Lecture 22:
• Cliques (wrap up)
• Weighted Graphs

Announcements
• HW-5 due
• Exam 2 Thurs
• Project Step 2 out (due Thurs 11/17)
• HW-6 out soon

---

Bron-Kerbosch Algorithm for Enumerating Cliques

Basic Idea:
• build up cliques from smaller complete subgraphs
• backtrack to find more

Notation used in the algorithm:
• $P$ denotes the “possible” vertices ... starts as all vertices
• $R$ is the (possibly non-maximal) clique ... starts empty
• $X$ denotes the “excluded” vertices .. starts as empty
• $R$, $P$, and $X$ are each disjoint
• At each step, add a candidate vertex $v$ from $P$ and remove it from $X$
Bron-Kerbosch Algorithm for Enumerating Cliques

Algorithm: MaximalCliques(P, R, X)

begin
if $P = \emptyset$ and $X = \emptyset$ then
    $R$ is a maximal clique
foreach $v \in P$ do
    MaximalCliques($P \cap \text{Adj}(v)$, $R \cup \{v\}$, $X \cap \text{Adj}(v)$)
    $P \leftarrow P - \{v\}$
    $X \leftarrow X \cup \{v\}$

Check In: Trace the algorithm on the graph:

CLIQUE is NP-Complete

Decision Problem: Does $G$ contain a $k$-clique? ...a clique with $k$ nodes

CLIQUE is in NP: Given collection of nodes, check if it is a $k$-clique
- this can be done in PTIME ... How?

CLIQUE is NP-Hard: 3-SAT $\leq_P$ CLIQUE
- Let $\phi$ be a 3-SAT formula with $k$ clauses
- Every literal (occurrence) becomes a node in $G$ (labeled by the literal)
- $G$ contains all non-“forbidden” edges
  1). edges between nodes within a clause are forbidden
  2). edges between inconsistent nodes are forbidden (e.g., $(x_1, \bar{x}_1)$)
- Claim: $\phi$ is satisfiable iff $G$ has a $k$-clique
CLIQUE is NP-Complete

Example: \( \phi = (x_1 \lor \bar{x}_2 \lor x_3) \land (\bar{x}_1 \lor x_2 \lor \bar{x}_3) \land (\bar{x}_1 \lor x_2 \lor \bar{x}_3) \)

Each literal occurrence becomes a node:

\[
\begin{array}{ccc}
  \bar{x}_1 & x_2 & \bar{x}_2 \\
  x_1 & x_3 & x_2 \\
  \bar{x}_1 & \bar{x}_3 & \bar{x}_2 \\
\end{array}
\]

Clause 1  Clause 2  Clause 3

Add in the non-forbidden edges (only some shown below)

Claim: \( \phi \) is satisfiable iff \( G \) has a \( k \)-clique

If \( G \) has a \( k \)-clique:
- the \( k \)-clique must have 1 node per clause (within-clause edges forbidden)
- assigning each node literal to true gives a satisfying assignment

If \( \phi \) is satisfiable:
- it must have one true literal per clause
- corresponding nodes in \( G \) must form a \( k \)-clique
Weighted Graphs

Edge labels as “weights” (or “distances”)

- Examples include road networks and shipment route costs
- Goal is to find shortest paths based on distance
- We’ll start with the single-source shortest path problem

A weighted graph $G = (V, E)$ has a weight (distance) function $w : E \to \mathbb{R}$

- we assume directed graphs
- and start by considering only non-negative integer weights

The path weight $w(p)$ of $p = (v_1, v_2, \ldots, v_n)$ is the sum of edge weights:

$$
\sum_{i=1}^{n-1} w(v_i, v_{i+1})
$$

The shortest path weight $\delta(u, v)$ from $u$ to $v$ is either:

(i). the minimum path weight $w(p)$ of the paths $p$ from $u$ to $v$; or
(ii). $\infty$ if no path exists from $u$ and $v$

A shortest path from $u$ to $v$ is any path $p$ with weight $w(p) = \delta(u, v)$

Note this extends the BFS approach for finding shortest (length) paths:

- specifically, by assuming each edge has weight 1
- BFS won’t find shortest weighted paths in general