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 isAnnotation(words[i]) 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:        i ← i + 1
22:        x ← x + w
23:        w ← width(words[i], font)

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

27:            createEdgesBetweenNeighboursIn(verticesAbove) ▷ Order:list
28:            emptyList(verticesAbove)
29:            addContentsToList(verticesBelow, verticesAbove)
30:            emptyList(verticesBelow)
```

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

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

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

22:             if (curr = source) ∧ (getAboveNeighbour(curr) = null) then
23:                 goto End                            ▷ No more options available
                                                        ▷ Initiate backtracking end
                                                        ▷ Label: End
```