Algorithm 1 Insertion tache

```
1: function ADD_TASK(du_t, cpu_t)                                  ▷ duration, number of cpu
2:     start_key ← [0, du_t, cpu_t]
3:     end_key ← [+∞, +∞, +∞]
4:     profile_tree → node_loop(start_key, end_key, {
5:         starting_time_min ← (freespace → {starting_time})
6:         processor_range_t ← (freespace → {cpu})
7:         return 1
8:     })
9:                                          ▷ Remove of the range, the number of cpu
10:    processor_range_t ← (processor_range_t → reduce_to_basic(cpu_t))
11:    start_key ← [0, 1, 1]
12:    end_key ← [starting_time_min + du_t, +∞, +∞]
13:    profile_tree → node_loop(start_key, end_key, {          ▷ Research with contraints
14:        push freespace_impacted, freespace
15:        return 0
16:    })
17:    for freespace ← freespace_impacted do
18:        cut_freespace(freespace, starting_time_min, du_t, processor_range_t)
19:    end for
20: end function
```

Algorithm 2 Decoupage Freespace

```
1: function CUT_FREESPACE(freespace, start_time, duration, processor_range)
2:     if Intersection(freespace → {cpu}, processor_range) > 0 then
3:         profile_tree → remove(freespace)
4:         if freespace → {starting_time} < start_time then
5:             new freespaceright_freespace ← (freespace → {starting_time}, start_time − freespace → {starting_time}, freespace → {cpu})
6:             profile_tree → add(right_freespace)
7:         end if
8:         if freespace → {starting_time} < (start_time + duration) then
9:             new freespaceleft_freespace ← (freespace → {starting_time}, start_time − freespace → {starting_time}, freespace → {cpu})
10:            profile_tree → add(left_freespace)
11:        end if
12:        range_test ← (freespace → {cpu} → copy())
13:        range_test → remove(processor_range)
14:        if range_test → size() > 0 then
15:            new freespacenew_freespace ← (freespace → {starting_time}, freespace → {duration}, range_test)
16:            profile_tree → add(new_freespace)
17:        end if
18:    end if
19: end function
```

Algorithm 3 Suppression tache

```
1: function REMOVE_TASK
2: end function
```