Algorithm 4 : phase_1_abs(current node, opposite open list, dir)

```
high_lim = max(gvalue(node in opposite open list))
for node in opposite open list do
  if abstract_isExpandable(node) then
    expandablelist.push(node)
  end if
end for
while expandablelist ≠ {} do
  node = Pop(expandablelist)
  if abstract_isExpandable(node) then
    node →′ closed′
    for neighbour in expand(node) do
      child = Node(neighbour, direction(node), g + 1,′ open′)
      if neighbour == currentnode then
        return gvalue(child)
      end if
      for node in opposite open list do
        opposite nodes list.push(child)
        check for duplicates and replace them if they have lower gvalue
      end for
      if abstract_isExpandable(child) then
        expandablelist.push(child)
      end if
    end for
  end if
end while
return phase_2_abs(current node,opposite open list,high_lim,current dir)
```