Τοπολογική Διάταξη

Topological-Sort(G)

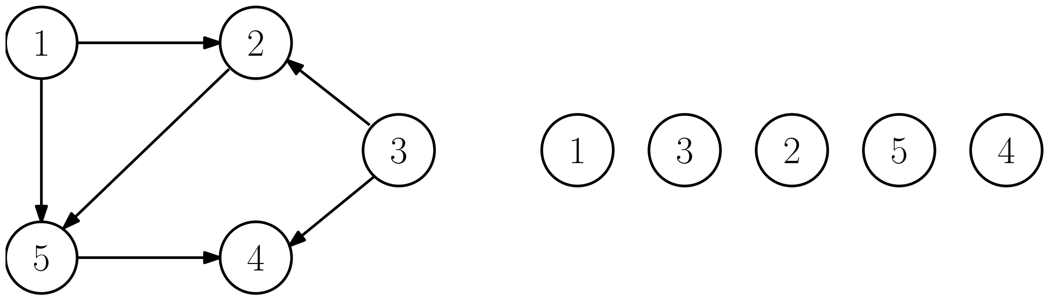
    while $(\exists u \in V :\ indegree(u) == 0\ \&\&\ !mark(u))$

        mark(u)=true

        $\forall v \in Adj(u) : indegree(v) = indegree(v) - 1$

        $Print(u)$

Παράδειγμα:



Πολυπλοκότητα: $O(V + E)$

Topological-Sort(G)

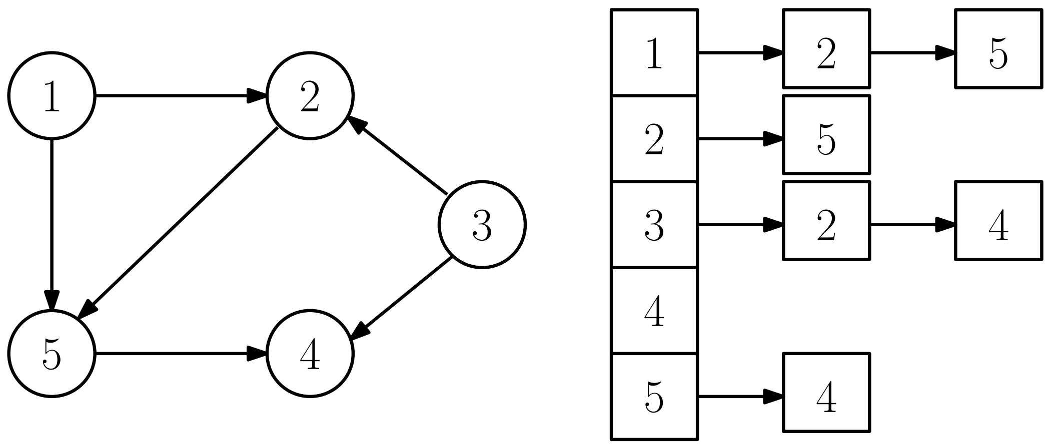
  while $(\exists u \in V : indegree(u) == 0$ && $!mark(u))$

    mark(u)=true

    $\forall v \in Adj(u) : indegree(v) = indegree(v) - 1$

    $Print(u)$

Η αναπαράσταση του γραφήματος:



$Adj()$

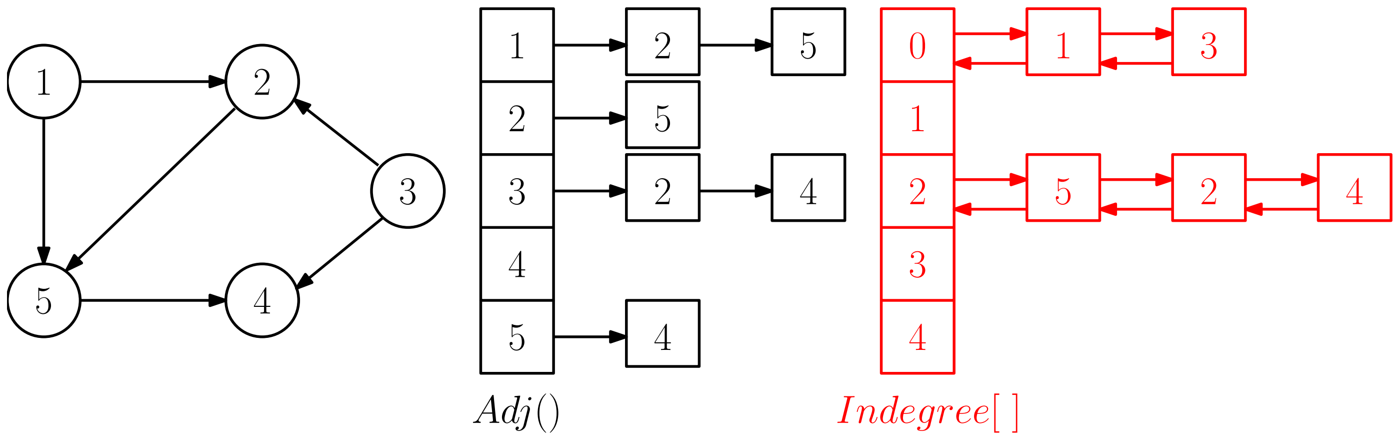
Topological-Sort(G)

  while $(\exists u \in V:\ indegree(u) == 0\ \&\&\ !mark(u))$

    mark(u)=true

    $\forall v \in Adj(u) : indegree(v) = indegree(v) - 1$

    $Print(u)$



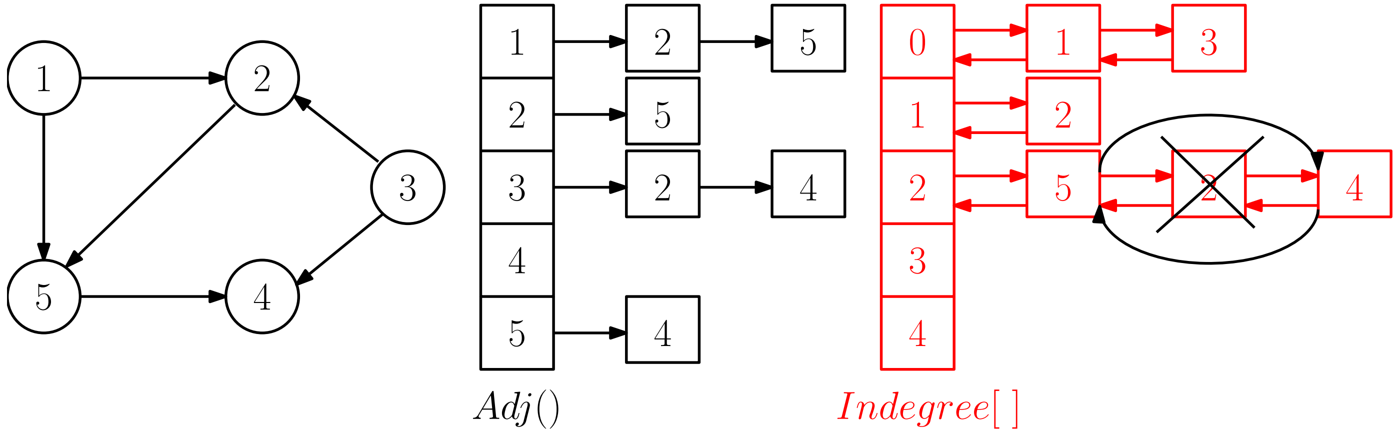
foreach $(edge\ (u, v) \in E)$

  indegree(v)$+= 1$

foreach $(v \in V)$

  add v **in list** $Indegree[indegree(v)]$

```
Topological-Sort(G)
    while (∃u ∈ V : indegree(u) == 0 && !mark(u))
        mark(u)=true
        ∀v ∈ Adj(u) : indegree(v) = indegree(v) − 1
        Print(u)
```



Πολυπλοκότητα: $O(V + E)$

Εφαρμογές της μεθόδου 'Διαίρει-και-Βασίλευε'

Η **insertion sort** μπορεί να εκφραστεί ως μία αναδρομική διαδικασία ως εξής: Για την ταξινόμηση του πίνακα $A[1 \dots n]$, αναδρομικά ταξινόμησε τον πίνακα $A[1 \dots n-1]$ και ακολούθως εισήγαγε το στοιχείο $A[n]$ στον ταξινομημένο πίνακα $A[1 \dots n-1]$.

```
RECURSIVE-INSERTION-SORT(A, n)
    if (n > 1)
        RECURSIVE-INSERTION-SORT (A, n − 1)
        INSERT (A, n)

INSERT(A, k)
    // Insert A[k] to the sorted A[1..k-1]
    key = A[k]
    i = k − 1
    while i > 0 and A[i] > key
        A[i + 1] = A[i]
        i − −
    A[i + 1] = key
```

Πολυπλοκότητα;;;

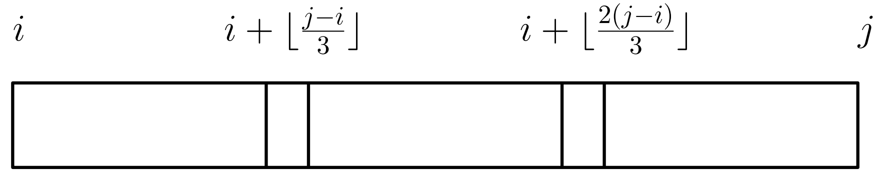
Πολυπλοκότητα:

Έστω $I(n)$ ο χρόνος που χρειάζεται για την εισαγωγή του στοιχείου $A[n]$ στον ταξινομημένο πίνακα $A[1 \ldots n-1]$. Τότε:

$$T(n) = \begin{cases} \Theta(1) & n \leq 1 \\ T(n-1) + I(n) & n > 1 \end{cases}$$

$$\begin{aligned} T(n) = \quad & T(n-1) + I(n) = \\ & T(n-2) + I(n-1) + I(n) = \\ & T(n-3) + I(n-2) + I(n-1) + I(n) = \\ & \ldots \\ & T(1) + I(2) + I(3) + \ldots + I(n) = \\ & \Theta(1) + I(1) + I(2) + I(3) + \ldots + I(n) = \\ & \Theta(1) + \textstyle\sum_{i=2}^{n}(i-1) = \\ & O(n^2) \end{aligned}$$

Τριαδική Αναζήτηση (ternary search):




```
T-SEARCH(A, i, j, key)
    if (i > j)
        return −1
    else
        thirdL = i + ⌊(j−i)/3⌋
        thirdR = i + ⌊2(j−i)/3⌋
        if key == A[thirdL] return thirdL
        else
            if key < A[thirdL] return T-SEARCH(A, i, thirdL − 1, key)
            else
                if key == A[thirdR] return thirdR
                else
                    if key < A[thirdR]
                        return T-SEARCH(A, thirdL + 1, thirdR − 1, key)
                    else
                        return T-SEARCH(A, thirdR + 1, j, key)
```

Πολυπλοκότητα:

Υποθέτουμε ότι $n = 3^k$. Τότε, για την περίπτωση μιας μη επιτυχημένης αναζήτησης:

$$T(n) = \begin{cases} \Theta(1) & n \leq 2 \\ T(\frac{n}{3}) + 2 & n > 2 \end{cases}$$

$$\begin{aligned} T(n) = \quad & T(\tfrac{n}{3}) + 2 = \\ & T(\tfrac{n}{9}) + 2 + 2 = \\ & \ldots \\ & T(\tfrac{n}{3^k}) + 2k = \\ & \Theta(1) + 2\log_3 n = \\ & \Theta(1) + \tfrac{2}{log_2 3}\log_2 n = \\ & \Theta(\log_2 n) \end{aligned}$$

Να αποδειχθεί ότι:

$$T(n) = \begin{cases} \Theta(1) & n \leq 1 \\ mT(\frac{n}{2}) + an^2 + 2 & n > 1 \end{cases} \quad \Rightarrow \quad T(n) = O(n^{\log m})$$

Απόδειξη

$$\begin{aligned} T(n) = \quad & mT(\tfrac{n}{2}) + an^2 = \\ & m^2T(\tfrac{n}{2^2}) + an^2(1 + \tfrac{m}{2^2}) = \\ & \ldots \\ & m^{\log n}T(\tfrac{n}{2^{\log n}}) + an^2 \textstyle\sum_{i=0}^{\log n - 1} m^i(\tfrac{1}{2^i})^2 (*) = \\ & m^{\log n}\Theta(1) + an^2 \textstyle\sum_{i=0}^{\log n - 1} (\tfrac{m}{4})^i \leq \\ & c_1 m^{\log n} + an^2(\tfrac{m}{4})^{\log n} = \\ & c_1 m^{\log n} + an^2 \tfrac{m^{\log n}}{2^{2\log n}} = \\ & c_1 m^{\log n} + an^2 \tfrac{m^{\log n}}{n^2} = \\ & cm^{\log n} = cn^{\log m} \end{aligned}$$

$$(*)\, \frac{m^i}{2^{i^2}} = \frac{2^{\log m^i}}{2^{i^2}} = (\frac{2^{\log m}}{2^2})^i = (2^{\log m - 2})^i = (\frac{m}{4})^i$$