Algorithm 1 Graph Creation

```
1: procedure createGraph
2:     i ← 0, x ← 0, y ← 0
3:     w ← width(words[i], font), h ← height(font)

4:     while i <length(words[]) do
5:         if words[i] ="\annotation{" then
6:             vert ← createVertex(x − width(words[i − 1], font)/2, y)
7:             ann ← parseAnnotation(words, i)   ▷ Also points i to next word
8:             setAnnotation(vert, ann)
9:             addtoGraph(vert)
10:            addToList(verticesAbove, vertex) ▷ Ordered by vertices' x-value

11:        else
12:            v1 ← createVertex(x, y)              ▷ Creating vertices at corners
13:            v2 ← createVertex(x + w, y)
14:            v3 ← createVertex(x, y + h)
15:            v4 ← createVertex(x + w, y + h)

16:            addAllToGraph(v1, v2, v3, v4)
17:            createEdgeBetween(v1, v3)
18:            createEdgeBetween(v2, v4)
19:            addToList(verticesAbove, v1, v2)
20:            addToList(verticesBelow, v3, v4)

21:        end if
22:        i ← i + 1
23:        x ← x + w
24:        w ← width(words[i], font)

25:        if (x + w) > rightTextBorder then                ▷ Start new line
26:            x ← 0
27:            y ← y + h

28:            createEdgesBetweenNeighboursIn(verticesAbove) ▷ Order:list
29:            emptyList(verticesAbove)
30:            addContentsToList(verticesBelow, verticesAbove)
31:            emptyList(verticesBelow)
32:        end if
33:    end while
34: end procedure
```

Algorithm 2 Routing Algorithm

```
1: procedure ROUTING
2:     curr ← source

3:     while xvalOf(curr) < rightTextBorder do
4:         new ← null

5:         if backtrack = false then
6:             new ← getAboveNeighbour(curr)
7:         end if

8:         if (new = null) || (isToTopLeftOf(new, previousSource)) then
9:             new ← getRightNeighbour(curr)
10:            backtrack ← false
11:        end if

12:        if new ≠ null then                          ▷ New vertex found
13:            addToPath(new)
14:            curr ← new

15:        else                                        ▷ Initiate backtracking
16:            backtrack ← true
17:            if curr ≠ source then
18:                repeat
19:                    new ← getPreviousPathVertex(curr)
20:                    deleteFromPath(curr)
21:                    old ← curr
22:                    curr ← new
23:                until (curr = source) ∨ (isAboveOf(old, new))
24:            end if

25:            if (curr = source) ∧ (getAboveNeighbour(curr) = null) then
26:                goto End                            ▷ No more options available
27:            end if
28:        end if                                      ▷ Initiate backtracking end
29:    end while
30: end procedure                                      ▷ Label: End
```