Algorithm 1 MergeSort$(A, left, right)$

Input: 数组$A[1...n]$，数组下标$left, right$
Output: 递归数组$A[left..right]$

```
1: if left ≥ right then
2:     return A[left..right]
3: end if
4: mid ← ⌊(left+right)/2⌋
5: MergeSort(A, left, mid).
6: MergeSort(A, mid + 1, right).
7: Merge(A, left, mid, right).
8: return A[left..right]
```

Algorithm 2 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

```
1: if M(s) covers all the nodes of G2 then
2:     OUTPUT M(s)
3: else
4:     Compute the set P(s) of the pairs candidate for inclusion in M(s)
5:     for each p in P(s) do
6:         if the feasibility rules succeed for the inclusion of p in M(s) then
7:             Compute the state s' obtained by adding p to M(s)
8:             CALL Match(s')
9:         end if
10:    end for
11:    Restore data structure
12: end if
```

Algorithm 3 Merge$(A, left, mid, right)$

Input: 数组$A[1...n]$，数组下标$left, mid, right$
Output: 递增数组$A[left..right]$

```
1: A'[left..right] ← A[left..right]
2: i ← left, j ← mid + 1, k ← 0
3: while i ≥ mid and j ≥ right do
4:     if A'[i] ≥ A'[j] then
5:         A[left + k] ← A'[i]
6:         k ← k + 1, i ← i + 1
7:     else
8:         A[left + k] ← A'[j]
9:         k ← k + 1, j ← j + 1
10:    end if
11: end while
12: if i ≥ mid then
13:    A[left + k..right] ← A'[i..mid]
14: else
15:    A[left + k..right] ← A'[j..right]
16: end if
17: return A[left..right]
```

Algorithm 4 蛮力枚举法

Input: 数组$X[1...n]$
Output: 最大子数组之和S_{max}

```
1: S_max ← −∞
2: for l ← 1 to n do
3:     for r ← l to n do
4:         S(l, r) ← 0
5:         for i ← l to r do
6:             S(l, r) ← S(l, r) + X[i]
7:         end for
8:         S_max ← max{S_max, S(l, r)}
9:     end for
10: end for
11: return S_max
```

Algorithm 5 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 6 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 7 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

```
1: Function VF2(G_1,G_2)
2: Solutions = ∅
3: M(s_0) = ∅
4: Match(M(s_0), G_2, G_1, Solutions)
5: return Solutions
```

```
1: Function Match(M(s), G_2, G_1, Solutions)
2: if M(s) covers all the nodes of G_1 then
3:     Append(M(s), Solutions)
4: else
5:     P(s) = GetCantitates(M(s))
6:     for each pair in P(s) do
7:         if IsFeasible(pair) then
8:             M(s') = ExtendMatch(M(s), pair)
9:             Match(M(s'), G_2, G_1, Solutions)
10:            BackTrack(M(s'), pair)
11:        end if
12:    end for
13: end if
```