RRT_CONNECT

```
Ta.initialize(q_start)
Tb.initialize(q_goal)
success = FALSE
for i=1 to MAXNODES
  q_rand=RANDOMCONFIG()
  [result,Ta,q_target]=RRT_EXTEND_SINGLE(Ta,q_rand,step_length)
  if (result==TRUE)    % not trapped
    [result2,Tb,q_connect]=RRT_EXTEND_MULTIPLE(Tb,q_target,step_length)
    if (result2==TRUE)   % connected the two trees
      success = TRUE
      return (success, q_connect, Ta, Tb)
    end
  end
SWAP(Ta,Tb)
end
return (success)  % FALSE
```

RRT_EXTEND_SINGLE

```
q_target = []
q_near = FIND_NEAREST (Ta, q_random)
q_int = LIMIT( q_random, q_near, step_length)   % be careful about angle correction
result = LOCAL_PLANNER(q_near, q_int, step_size)
if (result == TRUE)
   Ta = ADD_NODE_TO_TREE (Ta, q_int)
   q_target = q_int
end
return (result, Ta, q_target)
```

RRT_EXTEND_MULTIPLE

```
q_connect = []
q_near = FIND_NEAREST (Tb, q_target)
q_int = LIMIT( q_target, q_near, step_length)   % be careful about angle correction
q_last = q_near
num_steps = CEIL(NORM(q_target-q_near)/step_length)    % be careful about angle correction
for i=1:num_steps
  result = LOCAL_PLANNER(q_int, q_last, step_size)
  if (result == FALSE)
      return (result, Tb, q_connect)
  end
  Tb = ADD_NODE_TO_TREE (Tb, q_int)
  q_connect = q_int
  if (i < num_steps)
    q_last = q_int
    q_int = LIMIT( q_target, q_int, step_length)
  end
end
return (result, Tb, q_connect)   % result == TRUE
```

LOCAL_PLANNER

```
delta_q = q2 – q1    % be careful about angle correction
num_steps = CEIL(NORM(delta_q)/step_size)   % be careful about angle correction
step = delta_q / num_steps
collision = FALSE
q = q1
for i=1:num_steps
  q = q + step
  collision = collision || CHECK_COLLISION( q, obstacles)
end
success = ~ collision
return (success)
```