Algorithm 1 The Bellman-Kalaba algorithm

```
1: procedure BellmanKalaba(G, u, l, p)
2:     for all v ∈ V(G) do
3:         l(v) ← ∞
4:     end for
5:     l(u) ← 0
6:     repeat
7:         for i ← 1, n do
8:             min ← l(v_i)
9:             for j ← 1, n do
10:                if min > e(v_i, v_j) + l(v_j) then
11:                    min ← e(v_i, v_j) + l(v_j)
12:                    p(i) ← v_j
13:                end if
14:            end for
15:            l?(i) ← min
16:        end for
17:        changed ← l ≠ l?
18:        l ← l?
19:    until ¬changed
20: end procedure

21: procedure FindPathBK(v, u, p)
22:    if v = u then
23:        Write v
24:    else
25:        w ← v
26:        while w ≠ u do
27:            Write w
28:            w ← p(w)
29:        end while
30:    end if
31: end procedure
```

Algorithm 2 Part 1

```
1: procedure BellmanKalaba(G, u, l, p)
2:     for all v ∈ V(G) do
3:         l(v) ← ∞
4:     end for
5:     l(u) ← 0
6:     repeat
7:         for i ← 1, n do
8:             min ← l(v_i)
9:             for j ← 1, n do
10:                if min > e(v_i, v_j) + l(v_j) then
11:                    min ← e(v_i, v_j) + l(v_j)
12:                                    ▷ For some reason we need to break here!
```

And we need to put some additional text between...

Algorithm 3 Part 2

```
13:                    p(i) ← v_j
14:                end if
15:            end for
16:            l?(i) ← min
17:        end for
18:        changed ← l ≠ l?
19:        l ← l?
20:    until ¬changed
21: end procedure
```

Algorithm 4 A small pseudocode

```
1: s ← 0
2: p ← 0
3: for i ← 1, 10 do
4:     s ← s + i
5:     p ← p + s
6: end for
```