**Algorithm 1** A*

```
1: procedure MAIN()
2:     open := closed := φ;
3:     g(s_start) := 0;
4:     parent(s_start) := s_start;
5:     open.Insert(s_start, g(s_start) + h(s_start));
6:     while open ≠ φ do
7:         s := open.Pop();
8:         if s = s_goal then return "success";
9:         closed := closed ∪ {s};
10:        for each s' ∈ nghbr(s) do
11:            if s' ∉ closed then
12:                if s' ∉ open then
13:                    g(s') := ∞;
14:                    parent(s') := NULL;
15:                UpdateVertex(s, s');
16:    return "fail";
17:
18: procedure UPDATEVERTEX(S, S')
19:    g_old := g(s');
20:    ComputeCost(s, s');
21:    if g(s') < g_old then
22:        if s' ∈ open then
23:            open.Remove(s');
24:        open.Insert(s', g(s') + h(s');
25:
26: procedure COMPUTECOST(S, S')
27:    if g(s) + c(s, s') < g(s') then
28:        parent(s') := s;
29:        g(s') := g(s) + c(s, s');
```

**Algorithm 2** Theta*

```
1: procedure Main()
2:     open := closed := φ;
3:     g(s_start) := 0;
4:     parent(s_start) := s_start;
5:     open.Insert(s_start, g(s_start) + h(s_start));
6:     while open ≠ φ do
7:         s := open.Pop();
8:         if s = s_goal then return "success";
9:         closed := closed ∪ {s};
10:        for each s' ∈ nghbr(s) do
11:            if s' ∉ closed then
12:                if s' ∉ open then
13:                    g(s') := ∞;
14:                    parent(s') := NULL;
15:                UpdateVertex(s, s');
16:    return "fail";
17:
18: procedure UpdateVertex(s, s')
19:    g_old := g(s');
20:    ComputeCost(s, s');
21:    if g(s') < g_old then
22:        if s' ∈ open then
23:            open.Remove(s');
24:        open.Insert(s', g(s') + h(s'));
25:
26: procedure ComputeCost(s, s')
27:    if lineOfSight(parent(s), s') then
28:        /* Path2 */
29:        if g(parent(s)) + c(parent(s), s') < g(s') then
30:            parent(s') := parent(s);
31:            g(s') := g(parent(s)) + c(parent(s), s');
32:    else
33:        /* Path1 */
34:        if g(s) + c(s, s') < g(s') then
35:            parent(s') := s;
36:            g(s') := g(s) + c(s, s');
```

**Algorithm 3** Lazy Theta*

```
1:  procedure Main()
2:      open := closed := φ;
3:      g(s_start) := 0;
4:      parent(s_start) := s_start;
5:      open.Insert(s_start, g(s_start) + h(s_start));
6:      while open ≠ φ do
7:          s := open.Pop();
8:          SetVertex(s);
9:          if s = s_goal then return "success";
10:         closed := closed ∪ {s};
11:         for each s' ∈ nghbr(s) do
12:             if s' ∉ closed then
13:                 if s' ∉ open then
14:                     g(s') := ∞;
15:                     parent(s') := NULL;
16:                 UpdateVertex(s, s');
17:     return "fail";
18:
19: procedure UpdateVertex(s, s')
20:     g_old := g(s');
21:     ComputeCost(s, s');
22:     if g(s') < g_old then
23:         if s' ∈ open then
24:             open.Remove(s');
25:         open.Insert(s', g(s') + h(s');
26:
27: procedure ComputeCost(s, s')
28:     /* Path2 */
29:     if g(parent(s)) + c(parent(s), s') < g(s') then
30:         parent(s') := parent(s);
31:         g(s') := g(parent(s)) + c(parent(s), s');
32:
33: procedure SetVertex(s)
34:     if !lineOfSight(parent(s), s) then
35:         /* Path1 */
36:         parent(s) := argmin_{s'∈nghbr(s)∩closed}(g(s') + c(s', s));
37:         g(s) := min_{s'∈nghbr(s)∩closed}(g(s') + c(s', s));
```