Algorithms, TAU

- Uri Zwick, Haim Kaplan, Tal, Jad
Topics

• Basic graphs algorithms
  • Graphs search/traversal
  • Minimum spanning trees
  • Shortest path algorithms

• Dynamic programming

• Linear programming

• Back to graphs: Flows, matchings
Graphs

• Recall: A graph $G$ is a pair $(V, E)$
• $V = \{1, 2, 3, 4, 5\}$ and $E = \{(1, 2), (1, 5), (2, 5), (2, 4), (4, 5), (2, 3), (3, 4)\}$
• Can be directed/undirected, weighted
Graphs/applications

• We can think of almost any dataset as graph/s:
  • Facebook
  • Road maps
  • Emails
Put an edge \((x,y)\) between patient \(x\) and patient \(y\) if \(x\) and \(y\) were in contact.

Put an edge between a patient \(x\) and a place \(z\) if \(x\) was in \(z\).
Representations of graphs (1)

• **Adjacency lists:**

For every $v$, $G. Adj[v]$ is the list of the vertices $u$ such that $(v, u) \in E$
A directed graph

• Adjacency lists:
  For every $v$, $G.\text{Adj}[v]$ is the list of the vertices $u$ such that $(v, u) \in E$
Adjacency lists (pros and cons)

• Size $O(n + m)$, $|V| = n$, $|E| = m$
• Extends easily to weighted graphs

• No fast way to check if $(u, v) \in E$
• **Adjacency Matrix:**

Vertices are numbered 1, 2, ..., |V|, use matrix $A$, such that $A(i, j) = 1$ iff $(i, j) \in E$
Directed graph

• **Adjacency Matrix:**

Vertices are numbered $1, 2, \ldots, |V|$, use matrix $A$, such that $A(i, j) = 1$ iff $(i, j) \in E$
Adjacency matrix (pros and cons)

• Can find out if an edge \((u, v)\) is in the graph in \(O(1)\) time
• Extend easily to weighted graphs

• Takes \(O(n^2)\) space even if the graph is sparse \((m = o(n^2))\)
Two algorithms to **search** a graph

- **Search** = Visit/reach all vertices + accumulate some additional important information as you go