Algorithm 1 DEEPWALK(G, w, d, γ, t)

```
Input: 图$G(V,E)$,窗口大小$w$,嵌入尺寸$d$,对每个节点做随机游走的次数$\gamma$,游走长度$t$
Output: 顶点特征矩阵$\Phi \in \mathbb{R}^{|V|\times d}$
初始化：从$\mathcal{U}^{|V|\times d}$样本$\Phi$
从$V$中构建二叉树$T$
for $i = 0$ to $\gamma$ do
    $\mathcal{O}$ =Shuffle($V$)打乱网络中的节点
    for each $v_i \in \mathcal{O}$ do
        以每个节点为根结点生成随机游走 $\mathcal{W}_{v_i} = RandomWalk(G, v_i, t)$
        使用SkipGram($\Phi, \mathcal{W}_{v_i}, w$)模型利用梯度方法对参数进行更新
    end for
end for
```

Algorithm 2 MergeSort$(A, left, right)$

```
Input: 数组$A[1...n]$，数组下标$left, right$
Output: 递归数组$A[left..right]$
if $left \geq right$ then
    return $A[left..right]$
end if
$mid \leftarrow \lfloor \frac{left+right}{2} \rfloor$
MergeSort($A, left, mid$).
MergeSort($A, mid+1, right$).
Merge($A, left, mid, right$).
return $A[left..right]$
```

Algorithm 3 PROCEDURE Match(s)

```
Input: an intermediate state s; the initial state $s_0$ has $M(s_0) = \varnothing$
Output: the mapping between the two graphs
if $M(s)$ covers all the nodes of $G_2$ then
    OUTPUT $M(s)$
else
    Compute the set $P(s)$ of the pairs candidate for inclusion in $M(s)$
    for each $p$ in $P(s)$ do
        if the feasibility rules succeed for the inclusion of $p$ in $M(s)$ then
            Compute the state s' obtained by adding $p$ to $M(s)$
            CALL Match($s'$)
        end if
    end for
    Restore data structure
end if
```

Algorithm 4 Merge$(A, left, mid, right)$

```
Input: 数组$A[1...n]$，数组下标$left, mid, right$
Output: 递增数组$A[left..right]$
$A'[left..right] \leftarrow A[left..right]$
$i \leftarrow left, j \leftarrow mid + 1, k \leftarrow 0$
while $i \geq mid$ and $j \geq right$ do
    if $A'[i] \geq A'[j]$ then
        $A[left + k] \leftarrow A'[i]$
        $k \leftarrow k + 1, i \leftarrow i + 1$
    else
        $A[left + k] \leftarrow A'[j]$
        $k \leftarrow k + 1, j \leftarrow j + 1$
    end if
end while
if $i \geq mid$ then
    $A[left + k..right] \leftarrow A'[i..mid]$
else
    $A[left + k..right] \leftarrow A'[j..right]$
end if
return $A[left..right]$
```

Algorithm 5 蛮力枚举法

```
Input: 数组$X[1...n]$
Output: 最大子数组之和$S_{max}$
$S_{max} \leftarrow -\infty$
for $l \leftarrow 1$ to $n$ do
    for $r \leftarrow l$ to $n$ do
        $S(l, r) \leftarrow 0$
        for $i \leftarrow l$ to $r$ do
            $S(l, r) \leftarrow S(l, r) + X[i]$
        end for
        $S_{max} \leftarrow \max\{S_{max}, S(l, r)\}$
    end for
end for
return $S_{max}$
```

Algorithm 6 优化枚举法

```
Input: 数组$X[1...n]$
Output: 最大子数组之和$S_{max}$
$S_{max} \leftarrow -\infty$
for $l \leftarrow 1$ to $n$ do
    $S(l, r) \leftarrow 0$
    for $r \leftarrow l$ to $n$ do
        $S \leftarrow S + X[r]$
        $S_{max} \leftarrow \max\{S_{max}, S\}$
    end for
end for
return $S_{max}$
```

Algorithm 7 CrossingSubArray(X,low,mid,high)

Input: 数组$X[1...n]$，数组下标$low, mid, high$

Output: 跨越中点的最大子数组之和S_3

```
$S_{left} \leftarrow -\infty$
$Sum \leftarrow 0$
for $l \leftarrow mid\ downto\ low$ do
    $Sum \leftarrow Sum + X[l]$
    $S_{left} \leftarrow \max\{S_{left}, Sum\}$
end for
$S_{right} \leftarrow -\infty$
$Sum \leftarrow 0$
for $r \leftarrow mid + 1\ to\ high$ do
    $Sum \leftarrow Sum + X[r]$
    $S_{right} \leftarrow \max\{S_{right}, Sum\}$
end for
$S_3 \leftarrow S_{left} + S_{right}$
return $S_3$
```

Algorithm 8 MaxSubArray(X,low,high)

Input: 数组$X[1...n]$，数组下标$low, mid, high$

Output: 最大子数组之和S_{max}

```
if $low = high$ then
    return $X[low]$
else
    $mid \leftarrow \lfloor \frac{low+high}{2} \rfloor$
    $S_1 \leftarrow$ MaxSubArray(X,low,mid)
    $S_2 \leftarrow$ MaxSubArray(X,mid+1,high)
    $S_3 \leftarrow$ CrossingSubArray(X,low,mid,high)
    $S_{max} \leftarrow \max\{S_1, S_2, S_3\}$
    return $S_{max}$
end if
```

```
Function VF2($G_1$,$G_2$)
$Solutions = \varnothing$
$M(s_0) = \varnothing$
$Match(M(s_0), G_2, G_1, Solutions)$
return $Solutions$
```

```
Function $Match(M(s), G_2, G_1, Solutions)$
if $M(s)$ covers all the nodes of $G_1$ then
    $Append(M(s), Solutions)$
else
    $P(s) = GetCantitates(M(s))$
    for each pair in $P(s)$ do
        if $IsFeasible(pair)$ then
            $M(s') = ExtendMatch(M(s), pair)$
            $Match(M(s'), G_2, G_1, Solutions)$
            $BackTrack(M(s'), pair)$
        end if
    end for
end if
```