Function Zero-reachability($\mathcal{S}, d, p$)
input: pVASS $\mathcal{S} = (Q, \gamma, W)$ rozmeru $d \in \mathbb{N}^+$, stav p
output: hodnota *true*, ak pravdepodobnosť, že nedosiahneme z konfigurácie $(p, 1)$ takú konfiguráciu, v ktorej niektoré počítadlo je nula, je väčšia ako nula, hodnota *false* inak

```
begin
    c ← |Q|^2 + (2·|Q| + 1)^d · (d^2 · d! · |Q|^2 + 2 · d^3 · d!^2 · |Q|^4 · (2·|Q| + 1)^d)
    ↪ ← ∅
    foreach (q, α, r) ∈ γ do ↪ ← ↪ ∪ (q, r)
    Prob ← ∅
    foreach (q, r) ∈ ↪ do begin
        T ← 0
        foreach (q′, α, r) ∈ γ do
            if q = q′ then T ← T + W(q′, α, r)
        P(q, r) ← 0
        foreach (q′, α, r′) ∈ γ do
            if q = q′ and r = r′ then begin
                P(q′, α, r′) ← W(q′, α, r′)/T
                P(q, r) ← P(q, r) + P(q′, α, r′)
            end
        Prob ← Prob ∪ P(q, r)
    end
    F_A ← (Q, ↪, Prob)
    SCCs ← TarjansAlgorithm(Q, ↪)
    BSCCs ← ∅
    foreach C ∈ SCCs do begin
        a ← 0
        foreach (q, r) ∈ ↪ do
            if q ∈ C and r ∉ C then begin
                a ← −1
                break
            end
        if a = 0 then BSCCs ← BSCCs ∪ {C}
    end
    foreach C ∈ BSCCs do begin
        i ← 0
        foreach q ∈ C do begin
            A[0][i] ← 1
            x[i] ← q
            i ← i + 1
        end
        b[0] ← 1
        i ← 1
        foreach q ∈ C do begin
            for j ← 0 to |x| − 1 do begin
                a ← 0
                if q = x[j] then begin
                    A[i][j] ← 1
```

```
            a ← 1
        end
        foreach (r, q′) ∈ ↪ do
            if q = q′ and x[j] = r then begin
                if q = x[j] then A[i][j] ← A[i][j] − P(r, q′) else A[i][j] ← −P(r, q′)
                a ← 1
            end
        if a = 0 then A[i][j] ← 0
    end
    b[i] ← 0
    i ← i + 1
end
μ ← LinearEquationsSolver(A, b)
for i ← 0 to |μ| − 1 do μ(x[i]) ← μ[i]
for i ← 0 to d − 1 do begin
    t[i] ← 0
    foreach q ∈ C do begin
        change_i^q ← 0
        foreach (q′, α, r) ∈ γ do
            if q = q′ then change_i^q ← change_i^q + P(q′, α, r) · α[i]
        t[i] ← t[i] + μ(q) · change_i^q
    end
end
C′ ← C ∪ {bottom}
γ′ ← γ
foreach q ∈ C do γ′ ← γ′ ∪ {(q, {0}^d, bottom)}
b ← 0
for i ← 0 to d − 1 do begin
    a ← 0
    foreach q ∈ C do
        if BellmanFordAlgorithm(C′, γ′, i, q, bottom) = |C′| + 1 then begin
            botfin_i(q) ← 0
            a ← −1
        end
        else botfin_i(q) ← |BellmanFordAlgorithm(C′, γ′, i, q, bottom)| + 1
    if a = −1 then decreasing(i) ← true else decreasing(i) ← false
    if t[i] > 0 or (t[i] = 0 and decreasing(i) = false) then diverging(i) ← true else begin
        diverging(i) ← false
        b ← −1
    end
end
if b = 0 then begin
    S′ ← (Q, γ)
    q ← random(Q)
    for i ← 0 to d − 1 do
        if t[i] > 0 then begin
            u[i] ← c
            z[i] ← c + 1
```

```
                end
                else begin
                    u[i] ← botfin_i(q)
                    z[i] ← botfin_i(q)
                end
            if Coverability(S', d, (q, u), (q, z)) = true then begin
                if Coverability(S', d, (p, 1), (q, u)) = true then return true
            end
        end
    end
    return false
end
```