Algorithm 1 Bitonic TSP

Require: Distance function - $dist(p_1, p_2)$
Require: Input array $A[]$ consists of Point objects, with x and y fields.
Ensure: Input array $A[]$ is pre-sorted on x values.

```
1: function LEAST-BITONIC-PATH(A[])                    ▷ Set of x-y points
2:     LBP[n, n − 1] ← dist(n − 1, n)
3:     path[n − 1, n] ← n
4:     for i ← n − 2 to 1 do
5:         min ← ∞
6:         for k ← i + 2 to n do
7:             if min > LBP[i + 1, k] + dist[i, k] then
8:                 min ← LBP[i + 1, k] + dist(i, k)
9:                 mink ← k
10:            end if
11:        end for
12:        LBP[i, i + 1] ← min
13:        path[i, i + 1] ← mink
14:        for j = i + 2 to n do
15:            LBP[i, j] ← LBP[i + 1, j] + d(i, i + 1)
16:            path[i, j] ← i + 1
17:        end for
18:    end for
19:    LBP[1, 1] ← LBP[1, 2] + d(1, 2)
20:    path[1, 1] ← 2
21: end function
```

Algorithm 2 PrintBitonic-TSP

```
1: function PRINTTSP(path, i, j, n)
2:     if n ≤ 0 then
3:         return.
4:     end if
5:     if i ≤ j then
6:         k ← path[i, j]
7:         print(k)
8:         printTSP(path, k, j, n − 1)
9:     else
10:        k ← path[j, i]
11:        printTSP(path, i, k, n − 1)
12:        print(k)
13:    end if
14: end function
```