WRAP UP FFT
DFS FOR TOPOLOGICAL SORT

Feb 5, 2019
Polynomial multiplication

\[(1 + 2x + x^2) \cdot (2 + x + 3x^2)\]

\[= 2 + (2+1) \cdot x + (3+2+2) x^2 + (+) x^3 + 3x^4\]

given \(A(x) = a_0 + a_1 x + \ldots + a_n x^n\)

\(B(x) = b_0 + b_1 x + \ldots + b_n x^n\)

want to find coefficients of

\(C(x) = A(x) \cdot B(x)\)

\(C(x) = c_0 + c_1 x + \ldots + c_{2n} x^{2n}\)

\[c_0 = a_0 b_0\]

\[c_1 = a_0 b_1 + a_1 b_0\]

\[c_2 = a_2 b_0 + a_1 b_1 + a_0 b_2\]

\[\vdots\]

\[c_n = a_n b_0 + a_{n-1} b_1 + \ldots + a_0 b_n\]

\[c_{n+1} = a_n b_1 + \ldots\]

\(\Theta(n^2)\)
given \[ A(x) = a_0 + a_1 x + \ldots + a_n x^n \]
\[ B(x) = b_0 + b_1 x + \ldots + b_n x^n \]

want to find coefficients of \[ C(x) = A(x) \cdot B(x) \]
\[ C(x) = c_0 + c_1 x + \ldots + c_{2n} x^{2n} \]

**Inverse FFT** \( C \) is a polynomial of degree \( N-1 \) where \( N \) is a power of 2.

given \[ C(1), C(w), \ldots, C(w^{N-1}) \]
where \( 1, w, \ldots, w^{N-1} \) are \( N \)-th roots of unity
in \( O(N \log N) \) time find coefficients of \( C \)

given \[ A(1), A(w), \ldots, A(w^{N-1}) \]
\[ B(1), B(w), \ldots, B(w^{N-1}) \]
then in \( O(N) \) time can compute
\[ C(1) = A(1) \cdot B(1), C(w) = A(w) \cdot B(w), \ldots, C(w^{N-1}) = A(w^{N-1}) \cdot B(w^{N-1}) \]

with FFT given \( A, B \)
compute \[ A(1), \ldots, A(w^{N-1}), B(1), \ldots, B(w^{N-1}) \] in \( O(N \log N) \) time
Input: \[ A(x) = a_0 + a_1x + \ldots + a_nx^n \]
\[ B(x) = b_0 + b_1x + \ldots + b_nx^n \]

Let \( N \) be a power of \( 2 \geq 2^n+1 \) and \( \leq 4^n \)

\[ A(z), A(w), \rightarrow A(w^{N-1}) = \text{FFT}(A, N) \]
\[ B(z), B(w), \rightarrow B(w^{N-1}) = \text{FFT}(B, N) \]
\[ C(z) = A(2) \cdot B(2) \]
\[ \vdots \]
\[ C(w^{N-1}) = A(w^{N-1}) \cdot B(w^{N-1}) \]

\[ c_0 \ldots c_{N-1} = \text{IFFT}(c(1), \ldots c(w^{N-1})) \]

\[ \mathcal{O}(N \log N) = \mathcal{O}(n \log n) \]
Graph

Def: undirected graph

Vertices = \{ A, B, C, D, E \}
Edges = \{ (A, B), (A, C), (C) \}

Directed graph

Vertices = \{ A, B, C, D, E \}
Edges = \{ A \to B, B \to C \}

Path
A \to B \to D \to E

Reachability
u \to v

V is reachable from u

Connected component

Strongly connected component
If a directed graph has a cycle, then it does not have a topological sort.
visited = boolean array indexed by vertices initialized to False
L = empty list

def explore (v):
    visited[v] = True
    for each neighbor w of v
        if not visited[w]:
            explore(w)
    L = [v] + L

def DFS:
    for each vertex v
        if not visited[v]:
            explore(v)

(1) Runs in time $O(V + E)$

Trace execution of algorithm:
Consider nodes $v$ in order in which $\text{explore}(v)$ terminates.
Reverse of that order is a topological sort if no cycles
Suppose \( G \) has no cycles. Algorithm outputs a valid topological sort.

\[
\begin{align*}
\text{explore}(v) & \quad \text{is called when} \quad \text{visited}[v] = F \\
\text{explore}(w) & \quad \text{is called inside } \text{explore}(v) \\
\text{explore}(w) & \quad \text{terminates before } \text{explore}(v)
\end{align*}
\]

\[
\begin{align*}
\text{explore}(w) & \quad \text{is called when} \quad \text{visited}[v] = F \\
\text{can } \text{explore}(v) \text{ be called inside } \text{execution of} \\
\text{explore}(w) & \quad \text{?} \\
\text{NO} & \\
\text{explore}(v) & \quad \text{is called (and so it terminates)} \\
\text{after } \text{explore}(w) & \quad \text{terminates}
\end{align*}
\]