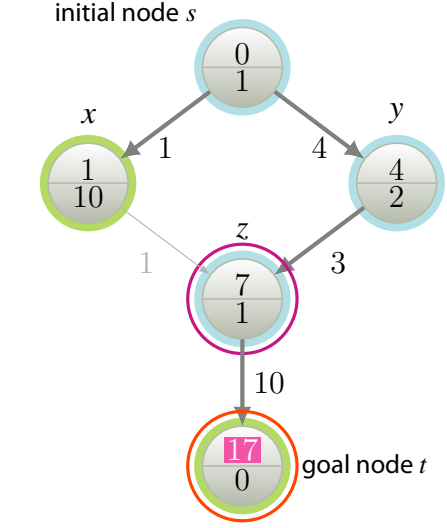



```
1  function A*(s)
2      OPEN ← new PriorityQueue_f
3      g[s] ← 0; f[s] ← h(s)
4      Insert_f(OPEN, s)
5      CLOSED ← ∅
6      loop do
7          if IsEmpty(OPEN) then return "failure"
8          v ← DeleteMin_f(OPEN)
9          CLOSED ← CLOSED ∪ {v}
10         if IsGoal(v) then return Solution(v, s)
11         Expand(v)
12 procedure Expand(v)
13     foreach u ∈ Succ(v) do
14         if u ∉ OPEN ∪ CLOSED then
15             g[u] ← g[v] + c(v, u)
16             f[u] ← g[u] + h(u)
17             Parent[u] ← v
18             Insert_f(OPEN, u)
19         else if u ∈ OPEN then
20             if g[v] + c(v, u) < g[v] then
21                 g[u] ← g[v] + c(v, u)
22                 f[u] ← g[u] + h(u)
23                 Parent[u] ← v
24         else                                  # if u ∈ CLOSED
25             if g[v] + c(v, u) < g[u] then
26                 g[u] ← g[v] + c(v, u)
27                 f[u] ← g[u] + h(u)
28                 Parent[u] ← v
29                 CLOSED ← CLOSED\{u}
30                 Insert_f(OPEN, u)
```

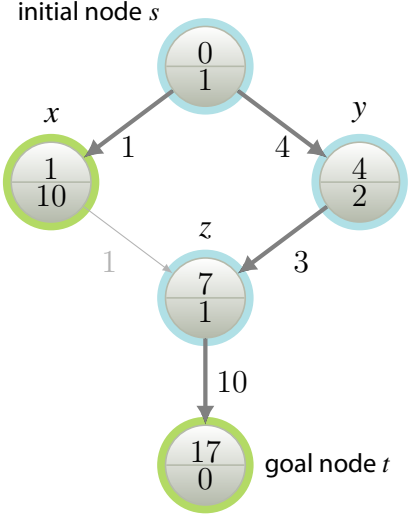




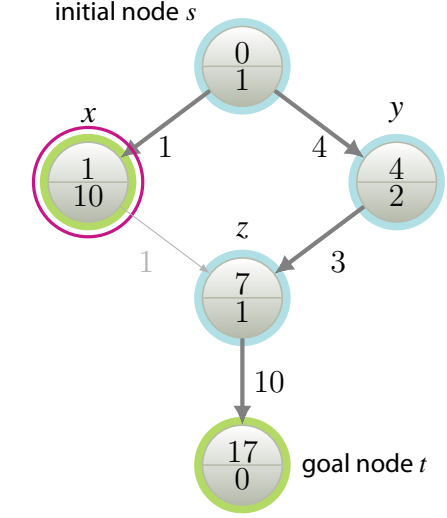

```
1  function A*(s)
2      OPEN ← new PriorityQueue_f
3      g[s] ← 0; f[s] ← h(s)
4      Insert_f(OPEN, s)
5      CLOSED ← ∅
6      loop do
7          if IsEmpty (OPEN) then return "failure"
8          v ← DeleteMin_f(OPEN)
9          CLOSED ← CLOSED ∪ {v}
10         if IsGoal(v) then return Solution(v, s)
11         Expand(v)
12 procedure Expand(v)
13     foreach u ∈ Succ(v) do
14         if u ∉ OPEN ∪ CLOSED then
15             g[u] ← g[v] + c(v, u)
16             f[u] ← g[u] + h(u)
17             Parent[u] ← v
18             Insert_f(OPEN, u)
19         else if u ∈ OPEN then
20             if g[v] + c(v, u) < g[v] then
21                 g[u] ← g[v] + c(v, u)
22                 f[u] ← g[u] + h(u)
23                 Parent[u] ← v
24         else                                  # if u ∈ CLOSED
25             if g[v] + c(v, u) < g[u] then
26                 g[u] ← g[v] + c(v, u)
27                 f[u] ← g[u] + h(u)
28                 Parent[u] ← v
29                 CLOSED ← CLOSED\{u}
30                 Insert_f(OPEN, u)
```



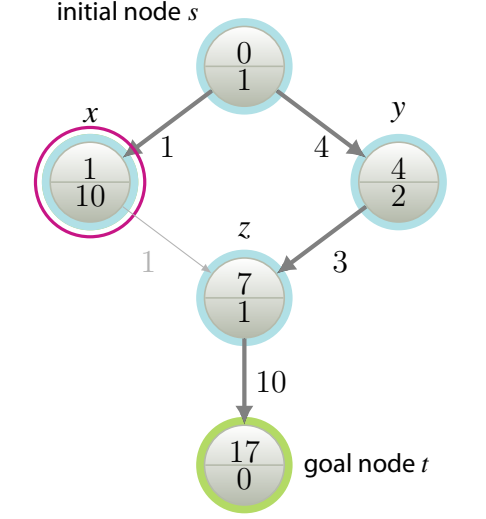

```
1  function A*(s)
2      OPEN ← new PriorityQueue_f
3      g[s] ← 0; f[s] ← h(s)
4      Insert_f(OPEN, s)
5      CLOSED ← ∅
6      loop do
7          if IsEmpty (OPEN) then return "failure"
8          v ← DeleteMin_f(OPEN)
9          CLOSED ← CLOSED ∪ {v}
10         if IsGoal(v) then return Solution(v, s)
11         Expand(v)
12 procedure Expand(v)
13     foreach u ∈ Succ(v) do
14         if u ∉ OPEN ∪ CLOSED then
15             g[u] ← g[v] + c(v, u)
16             f[u] ← g[u] + h(u)
17             Parent[u] ← v
18             Insert_f(OPEN, u)
19         else if u ∈ OPEN then
20             if g[v] + c(v, u) < g[v] then
21                 g[u] ← g[v] + c(v, u)
22                 f[u] ← g[u] + h(u)
23                 Parent[u] ← v
24         else                                  # if u ∈ CLOSED
25             if g[v] + c(v, u) < g[u] then
26                 g[u] ← g[v] + c(v, u)
27                 f[u] ← g[u] + h(u)
28                 Parent[u] ← v
29                 CLOSED ← CLOSED\{u}
30                 Insert_f(OPEN, u)
```



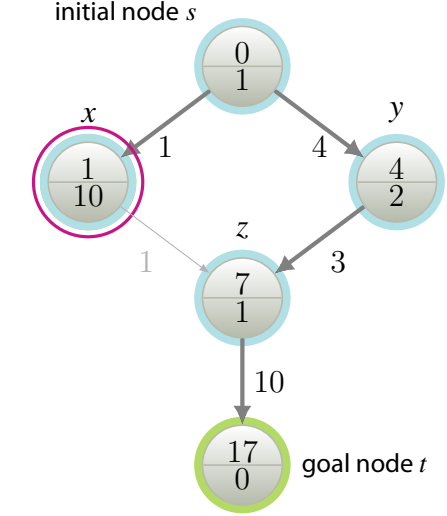

```
1  function A*(s)
2      OPEN ← new PriorityQueue_f
3      g[s] ← 0; f[s] ← h(s)
4      Insert_f(OPEN, s)
5      CLOSED ← ∅
6      loop do
7          if IsEmpty (OPEN) then return "failure"
8          v ← DeleteMin_f(OPEN)
9          CLOSED ← CLOSED ∪ {v}
10         if IsGoal(v) then return Solution(v, s)
11         Expand(v)
12 procedure Expand(v)
13     foreach u ∈ Succ(v) do
14         if u ∉ OPEN ∪ CLOSED then
15             g[u] ← g[v] + c(v, u)
16             f[u] ← g[u] + h(u)
17             Parent[u] ← v
18             Insert_f(OPEN, u)
19         else if u ∈ OPEN then
20             if g[v] + c(v, u) < g[v] then
21                 g[u] ← g[v] + c(v, u)
22                 f[u] ← g[u] + h(u)
23                 Parent[u] ← v
24         else                                  # if u ∈ CLOSED
25             if g[v] + c(v, u) < g[u] then
26                 g[u] ← g[v] + c(v, u)
27                 f[u] ← g[u] + h(u)
28                 Parent[u] ← v
29                 CLOSED ← CLOSED\{u}
30                 Insert_f(OPEN, u)
```



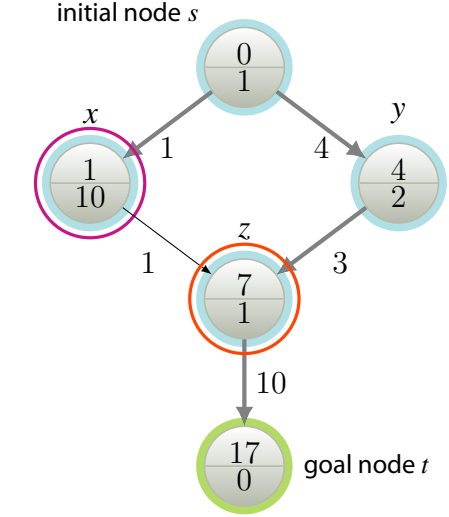

```
1  function A*(s)
2      OPEN ← new PriorityQueue_f
3      g[s] ← 0; f[s] ← h(s)
4      Insert_f(OPEN, s)
5      CLOSED ← ∅
6      loop do
7          if IsEmpty (OPEN) then return "failure"
8          v ← DeleteMin_f(OPEN)
9          CLOSED ← CLOSED ∪ {v}
10         if IsGoal(v) then return Solution(v, s)
11         Expand(v)
12 procedure Expand(v)
13     foreach u ∈ Succ(v) do
14         if u ∉ OPEN ∪ CLOSED then
15             g[u] ← g[v] + c(v, u)
16             f[u] ← g[u] + h(u)
17             Parent[u] ← v
18             Insert_f(OPEN, u)
19         else if u ∈ OPEN then
20             if g[v] + c(v, u) < g[v] then
21                 g[u] ← g[v] + c(v, u)
22                 f[u] ← g[u] + h(u)
23                 Parent[u] ← v
24         else                                  # if u ∈ CLOSED
25             if g[v] + c(v, u) < g[u] then
26                 g[u] ← g[v] + c(v, u)
27                 f[u] ← g[u] + h(u)
28                 Parent[u] ← v
29                 CLOSED ← CLOSED\{u}
30                 Insert_f(OPEN, u)
```

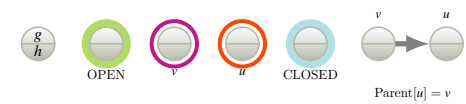



```
1  function A*(s)
2      OPEN ← new PriorityQueue_f
3      g[s] ← 0; f[s] ← h(s)
4      Insert_f(OPEN, s)
5      CLOSED ← ∅
6      loop do
7          if IsEmpty (OPEN) then return "failure"
8          v ← DeleteMin_f(OPEN)
9          CLOSED ← CLOSED ∪ {v}
10         if IsGoal(v) then return Solution(v, s)
11         Expand(v)
12 procedure Expand(v)
13     foreach u ∈ Succ(v) do
14         if u ∉ OPEN ∪ CLOSED then
15             g[u] ← g[v] + c(v, u)
16             f[u] ← g[u] + h(u)
17             Parent[u] ← v
18             Insert_f(OPEN, u)
19         else if u ∈ OPEN then
20             if g[v] + c(v, u) < g[v] then
21                 g[u] ← g[v] + c(v, u)
22                 f[u] ← g[u] + h(u)
23                 Parent[u] ← v
24         else                                  # if u ∈ CLOSED
25             if g[v] + c(v, u) < g[u] then
26                 g[u] ← g[v] + c(v, u)
27                 f[u] ← g[u] + h(u)
28                 Parent[u] ← v
29                 CLOSED ← CLOSED\{u}
30                 Insert_f(OPEN, u)
```

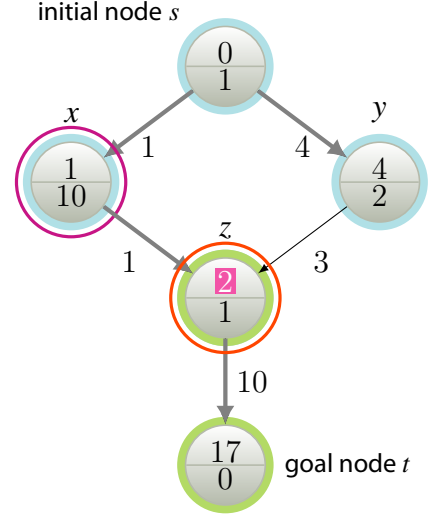



```
1  function A*(s)
2      OPEN ← new PriorityQueue_f
3      g[s] ← 0; f[s] ← h(s)
4      Insert_f(OPEN, s)
5      CLOSED ← ∅
6      loop do
7          if IsEmpty(OPEN) then return "failure"
8          v ← DeleteMin_f(OPEN)
9          CLOSED ← CLOSED ∪ {v}
10         if IsGoal(v) then return Solution(v, s)
11         Expand(v)
12 procedure Expand(v)
13     foreach u ∈ Succ(v) do
14         if u ∉ OPEN ∪ CLOSED then
15             g[u] ← g[v] + c(v, u)
16             f[u] ← g[u] + h(u)
17             Parent[u] ← v
18             Insert_f(OPEN, u)
19         else if u ∈ OPEN then
20             if g[v] + c(v, u) < g[v] then
21                 g[u] ← g[v] + c(v, u)
22                 f[u] ← g[u] + h(u)
23                 Parent[u] ← v
24         else                                  # if u ∈ CLOSED
25             if g[v] + c(v, u) < g[u] then
26                 g[u] ← g[v] + c(v, u)
27                 f[u] ← g[u] + h(u)
28                 Parent[u] ← v
29                 CLOSED ← CLOSED\{u}
30                 Insert_f(OPEN, u)
```

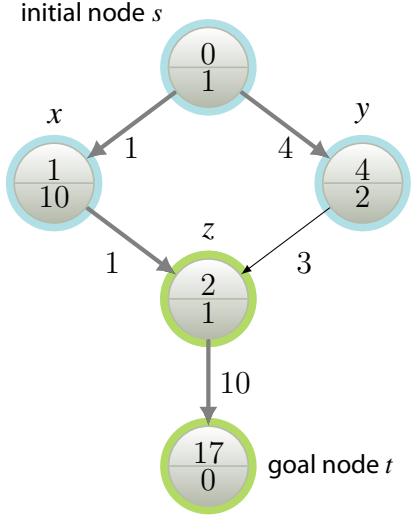




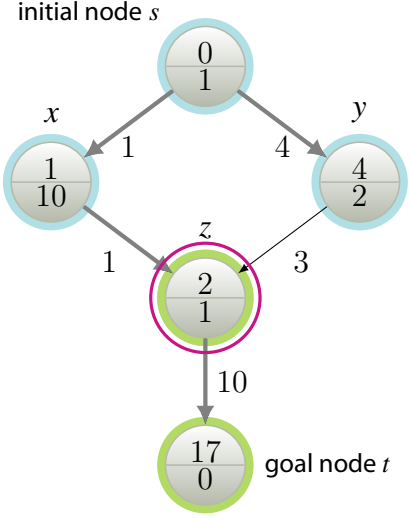

```
1  function A*(s)
2      OPEN ← new PriorityQueue_f
3      g[s] ← 0; f[s] ← h(s)
4      Insert_f(OPEN, s)
5      CLOSED ← ∅
6      loop do
7          if IsEmpty (OPEN) then return "failure"
8          v ← DeleteMin_f(OPEN)
9          CLOSED ← CLOSED ∪ {v}
10         if IsGoal(v) then return Solution(v, s)
11         Expand(v)
12 procedure Expand(v)
13     foreach u ∈ Succ(v) do
14         if u ∉ OPEN ∪ CLOSED then
15             g[u] ← g[v] + c(v, u)
16             f[u] ← g[u] + h(u)
17             Parent[u] ← v
18             Insert_f(OPEN, u)
19         else if u ∈ OPEN then
20             if g[v] + c(v, u) < g[v] then
21                 g[u] ← g[v] + c(v, u)
22                 f[u] ← g[u] + h(u)
23                 Parent[u] ← v
24         else                                  # if u ∈ CLOSED
25             if g[v] + c(v, u) < g[u] then
26                 g[u] ← g[v] + c(v, u)
27                 f[u] ← g[u] + h(u)
28                 Parent[u] ← v
29                 CLOSED ← CLOSED\{u}
30                 Insert_f(OPEN, u)
```



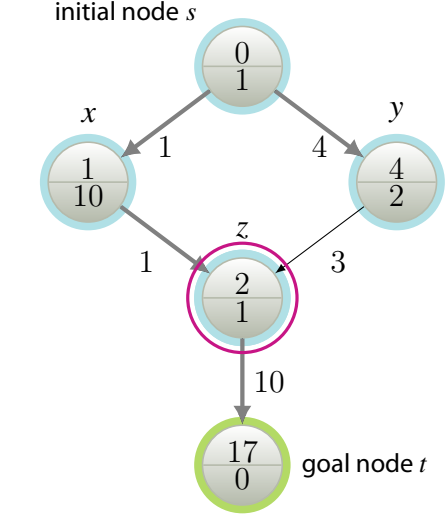

```
1  function A*(s)
2      OPEN ← new PriorityQueue_f
3      g[s] ← 0; f[s] ← h(s)
4      Insert_f(OPEN, s)
5      CLOSED ← ∅
6      loop do
7          if IsEmpty (OPEN) then return "failure"
8          v ← DeleteMin_f(OPEN)
9          CLOSED ← CLOSED ∪ {v}
10         if IsGoal(v) then return Solution(v, s)
11         Expand(v)
12 procedure Expand(v)
13     foreach u ∈ Succ(v) do
14         if u ∉ OPEN ∪ CLOSED then
15             g[u] ← g[v] + c(v, u)
16             f[u] ← g[u] + h(u)
17             Parent[u] ← v
18             Insert_f(OPEN, u)
19         else if u ∈ OPEN then
20             if g[v] + c(v, u) < g[v] then
21                 g[u] ← g[v] + c(v, u)
22                 f[u] ← g[u] + h(u)
23                 Parent[u] ← v
24         else                                  # if u ∈ CLOSED
25             if g[v] + c(v, u) < g[u] then
26                 g[u] ← g[v] + c(v, u)
27                 f[u] ← g[u] + h(u)
28                 Parent[u] ← v
29                 CLOSED ← CLOSED\{u}
30                 Insert_f(OPEN, u)
```

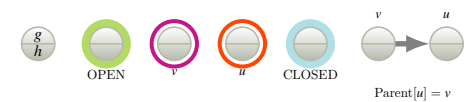



```
1  function A*(s)
2      OPEN ← new PriorityQueue_f
3      g[s] ← 0; f[s] ← h(s)
4      Insert_f(OPEN, s)
5      CLOSED ← ∅
6      loop do
7          if IsEmpty (OPEN) then return "failure"
8          v ← DeleteMin_f(OPEN)
9          CLOSED ← CLOSED ∪ {v}
10         if IsGoal(v) then return Solution(v, s)
11         Expand(v)
12 procedure Expand(v)
13     foreach u ∈ Succ(v) do
14         if u ∉ OPEN ∪ CLOSED then
15             g[u] ← g[v] + c(v, u)
16             f[u] ← g[u] + h(u)
17             Parent[u] ← v
18             Insert_f(OPEN, u)
19         else if u ∈ OPEN then
20             if g[v] + c(v, u) < g[v] then
21                 g[u] ← g[v] + c(v, u)
22                 f[u] ← g[u] + h(u)
23                 Parent[u] ← v
24         else                                  # if u ∈ CLOSED
25             if g[v] + c(v, u) < g[u] then
26                 g[u] ← g[v] + c(v, u)
27                 f[u] ← g[u] + h(u)
28                 Parent[u] ← v
29                 CLOSED ← CLOSED\{u}
30                 Insert_f(OPEN, u)
```

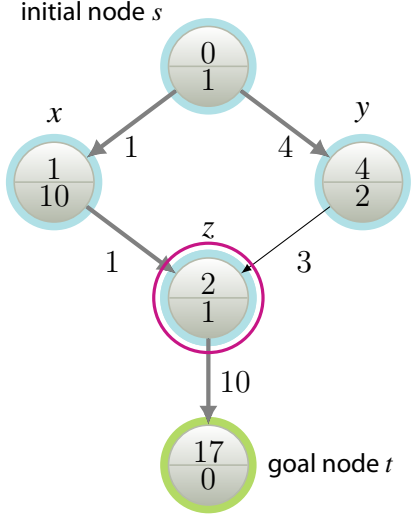



```
1  function A*(s)
2      OPEN ← new PriorityQueue_f
3      g[s] ← 0; f[s] ← h(s)
4      Insert_f(OPEN, s)
5      CLOSED ← ∅
6      loop do
7          if IsEmpty (OPEN) then return "failure"
8          v ← DeleteMin_f(OPEN)
9          CLOSED ← CLOSED ∪ {v}
10         if IsGoal(v) then return Solution(v, s)
11         Expand(v)
12 procedure Expand(v)
13     foreach u ∈ Succ(v) do
14         if u ∉ OPEN ∪ CLOSED then
15             g[u] ← g[v] + c(v, u)
16             f[u] ← g[u] + h(u)
17             Parent[u] ← v
18             Insert_f(OPEN, u)
19         else if u ∈ OPEN then
20             if g[v] + c(v, u) < g[v] then
21                 g[u] ← g[v] + c(v, u)
22                 f[u] ← g[u] + h(u)
23                 Parent[u] ← v
24         else                                  # if u ∈ CLOSED
25             if g[v] + c(v, u) < g[u] then
26                 g[u] ← g[v] + c(v, u)
27                 f[u] ← g[u] + h(u)
28                 Parent[u] ← v
29                 CLOSED ← CLOSED\{u}
30                 Insert_f(OPEN, u)
```

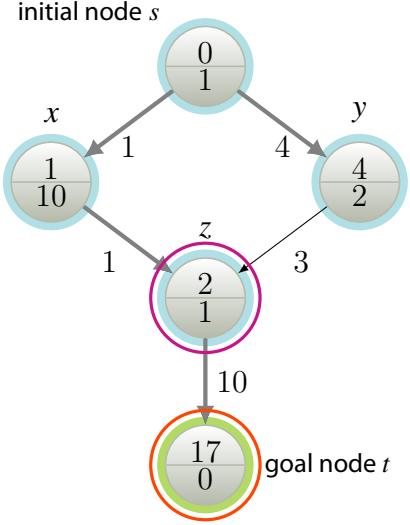




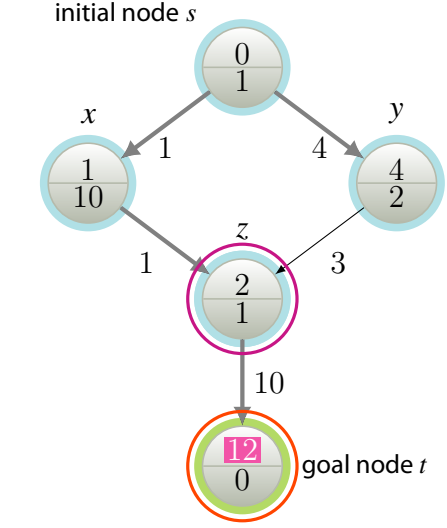

```
1  function A*(s)
2      OPEN ← new PriorityQueue_f
3      g[s] ← 0; f[s] ← h(s)
4      Insert_f(OPEN, s)
5      CLOSED ← ∅
6      loop do
7          if IsEmpty (OPEN) then return "failure"
8          v ← DeleteMin_f(OPEN)
9          CLOSED ← CLOSED ∪ {v}
10         if IsGoal(v) then return Solution(v, s)
11         Expand(v)
12 procedure Expand(v)
13     foreach u ∈ Succ(v) do
14         if u ∉ OPEN ∪ CLOSED then
15             g[u] ← g[v] + c(v, u)
16             f[u] ← g[u] + h(u)
17             Parent[u] ← v
18             Insert_f(OPEN, u)
19         else if u ∈ OPEN then
20             if g[v] + c(v, u) < g[v] then
21                 g[u] ← g[v] + c(v, u)
22                 f[u] ← g[u] + h(u)
23                 Parent[u] ← v
24         else                                  # if u ∈ CLOSED
25             if g[v] + c(v, u) < g[u] then
26                 g[u] ← g[v] + c(v, u)
27                 f[u] ← g[u] + h(u)
28                 Parent[u] ← v
29                 CLOSED ← CLOSED\{u}
30                 Insert_f(OPEN, u)
```



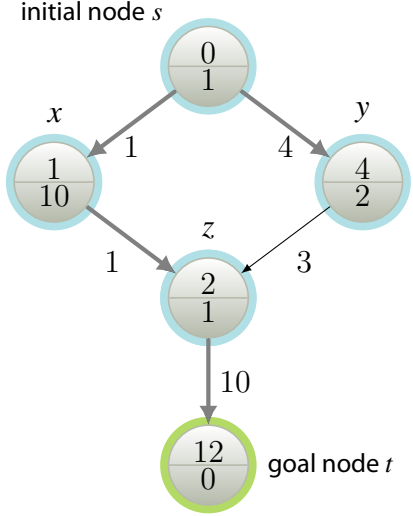

```
1  function A*(s)
2      OPEN ← new PriorityQueue_f
3      g[s] ← 0; f[s] ← h(s)
4      Insert_f(OPEN, s)
5      CLOSED ← ∅
6      loop do
7          if IsEmpty (OPEN) then return "failure"
8          v ← DeleteMin_f(OPEN)
9          CLOSED ← CLOSED ∪ {v}
10         if IsGoal(v) then return Solution(v, s)
11         Expand(v)
12 procedure Expand(v)
13     foreach u ∈ Succ(v) do
14         if u ∉ OPEN ∪ CLOSED then
15             g[u] ← g[v] + c(v, u)
16             f[u] ← g[u] + h(u)
17             Parent[u] ← v
18             Insert_f(OPEN, u)
19         else if u ∈ OPEN then
20             if g[v] + c(v, u) < g[v] then
21                 g[u] ← g[v] + c(v, u)
22                 f[u] ← g[u] + h(u)
23                 Parent[u] ← v
24         else                                  # if u ∈ CLOSED
25             if g[v] + c(v, u) < g[u] then
26                 g[u] ← g[v] + c(v, u)
27                 f[u] ← g[u] + h(u)
28                 Parent[u] ← v
29                 CLOSED ← CLOSED\{u}
30                 Insert_f(OPEN, u)
```

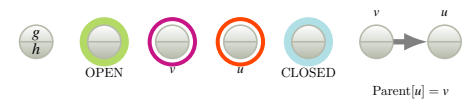



```
1  function A*(s)
2      OPEN ← new PriorityQueue_f
3      g[s] ← 0; f[s] ← h(s)
4      Insert_f(OPEN, s)
5      CLOSED ← ∅
6      loop do
7          if IsEmpty (OPEN) then return "failure"
8          v ← DeleteMin_f(OPEN)
9          CLOSED ← CLOSED ∪ {v}
10         if IsGoal(v) then return Solution(v, s)
11         Expand(v)
12 procedure Expand(v)
13     foreach u ∈ Succ(v) do
14         if u ∉ OPEN ∪ CLOSED then
15             g[u] ← g[v] + c(v, u)
16             f[u] ← g[u] + h(u)
17             Parent[u] ← v
18             Insert_f(OPEN, u)
19         else if u ∈ OPEN then
20             if g[v] + c(v, u) < g[v] then
21                 g[u] ← g[v] + c(v, u)
22                 f[u] ← g[u] + h(u)
23                 Parent[u] ← v
24         else                                  # if u ∈ CLOSED
25             if g[v] + c(v, u) < g[u] then
26                 g[u] ← g[v] + c(v, u)
27                 f[u] ← g[u] + h(u)
28                 Parent[u] ← v
29                 CLOSED ← CLOSED\{u}
30                 Insert_f(OPEN, u)
```

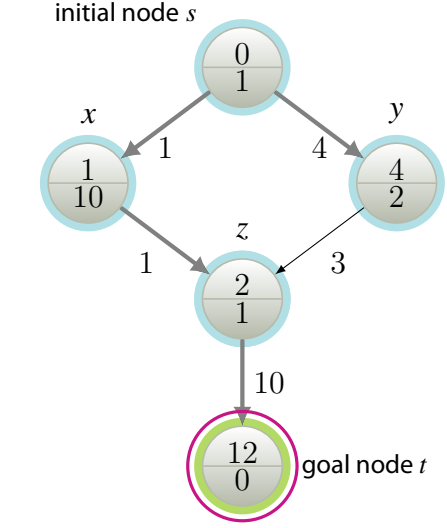



```
1  function A*(s)
2      OPEN ← new PriorityQueue_f
3      g[s] ← 0; f[s] ← h(s)
4      Insert_f(OPEN, s)
5      CLOSED ← ∅
6      loop do
7          if IsEmpty(OPEN) then return "failure"
8          v ← DeleteMin_f(OPEN)
9          CLOSED ← CLOSED ∪ {v}
10         if IsGoal(v) then return Solution(v, s)
11         Expand(v)
12 procedure Expand(v)
13     foreach u ∈ Succ(v) do
14         if u ∉ OPEN ∪ CLOSED then
15             g[u] ← g[v] + c(v, u)
16             f[u] ← g[u] + h(u)
17             Parent[u] ← v
18             Insert_f(OPEN, u)
19         else if u ∈ OPEN then
20             if g[v] + c(v, u) < g[v] then
21                 g[u] ← g[v] + c(v, u)
22                 f[u] ← g[u] + h(u)
23                 Parent[u] ← v
24         else                                  # if u ∈ CLOSED
25             if g[v] + c(v, u) < g[u] then
26                 g[u] ← g[v] + c(v, u)
27                 f[u] ← g[u] + h(u)
28                 Parent[u] ← v
29                 CLOSED ← CLOSED\{u}
30                 Insert_f(OPEN, u)
```

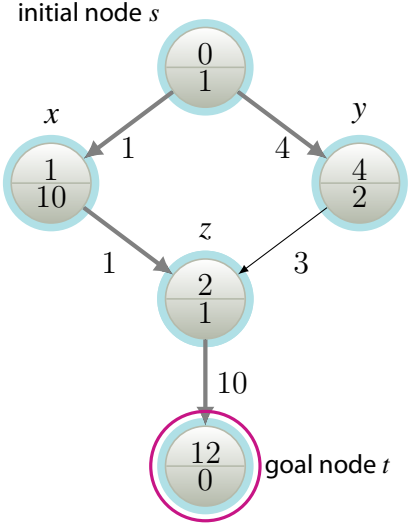




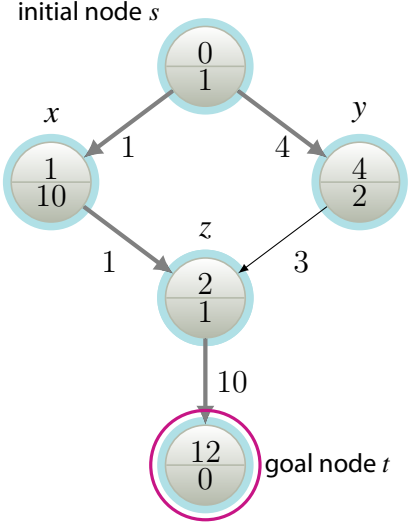

```
1  function A*(s)
2      OPEN ← new PriorityQueue_f
3      g[s] ← 0; f[s] ← h(s)
4      Insert_f(OPEN, s)
5      CLOSED ← ∅
6      loop do
7          if IsEmpty (OPEN) then return "failure"
8          v ← DeleteMin_f(OPEN)
9          CLOSED ← CLOSED ∪ {v}
10         if IsGoal(v) then return Solution(v, s)
11         Expand(v)
12 procedure Expand(v)
13     foreach u ∈ Succ(v) do
14         if u ∉ OPEN ∪ CLOSED then
15             g[u] ← g[v] + c(v, u)
16             f[u] ← g[u] + h(u)
17             Parent[u] ← v
18             Insert_f(OPEN, u)
19         else if u ∈ OPEN then
20             if g[v] + c(v, u) < g[v] then
21                 g[u] ← g[v] + c(v, u)
22                 f[u] ← g[u] + h(u)
23                 Parent[u] ← v
24         else                                  # if u ∈ CLOSED
25             if g[v] + c(v, u) < g[u] then
26                 g[u] ← g[v] + c(v, u)
27                 f[u] ← g[u] + h(u)
28                 Parent[u] ← v
29                 CLOSED ← CLOSED\{u}
30                 Insert_f(OPEN, u)
```




```
1  function A*(s)
2      OPEN ← new PriorityQueue_f
3      g[s] ← 0; f[s] ← h(s)
4      Insert_f(OPEN, s)
5      CLOSED ← ∅
6      loop do
7          if IsEmpty (OPEN) then return "failure"
8          v ← DeleteMin_f(OPEN)
9          CLOSED ← CLOSED ∪ {v}
10         if IsGoal(v) then return Solution(v, s)
11         Expand(v)
12 procedure Expand(v)
13     foreach u ∈ Succ(v) do
14         if u ∉ OPEN ∪ CLOSED then
15             g[u] ← g[v] + c(v, u)
16             f[u] ← g[u] + h(u)
17             Parent[u] ← v
18             Insert_f(OPEN, u)
19         else if u ∈ OPEN then
20             if g[v] + c(v, u) < g[v] then
21                 g[u] ← g[v] + c(v, u)
22                 f[u] ← g[u] + h(u)
23                 Parent[u] ← v
24         else                                  # if u ∈ CLOSED
25             if g[v] + c(v, u) < g[u] then
26                 g[u] ← g[v] + c(v, u)
27                 f[u] ← g[u] + h(u)
28                 Parent[u] ← v
29                 CLOSED ← CLOSED\{u}
30                 Insert_f(OPEN, u)
```




```
1  function A*(s)
2      OPEN ← new PriorityQueue_f
3      g[s] ← 0; f[s] ← h(s)
4      Insert_f(OPEN, s)
5      CLOSED ← ∅
6      loop do
7          if IsEmpty (OPEN) then return "failure"
8          v ← DeleteMin_f(OPEN)
9          CLOSED ← CLOSED ∪ {v}
10         if IsGoal(v) then return Solution(v, s)
11         Expand(v)
12 procedure Expand(v)
13     foreach u ∈ Succ(v) do
14         if u ∉ OPEN ∪ CLOSED then
15             g[u] ← g[v] + c(v, u)
16             f[u] ← g[u] + h(u)
17             Parent[u] ← v
18             Insert_f(OPEN, u)
19         else if u ∈ OPEN then
20             if g[v] + c(v, u) < g[v] then
21                 g[u] ← g[v] + c(v, u)
22                 f[u] ← g[u] + h(u)
23                 Parent[u] ← v
24         else                                  # if u ∈ CLOSED
25             if g[v] + c(v, u) < g[u] then
26                 g[u] ← g[v] + c(v, u)
27                 f[u] ← g[u] + h(u)
28                 Parent[u] ← v
29                 CLOSED ← CLOSED\{u}
30                 Insert_f(OPEN, u)
```




```
1  function A*(s)
2      OPEN ← new PriorityQueue_f
3      g[s] ← 0; f[s] ← h(s)
4      Insert_f(OPEN, s)
5      CLOSED ← ∅
6      loop do
7          if IsEmpty (OPEN) then return "failure"
8          v ← DeleteMin_f(OPEN)
9          CLOSED ← CLOSED ∪ {v}
10         if IsGoal(v) then return Solution(v, s)
11         Expand(v)
12 procedure Expand(v)
13     foreach u ∈ Succ(v) do
14         if u ∉ OPEN ∪ CLOSED then
15             g[u] ← g[v] + c(v, u)
16             f[u] ← g[u] + h(u)
17             Parent[u] ← v
18             Insert_f(OPEN, u)
19         else if u ∈ OPEN then
20             if g[v] + c(v, u) < g[v] then
21                 g[u] ← g[v] + c(v, u)
22                 f[u] ← g[u] + h(u)
23                 Parent[u] ← v
24         else                                  # if u ∈ CLOSED
25             if g[v] + c(v, u) < g[u] then
26                 g[u] ← g[v] + c(v, u)
27                 f[u] ← g[u] + h(u)
28                 Parent[u] ← v
29                 CLOSED ← CLOSED\{u}
30                 Insert_f(OPEN, u)
```