VE203 Final Review

Presenter: Yue & Yinchen

2023/7/30

Outline

- Master Theorem
- Partial order
- Graph theory
  - Connectivity
  - Bipartition
  - Matching
    - Hall’s Theorem
    - Kőnig-Egerváry Theorem
  - Tree
  - algorithm

Master Theorem - Notation

	Notation	Formal definition	Limit definition
Asymptotic upper bound	$f(n) = O(g(n))$	exist positive constants c and n_0 such that $0 \le f(n) \le cg(n)$ for all $n \ge n_0$	$\lim_{n\to\infty} sup\left(\frac{f(n)}{g(n)}\right) < \infty$
Asymptotic lower bound	$f(n) = \Omega(g(n))$	exist positive constants c and n_0 such that $0 \le cg(n) \le f(n)$ for all $n \ge n_0$	$\lim_{n\to\infty} inf\left(\frac{f(n)}{g(n)}\right) > 0$
Asymptotic tight bound	$f(n) = \Theta(g(n))$	exist positive constants c1, c2, and n_0 such that $0 \le c1g(n) \le f(n) \le c2g(n)$ for all $n \ge n_0$	The two above

Stirling approximation: $n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$

Given $f(n) = 1 + \cos(\pi n/2)$ and $g(n) = 1 + \sin(\pi n/2)$, then (Summer 2021)

- ☐ f(n) = O(g(n))
- ☐ g(n) = O(f(n))
- ☐ f(n) = Θ(g(n))
- ☐ g(n) = Θ(f(n))

	Notation	Formal definition	Limit definition
Asymptotic upper bound	$f(n) = O(g(n))$	exist positive constants c and n_0 such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0$	$\lim_{n\to\infty} sup\left(\frac{f(n)}{g(n)}\right) < \infty$
Asymptotic lower bound	$f(n) = \Omega(g(n))$	exist positive constants c and n_0 such that $0 \leq cg(n) \leq f(n)$ for all $n \geq n_0$	$\lim_{n\to\infty} inf\left(\frac{f(n)}{g(n)}\right) > 0$
Asymptotic tight bound	$f(n) = \Theta(g(n))$	exist positive constants c1, c2, and n_0 such that $0 \leq c1g(n) \leq f(n) \leq c2g(n)$ for all $n \geq n_0$	The two above

Master Theorem

If $T(n) = aT(n/b) + f(n)$ (for constants $a \geq 1$, $b > 1$), then

1. $T(n) = \Theta(n^{\log_b a})$ if $f(n) = O(n^{\log_b a-\varepsilon})$ for some constant $\varepsilon > 0$.
2. $T(n) = \Theta(n^{\log_b a} \lg n)$ if $f(n) = \Theta(n^{\log_b a})$.
3. $T(n) = \Theta(f(n))$, if $f(n) = \Omega(n^{\log_b a+\varepsilon})$ for some constant $\varepsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n (regularity condition).

Exercise 5.2 (2 pts) Let $a \geq 1$ and $b > 1$ be constants, and $T(n)$ satisfies the recurrence

$$T(n) = aT(n/b) + f(n)$$

Show that if $f(n) = \Theta(n^{\log_b a} \lg^k n)$, $k \geq 0$, then the recurrence has solution $T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$. Assume n is integer power of b for simplicity.

If $T(n) = aT(n/b) + f(n)$ *(for constants* $a \geq 1$, $b > 1$*), then*

1. $T(n) = \Theta(n^{\log_b a})$ *if* $f(n) = O(n^{\log_b a - \varepsilon})$ *for some constant* $\varepsilon > 0$.
2. $T(n) = \Theta(n^{\log_b a} \lg n)$ *if* $f(n) = \Theta(n^{\log_b a})$.
3. $T(n) = \Theta(f(n))$, *if* $f(n) = \Omega(n^{\log_b a + \varepsilon})$ *for some constant* $\varepsilon > 0$, *and if* $af(n/b) \leq cf(n)$ *for some constant* $c < 1$ *and all sufficiently large* n *(regularity condition).*

Exercise:

1. $T(n) = kT\left(\frac{n}{2}\right) + \theta(n^2)$
2. $T(n) = T(\sqrt{n}) + \lg(n)$

Partial Order

Poset $(P, \leq)$

- Reflexive: $\forall x \in P, x \leq x$
- Antisymmetric: $\forall x, y \in P, x \leq y \wedge y \leq x \rightarrow x = y$
- Transitive: $\forall x, y, z \in P, x \leq y \wedge y \leq z \rightarrow x \leq z$

(maybe for some x, y no relation between them)

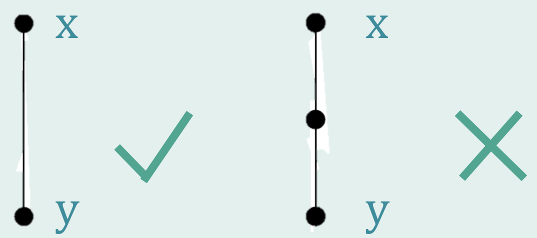
\+ dichotomy $\forall x, y \in P$ ($x \leq y$ or $y \leq x$)
(any two elements are comparable)

$\Rightarrow$ Linear/Total order

\+ original order relation kept

$\Rightarrow$ Linear extention

y cover x

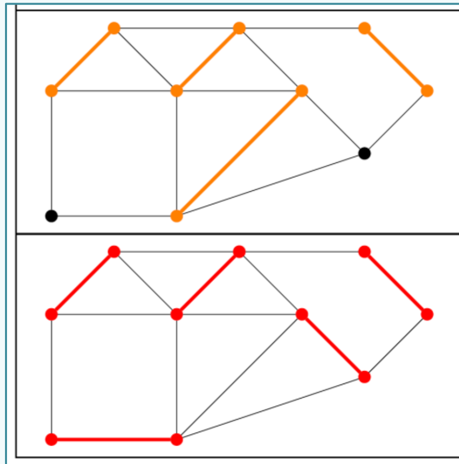


Maximal & maximum ?

Minimal/maximal: (among those who comparable with it) no larger/smaller (may not unique), can't be extended

Compare with every element

Minimum/maximum(unique if exist)



- If $z \in P$ but $\nexists x \in P$ such that $z < x$, then z is a ***maximal element***.
- If $x \leq z$ for all $x \in P$, then z is the ***maximum element***.

Definition

A chain C in P is

- ***maximal*** if there exists no chain C' such that $C \subsetneq C'$.
- ***maximum*** if for all chain C', $|C| \not< |C'|$.

Definition

- A matching M is maximal if there is no matching M' such that $M \subsetneq M$
- A matching M is maximum if there is no matching M' such that $|M| < |M'|$.
- A ***perfect matching*** is a matching M such that every vertex of G is incident with an edge in M.

Definition

A maximal connected subgraph of G is a subgraph that is connected and is ***not*** contained in any other connected subgraph of G.

Chain & Antichain

Chain: a subset of comparable elements (a complete graph)

Antichain: a subset of incomparable elements

- **Maximal**: can't be extended
- **Maximum**: max length

Height: maximum size of chain

Width: maximum size of antichain

Exercise

Given a finite set S, then

- ☐ $(2^S, \preceq)$ is a poset, where $A \preceq B$ iff $|A| \leq |B|$ for $A, B \subset S$.
- ☐ The width of $(2^S, \subset)$ is at least $|S|$.
- ☐ The height of $(2^S, \supset)$ is at most $|S|$.
- ☐ The height of $(2^S, \supset)$ is at least $|S|$.

Dilworth's Theorem

k: least integer that P is a union of k chains

m: size of largest antichain of P

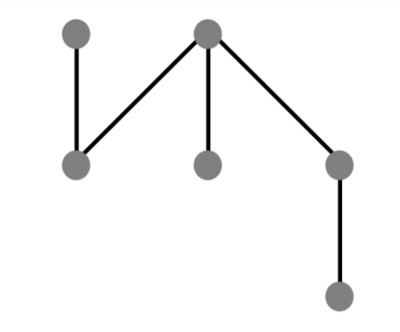
Dilworth Theorem: k=m

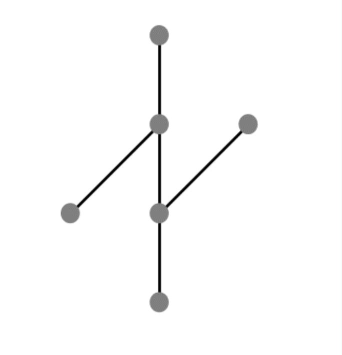
"dual":

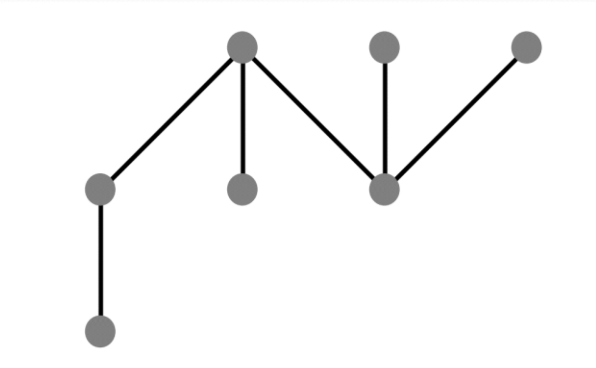
k: least integer that P is a union of k antichains

m: size of largest chain

Mirsky's Theorem: k=m







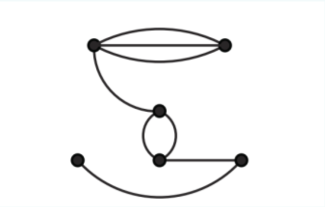
Example:

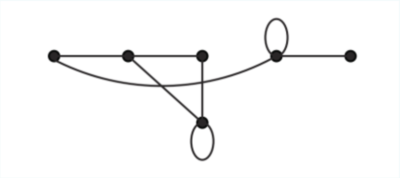
width of the graph on the right?

Given a finite poset, would removing a maximal chain decreases the width of the poset?

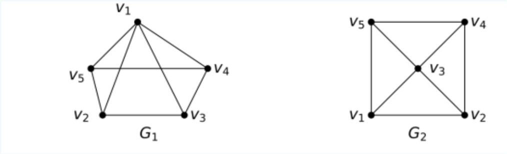
Basic Graph Definitions

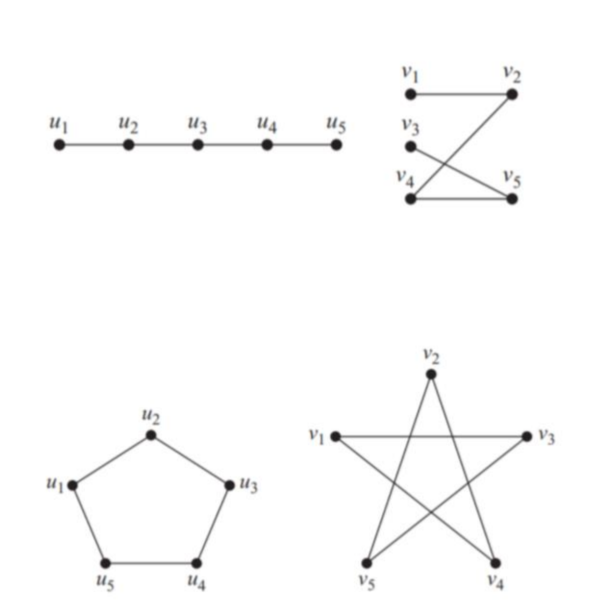
- Loop, parallel, simple graph





- Isomorphism G ≅ H
  - Bijection from V(G) -> V(H) that keep the edges
  - Equivalence relation





- Complement: $uv \in E(\overline{G})$ iff $uv \notin E(G)$.
- Complete graph(K_n)/**Clique**: pairwise adjacent, simple graph
- Path(P_n): no repeat vertices
- Cycle graph(C_n): Path + $e_n = v_n v_1$
- Induced subgraph: every edge: both ends in the subgraph => edge in subgraph
- Bipartition: V(G) => (A, B), no edge has both ends in A or B

Double Counting

- Relation between Degree & Edge
- Handshaking lemma
- Exercise:
  - In any graph with at least two nodes, there are at least two nodes of the same degree
  - Is it true that a finite graph having exactly two vertices of odd degree must contain a path from one to the other? Give a proof or a counterexample.
  - Theorem: Consider a 6-clique where every edge is colored red or blue. The graph contains a red triangle or a blue triangle

For all finite graph $G = (V, E)$,

$$\sum_{v \in V} \deg(v) = 2|E|$$

Connectivity

Path: the vertices can be ordered as $v_1, v_2, \ldots, v_k$ and edges can be ordered as $e_1, e_2, \ldots, e_{n-1}$ that $e_i = v_i v_{i+1}$

Walk: a sequence of (not necessarily distinct) vertices $v_1, v_2, \ldots, v_k$ such that $v_i v_{i+1} \in E$ for $i = 1, 2, \ldots, k-1$.

- Distinct Vertices => path
- $v_0 = v_n$ => closed

Length: number of edges

Theorem: If there is a walk from u to v, then there is a path from u to v.

Connected: A graph G is connected if for all u, v ∈ V(G), there is a walk from u to v

(intuitively, one can pick up an entire graph by grabbing just one vertex)

G is **disconnected** iff there is a partition {X,Y } of V(G) such that no edge has an end in X and an end in Y

Each **maximal connected** piece of a graph is called a connected **component**

Which of the following statements about graphs are correct?

- ☐ $C5$ is self-complementary.
- ☐ $P4$ is self-complementary.
- ☐ $K2,2$ is induced in $C4$.
- ☐ $C1$ is induced in $K5$.

Bridge

If the deletion of a edge/vertex v from G causes the number of components to increase, then v is called a **cut edge**/vertex

- *either e is a cut-edge and* $\text{comp}(G - e) = \text{comp}(G) + 1$;
- *or e is NOT a cut-edge and* $\text{comp}(G - e) = \text{comp}(G)$.

an edge e is a bridge of G if and only if e lies on no cycle of G

Bipartition & Matching

Matching:

- A subset of edges
- No common vertices

Or each node has either zero or one edge incident to it.

Perfect matching: every vertex of G is incident with an edge in M.

Theorem

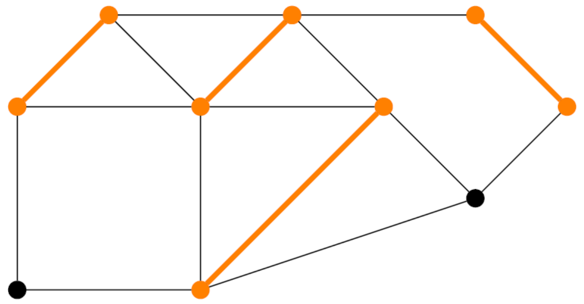
For every graph G, TFAE

(i) *G is bipartite.*

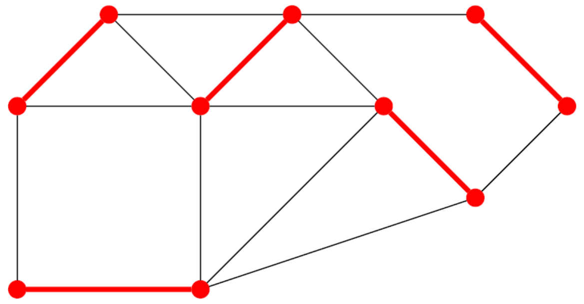
(ii) *G has no cycle of odd length.*

(iii) *G has no closed walk of odd length.*

(iiii) *G has no induced cycle of odd length.*



The matching M is maximal in G



The matching M' is maximal, maximum, and perfect in G

Matching

Hall's theorem

Let G be a finite bipartite graph with bipartition (A, B).

There exists a matching covering A iff $|N(X)| \geq |X| \ \forall X \subseteq A$ **(Hall's condition)**

- If $X \subset V(G)$, the ***neighbors*** of X is $N(X) := \{v \in V(G) \setminus X \mid v$ is adjacent to a vertex in $X\}$
- The edges $S \subset E(G)$ ***covers*** $X \subset V(G)$ if every $x \in X$ is incident to some $e \in S$.

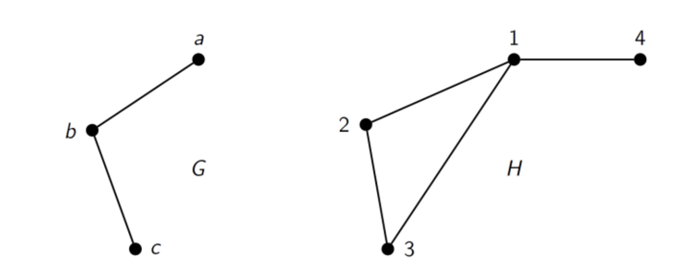
Exercise 7 (10 Marks)
Let G be a bipartite graph with bipartation (A, B), and G has no isolated vertices. If the minimum degree of vertices in A is no less than the maximum degree of vertices in B, show that there exists a matching covering A.

König-Egeváry Theorem

The matching number (i.e., size of a largest matching(edge set)) is equal to the vertex cover number (i.e., size of a smallest vertex cover) for a bipartite graph.

- Prove that a k-regular bipartite graph has a perfect matching (k>=1)
  k-regular: deg(v) = k for all v in V(G)

Homomorphism



Definition:

- simple graphs G and H
- a map from V(G) to V(H) which takes edges to edges

=> nonedge can be mapped to anything

=> There is an injective homomorphism from G to H (i.e., one that never maps distir vertices to one vertex) if and only if G is a subgraph of H.

If a homomorphism f : G → H is a bijection whose inverse function is also a graph homomorphism, then f is a graph isomorphism. This is same as the Definition in slides

If there is a homomorphism G → H and another homomorphism H → G. Are the maps surjective or injective?

Tree

forest: no cycles => comp(G) = |V(G)| – |E(G)|.

tree: any two of {connected, no cycles, |V(T)| = |E(T)| + 1}

spanning tree of G = subgraph + tree + contain all vertices

Theorem

Let T be a graph with n vertices. TFAE

(i) *T is a tree;*

(ii) *T contains no cycles, and has $n-1$ edges;*

(iii) *T is connected, and has $n-1$ edges;*

(iv) *T is connected, and each edge is a bridge;*

(v) *any two vertices of T are connected by exactly one path;*

(vi) *T contains no cycles, but the addition of any new edge creates exactly one cycle.*

Theorem:

For connected graph with |V(G)|>2,

- subgraph H is a spanning tree
- Iff H is a minimal connected graph with V(T) =V(G)
- Iff H is a maximal subgraph without cycles

Exercise 5 (10 pts) Given a graph G. Show that an edge $e \in E(G)$ is a cut-edge iff e is contained in every spanning tree of G.

Which of the following graph is a tree?

- ☐ A simple graph with a unique path between any 2 vertices.
- ☐ A connected simple graph in which every edge is a cut edge.
- ☐ A connected simple graph with n vertices and $n - 1$ edges.
- ☐ A connected simple graph with no cycle.

G is a finite graph

(10 pts) Let T be a spanning tree of G, $e \in E(T)$, and $f \in E(G) - E(T)$. Let $P \subset T$ be the unique path connecting the ends of f, and $e \in P$. Show that $T - e + f$ is a spanning tree.

(ii) (10 pts) Given two **distinct** cycles $C, D \subset G$, and an edge $e \in C \cap D$. Show that $C \cup D - e$ contains a cycle.

Algorithm

Kruskal's Algorithm

Aim: Find a minimum-cost tree

Greedy approach

- Maintain a "forest," or a group of trees /disjoint sets
- Iteratively select cheapest edge in graph
    - If adding the edge forms a cycle, don't add it
    - Otherwise, add it to the forest
- Continue until all vertices are part of the same set

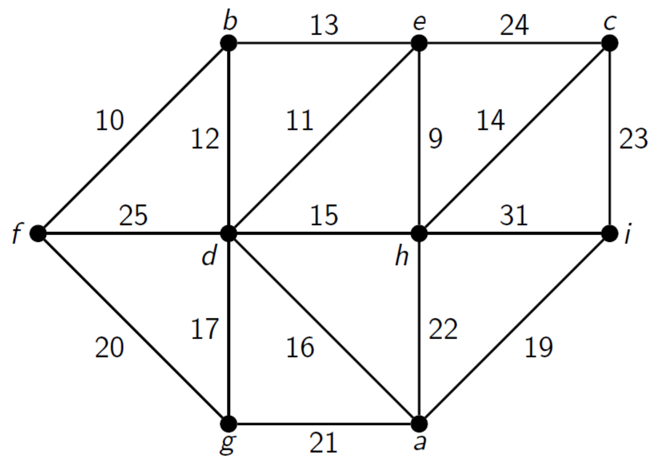
Dijkstra's Algorithm

Aim: shortest path spanning tree for a certain vertex

Greedy Approach

- Separate vertices into two groups:
    - "Innies": vertices that are present in your partial spenning tree at any point in time
    - "Outies" : the other vertices
- Iteratively add **nearest outie**, converting to an innies

Given the following weighted graph G:



- Find a minimum-weight spanning tree using Kruskal’s Algorithm
- Given the root vertex a, find a shortest path spanning tree using Dijkstra’s Algorithm

End

~~QAQ~~Q&A

Reference

- Summer 2021 final exam
- [Kőnig-Egerváry theorem (omath.club)]