MATH 340 - Discrete Structures 2

McGill University - Winter 2013

Last Updated: April 19, 2013

Contents

Information

- Instructor: Bruce Shepherd
- LaTeX: Ehsan Kia
- Notes: Catherine Hilgers

1 Summary of Graph Theory Terms

A (simple) graph G is an ordered pair $(V(G), E(G))$, sometimes written (V, E), where $V(G)$ is a finite set of vertices (aka nodes), and $E(G)$ is a finite set of edges.

Each edge is of the form $\{u, v\}$ sometimes written uv, where $u \neq v$ are two vertices that are the end points of the edge. An edge $e \in E$ is *incident* to a vertex $v \in V$ if $e = (u, v)$ for some $u \in V$. A vertex $v \in V$ is called *adjacent* to a vertex $u \in V$ if $(u, v) \in E$.

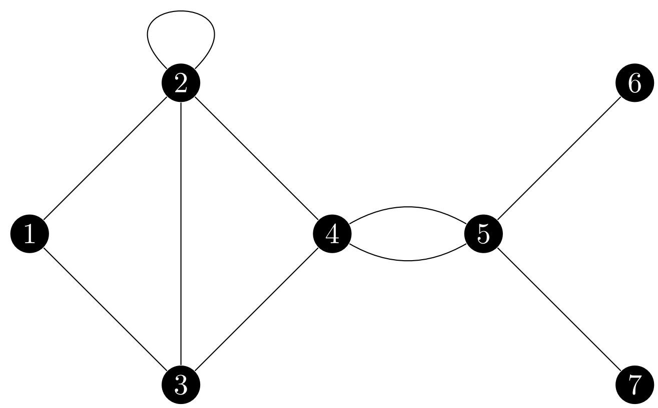
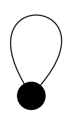
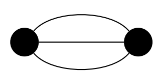


Figure 1: Example of a graph where $V = \{1, 2, 3, 4, 5, 6, 7\}$

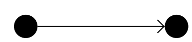
Note: simple, undirected graph mean that we have no:



(a) Loops



(b) Multiple edges



(c) Direction

Suppose we have $H = (V(H), E(H))$ such that:

i) $V(H) \subseteq V(G)$

ii) $E(H) \subseteq E(G)$

iii) $\forall\ e = (u, v) \in E(H) : u, v \in V(H)$

Then, H is a *subgraph* of G.

Given a set $S \subset V$, we define the *subgraph induced by* S to be the graph denoted by $G[S]$ to be a subgraph of G whose vertex set is S and whose edge set is the set of edges with both ends in S.

Similarly, for $F \subset E$, define the subgraph induced by F, denoted $G[F]$, to be the subgraph of G whose edge set is F and whose vertex set is the set of all endpoints in F.

1.1 Terminology

The *degree* of a vertex v is the number of edges of which it is an endpoint, denoted by $deg_G(v)$.

A *walk* of a graph G is a sequence of alternating vertices and edges $v_0e_1v_1e_2...v_{n-1}e_nv_n$ such that e_i is incident to v_{i-1} and v_i, $\forall\ i = 1, ..., n$, where n is the length of the walk.

A *trail* is a walk in which the edges are distinct.

A *path* is a trail in which vertices are distinct.

A *cycle* is a trail of length at least 1 in which the vertices are distinct, except v_0 and v_n which are the same.

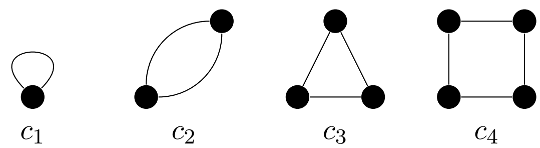


Figure 2: Cycles of size 1 to 4.

A graph is *connected* if $\exists$ a path between any two vertices. Else, it's disconnected.

A *component* of G is a maximal connected subgraph.

1.2 Special Graphs

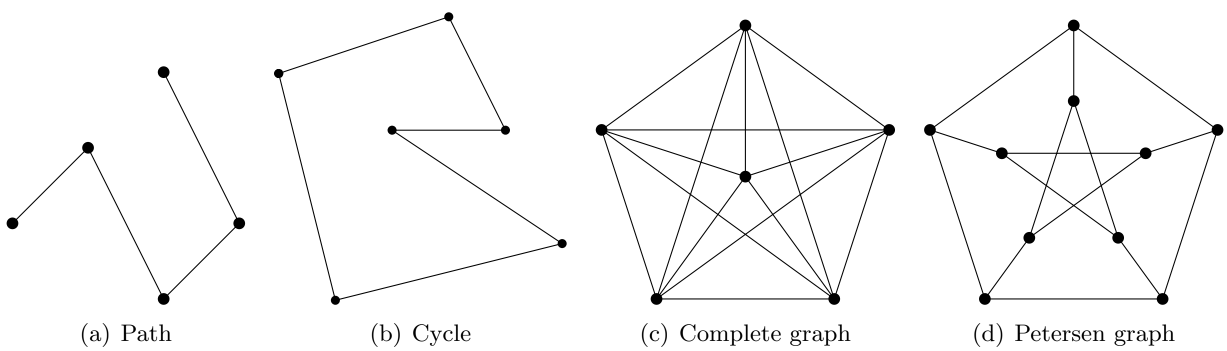


Figure 3: Examples of simple graphs

A *tree* is a connected graph with no cycles (Figure 4).

A graph G is *bipartite* if $\exists$ a partition (X, Y) of $V(G)$ such that for every edge $e \in E(G)$, e has one endpoint in X and the other in Y. X and Y are called the *parts* of G and (X, Y) is called the bipartition.

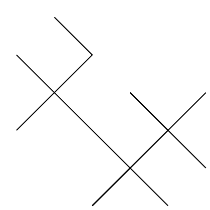
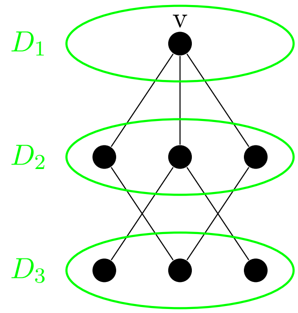


Figure 4: A tree

Theorem: G is bipartite $\Leftrightarrow$ G contains no odd cycles.

Proof: WLOG[1], assume G is connected, since G is bipartite $\Rightarrow$ each of its components are.

($\Rightarrow$) Suppose G is bipartite, with bipartition (X, Y). Let $v_0e_1v_1e_2...e_nv_n$ be an odd cycle (n is odd). Assume $v_0 \in X$. We then show that for $0 \leq k < \frac{n}{2}$, $v_{2k} \in X$. Assume inductively that $V_{2k-2} \in X$, where $k \geq 1$. Then v_{2k-1} lies in Y, since e_{2k-1} has endpoints in both X and Y. But v_{2k-1} inplies $v_{2k} \in X$ for the same reason. In particular, $v_{n-1} \in X$, but this means the two endpoints of e_n, v_0 and v_{n-1}, both lie in X. This contradict the fact that G is bipartite.



($\Leftarrow$) Suppose G contains no odd cycles. Let $v \in V$, and for all $u \in V$, define $d(v)$ = length of the shortest path from u to v. Let $D_i = \{u \in V : d(u) = i\}$.

Claim 1: $j \geq i+2 \Rightarrow$ there are no edges with endpoints in D_i or D_j.

Claim 2: any $i \geq 0$, there are no edges with both endpoints in D_i.

Then, letting $X = \bigcup_{i \text{ even}} D_i$ and $Y = \bigcup_{i \text{ odd}} D_i$, then (X, Y) forms a bipartition of G.

Proof of claim 1: Suppose there were some vertices u, v, and integers i, j, such that $j \geq i + 2$, $u \in D_i$, $w \in D_j$, and $uw \in E$. Then, a shortest path from v to w is no longer than the path by adjoining uw to the shortest path from v to u. So, $d(w) \leq i + 1$. This contracting the fact that $w \in D_j$, that is, $d(w) \geq qi + 1$.

Proof of claim 2: Suppose there were some $i \geq 0$ and vertices $u, w \in D_i$ such that $uw \in D_i$. Then, $\exists$ two paths: $P_1 = (v = a_0, a_1, a_2, ..., a_{i-1}, u = a_i)$ and $P_2 = (v = b_0, b_1, b_2, ..., b_{i-1}, w = b_i)$. Let m be thte largest index such that $a_k \neq b_k \; \forall \; m + 1 \leq k \leq i$. Then, $a_m a_{m+1} ... a_{i-1} u w b_{i-1} ... b_{m+1} b_m$ is a cycle of length 2(i-m)+1, which is odd. $\Rightarrow\Leftarrow$.

■

2 Matching

2.1 Stable Marriages

We have n boys and n girls. Each boy has an ordered list of girls and vice versa.

A set M of marriages is *stable* if there is no boy-girl pair who prefer each other to their current pairings in M. We call this situation an unstable (unblocking) pair [Figure 5].

2.1.1 Example

In the following example [Figure 6], we have 3 boys and 3 girls, each with their own preference list, but the given matching isn't a stable marriage.

[1] Without loss of generality

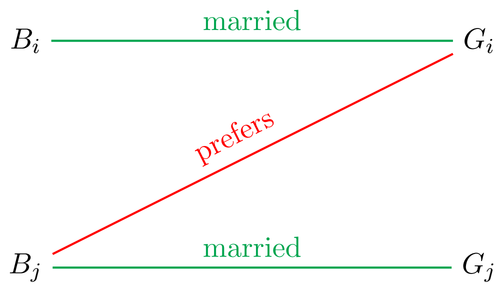


Figure 5: **Unstable pair** B_j prefers G_i to G_j and G_i prefers B_j to B_i

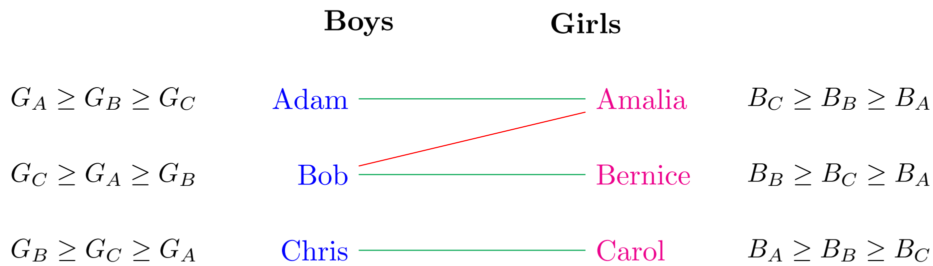


Figure 6: Unstable because Amalia and Bob prefer each other over their current partner

But when trying again, we can easily find two stable configuations [Figure 7]

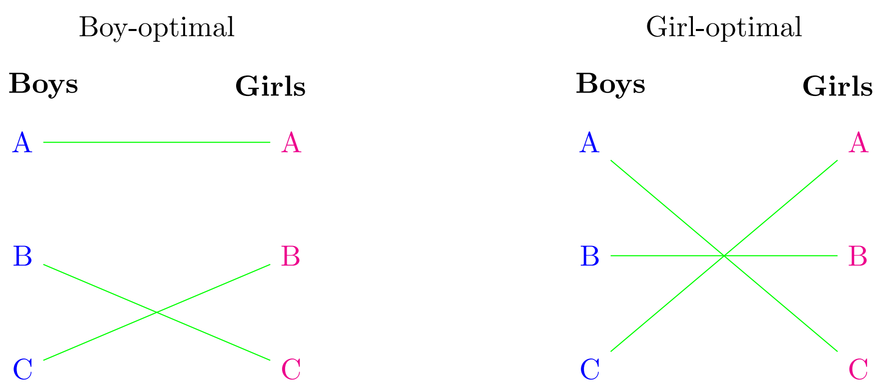


Figure 7: These work because each boy prefers a different girl, and each girl prefers a different boy.

Do stable matchings exist in general?

Theorem (Gale & Shapley): A stable matching always exists

Proof (by algorithm): While there is some "single" boy B, B proposes to the next girl on his list, call her G. Girl G accepts if she is single or prefers B to her current fiancé. Claim is that the algorithm terminates for any set of lists with a stable matching.

NOTE: as the algorithm proceeds, girls' choices only get better and mens' only get worse. Each time a girl changes fiancè, she trades up. A boy only changes if he gets dumped by G and he then proposes to the next girl on his list.

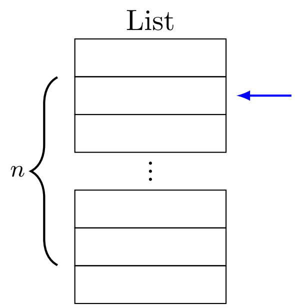


Figure 8: Preference list

Corollary: The algorithm terminates. Say boy B has his list. The pointer aims at his current match. There are n boys and n possible pointers into their lists [Figure 8]. each dumping moves the pointer down the list by one. We have $\leq n^2$ total dumpings. The algorithm terminates after $\mathcal{O}(n^2)$.

The matching returned by the algorithm is stable. Suppose M is the output matching, and has unstable pair (B_i, G_j), for a contradiction:

- B_i prefers G_j to current match G_i
- G_j prefers B_i to current match B_j

Since B_i prefers G_j to G_i, he proposed to her earlier and she either rejected him, or accepted and dumped him later. In either case, she was at some point matched to some B_k she preferred to B_i. By observation, her partners only improved from that point on. Thus, she prefers B_j to B_k and B_k to $B_i \Rightarrow$ prefers B_j to B_i and (B_i, G_j) is not unstable. $\Rightarrow\Leftarrow$ (contradiction)

There can be many stable matchings. Let:

$$\mathcal{S} = \{M_1, M_2, ..., M_k\}$$

be the set of all stable matchings. Call G_j a *valid partner* for B_i if (B_i, G_j) are matched in some $M_i \in \mathcal{S}$. For each B, let $G^+(B)$ be his most preferred valid partner.

Remarkably, the boy-proposal algorithm matches each boy B to $G^+(B)$. To show this, we require a lemma:

Lemma: a girl never rejects a valid partner

Proof (by contradiction): Suppose not. Consider the first time G_j rejects a valid partner B_i. Say (B_i, G_j) were matched in $M_t \in \mathcal{S}$. Say G_j dumps B_i for B_j at that time. Say (B_j, G_k) is a match in M_t [Figure 9].

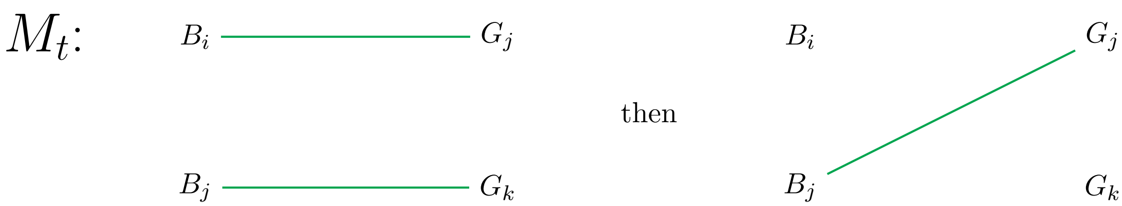


Figure 9: A valid partner being dumped by a girl in boy-proposal

Since B_i is the first valid partner to be dumped, we claim B_j prefers G_j to G_k. Why? Supposed B_j prefers G_k to G_j. Thus he proposes first to G_k. But $(G_k, B_j) \in M_t$, and therefore G_k is valid for B_j. But B_j was as we supposed in the beginning the first valid person to be dumped, which means B_j did not get dumped and $B - j$ is not free to propose to G_j. $\Rightarrow\Leftarrow$

So B_j prefers G_j to G_k and G_j prefers B_j to B_i, therefore (B_i, G_j) is unstable in M_t. But $M_t \in \mathcal{S}$ and in thus stable. $\Rightarrow\Leftarrow$. Hence a girl never rejects a valid partner.

$\blacksquare$

Now we will show that the boy-proposal algorithm matches each boy B with $G^+(B)$.

Proof: If B_i is matched by algorithm to G_j, who he doesn't like as much as $G^+(B_i)$, then he proposed to $G^+(B_i)$ first. But $G^+(B_i)$ and B_i are valid, hence $G^+(B_i)$couldn't have rejected him. $\Rightarrow\Leftarrow$

Let $B^-(G_j)$ be the worst partner for G_j amongst all stable matchings.

Lemma: The boy-proposal algorithm matches each G_j to $B^-(G_j)$.

Proof: Supposed B_j and G_j are matched, whom she prefers to $B^-(G_j)$. Say $(G_j, B^-(G_j)) \in M_r$ and $(G_i, B_j) \in M_r$ [Figure 10].

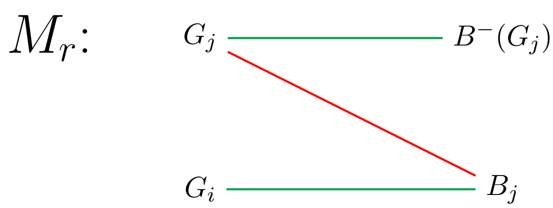


Figure 10: By the previous, B_j gets $G^+(B_j)$, so $G_j = G^+(B_j)$

Thus, B_j prefers G_j to G_i and G_j prefers B_j to $B^-(G_j)$, therefore the valid pair (B_i, G_j) is unstable in M_r. $\Rightarrow\Leftarrow$. It follows that G_j gets $B^-(G_j)$ with boy proposal.

$\blacksquare$

2.2 Matching

A *matching* in a graph $G(V, E)$ is a set $M \subseteq E$ of vertex-disjoint edges, i.e., each vertex of G is the endpoint of at most one edge in M.

we say $v \in V$ is *matched* (or *saturated*) by M if it is the endpoint of some edges in M. Otherwise, it is *unmatched*. A path P is *M-alternating* if its edges are alternatively in M and not in M.

An alternating path is *M-augmenting* if its endpoints are unmatched.

Theorem: A matching in G is of maximum cardinality $\iff$ there is no M-augmenting path.

Proof:($\Rightarrow$) Suppose P is an M-augmenting path, then switching the edges in P produces a larger matching. Let $M' = M \oplus E(P)$ (Symmetric difference of M and the edges in the path P).

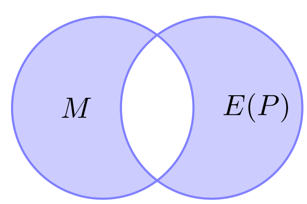
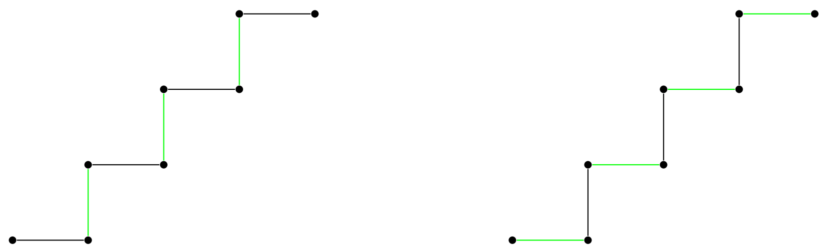


Figure 11: Symmetric difference of M and $E(P)$

$$
\begin{aligned}
M \oplus E(P) &= (M \cup E(P)) - (M \cap E(P)) \\
&= (M - E(P)) \cup (E(P) - M)
\end{aligned}
$$



(a) M has matching of size 3 (b) M' has matching of size 4

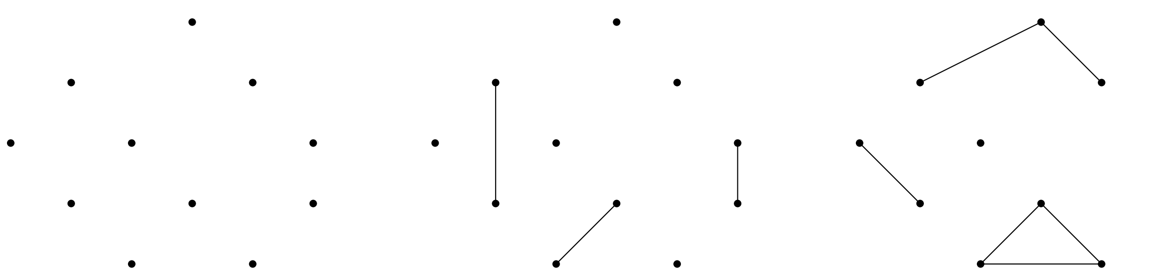
($\Leftarrow$) Suppose M has no augmenting path. Claim that it is a maximum matching. Suppose not, and that M^* is a maximum matching where $|M^*| > |M|$. Consider $M \oplus M^*$. Let H be the subgraph induced by the edges.

Claim:

$$
\begin{aligned}
|M| &= \# \text{ of } M \text{ -edges } \in H + |M \cap M^*| \\
&= \# \text{ of } M^*\text{-edges} \in H + |M \cap M^*|
\end{aligned}
$$

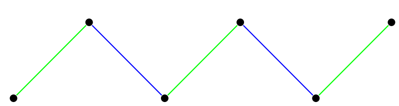
What is the degree of any vertex in H? It's at most two, since each vertex is incident to at most one edge in M and at most one edge in M^*. $deg_H(v) \in \{0, 1, 2\}$.

What does H look like?

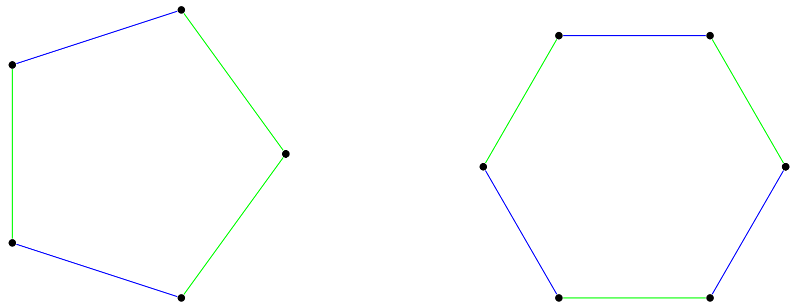


(a) $d_H(v) = 0 \ \forall \ v$ (b) $d_H(v) \in \{0, 1\} \ \forall \ v$ (c) $d_H(v) \in \{0, 1\} \ \forall \ v$

Say blue is M and green is M^*. They alternate:



This means that a cycle must be even:



(a) No (b) Yes

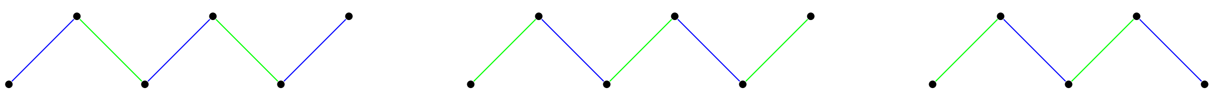
Each component is either

- an even cycle
- a path

Since alternating an even cycle doesn't change the size of M nor M^*, we will focus on paths.

Consider the 3 following types of paths:

1. M^*-augmenting
2. M-augmenting
3. augments nothing



(a) Type 1 (b) Type 2 (c) Type 3

There are no type 1 paths since they are M^*-augmenting and we assumed M^* was maximum! (See $\Rightarrow$ path of the proof). Each type 3 path, similarly to the cycle components, have the same number M and M^* edges. But, by the claim, H must have more M^* edges than M-edges. Therefore there is a type 2 component, and thus is an M-augmenting path. $\Rightarrow\Leftarrow$

$\blacksquare$

NOTE: This theorem holds for all graphs.

2.3 Matching in Bipartite Graph

G is bipartite if there is a partition $V(G) = X \cup Y$, such that each edge has one endpoint in Z and the other in Y (figure 12).

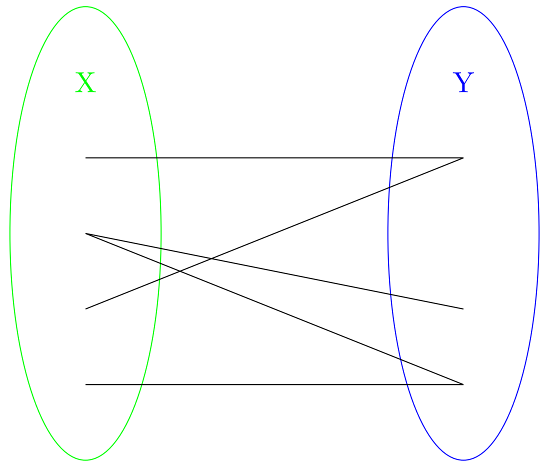
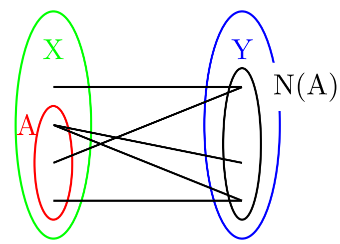


Figure 12: Bipartite graph partitioned into vertex set X and Y

Definition: A matching is perfect if it matches each vertex of G (we can only have degree 1 matching here).



Fundamental question: "When does a graph have a perfect matching?"

Definition: For $A \in V$, denote by $N(A)$ the set of neighbors of A, i.e., $N(A) = \{v \notin A : \exists\, uv \in E, u \in A\}$

Theorem (Hall's): A bipartite graph G with $|X| = |Y|$ has a perfect matching $\iff$ $|N(A) \geq |A| \;\forall\; A \subseteq X$. (Known as Hall's condition)

Proof: ($\Rightarrow$) Trivially holds since we can't have this:

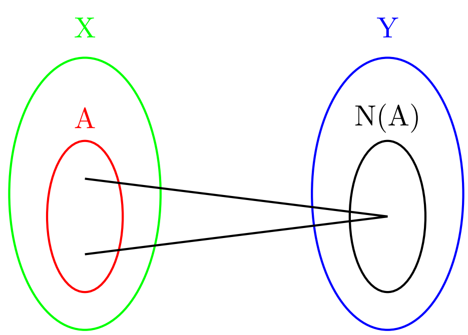


Figure 13: The two vertices in A have only one possible vertex they can match with, therefore there is no perfect matching that would match both.

($\Leftarrow$) If we have some matching M with unmatched vertex u, then we showed how to find an M-augmenting path from u. This gives a new matching which is larger. Repeat until you get a perfect matching.

Algorithm to find M-augmenting path from u:

Let $A = \{u\}, B = \emptyset$. Maintain two properties of A and B as we proceed.

i) $A \subset X$, $B \subset Y$. $A - u$ matched to B by M. (N.B. $|A| = |B| + 1$)

ii) There is an M-alternating path from u to any vertex in $A - u$ in the graph $G[A \cup B]$.

Repeat:

- Choose $v \in N(A) - B$. Let $e = wv, w \in A$. Combine an alternating path from u to w (by ii), with edge e, then get an M-augmenting path and quit.
- if v is matched to some $u' \in X - A$, and u' not in A by i), get $A \in A \cup \{u'\}$, $B \in B \cup \{v\}$. Clearly i) holds. Check that ii) holds (similar to previous argument).
- We can always find another vertex because Hall's condition implies that $N(A) \geq |A| = |B| + 1 > |B|$.

  This proof gives an algorithm for finding a perfect matching in G if it satisfies the Hall Condition.

  NOTE: Runtime $\mathcal{O}(VE)$ steps. There exists faster algorithms.

2.4 Applications

A graph is *d-regular* if every vertex has degree d.

Theorem: Any d-regular bipartite graph can be decomposed into d perfect matchings, i.e., the edges $E = M_1 \cup M_2 \cup ... \cup M_d$ where each M_i is a perfect matching.

Proof: It is enough to show that we have one perfect matching in G, since if M is a perfect matching, then $G - M$ is a $(d-1)$-regular, and we can repeat.

First, note that since each edge has one end in X and one in Y:

$$\sum_{x \in X} deg(x) = |E| = \sum_{y \in Y} deg(y) \Rightarrow |X| = |Y|$$

We also have that $|E| = d|X| = d|Y|$ since $deg(x) = d \; \forall x \; inX$. Consider $A \subset X$:

$$\begin{aligned} d|A| &= \sum_{x \in A} deg(x) \\ &= \#\text{ of edges with 1 end in } A \\ &\leq \#\text{ of edges with one end in } N(A) \\ &= \sum_{y \in N(A)} deg(y) \\ &= d|N(A)| \end{aligned}$$

Thus $|A| \leq |N(A)|$. So G satisfies Hall's Condition and hence has a perfect matching.

■

2.4.1 Latin Squares

An $r \times n$ grid is a *Latin Rectangle* if the numbers in each row and column are distinct.

Theorem: Every $r \times n$ Latin rectangle with $r \leq n$ can be extended to an $n \times n$ Latin square.

Example

1	2	3	4
2	3	4	1
4	1	2	3

Proof Define a bipartite graph with a vertex for each column (X), and a vertex for each number $1, 2, ..., n$ (Y). Add an edge from column j to number i if i does not appear in column j.

Each vertex in X will be connected with $n - r$ vertices on the other side. Hence, G is $(n-r)$-regular and has $n - r$ perfect matchings. These give $n - r$ rows which we can add to make the Latin square.

■

2.4.2 Systems of Distinct Representatives

Let $Y = \{y_1, y_2, ..., y_m\}$ and $S_1, S_2, ..., S_n \subseteq Y$. $D = \{y_1, y_2, ..., y_k\} \subseteq Y$ is a *system of representatives* (SDR) if $y_i \in S_i\ \forall i$.

Theorem: An SDR exists for a set family $\mathcal{S} = \{s_1, s_2, ...s_n\} \Leftrightarrow$ for any k sets from $\mathcal{S}$, their union contains at least k elements.

Proof: ($\Rightarrow$) Since if $S_1, ..., S_k$ are k sets and they have representatives $y_1, ..., y_k$, then:

$$\bigcup_{i=1}^{k} S_i \supseteq y_1, ..., y_k \Rightarrow |\bigcup_{i=1}^{k} S_i| \geq k$$

($\Leftarrow$) Set up a bipartite graph $G = (X \cup Y, E)$, where each $x_i \in X$ reps S_i. Put edges $x_i y_j$ if $y_j \in S_i$. Then an SDR corresponds precisely to a perfect matching, and Hall's Condition is just the Condition that for any k sets in $\mathcal{S}$, their union contains at least k elements.

■

2.4.3 Maximum Bipartite Matching

Given a $0-1$ $m \times n$ matrix M, its *term rank*, denoted $\tau(M)$, is the largest number of 1's that can be chosen such that no two lie on the same line (row or column). Call such a set of entries a "*packing of 1's in M*".

Note that the four circled lines contain all the ones, therefore $\tau(M) \leq 4$, since we can choose at most one 1 from each line.

Example

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}$$

The *cover number* of M, denoted $\gamma(M)$, is the minimum number of line whose deletion results in a 0-matrix. That is, these lines "cover" all the 1's.

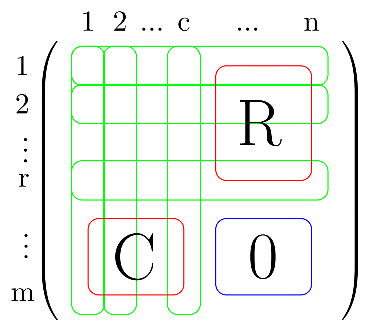
For any M, $\tau(M) \leq \gamma(M)$. Since, again, we can have at most one 1 on each of the $\gamma(M)$ lines of the cover.

Theorem: For any $0-1$ matrix M, $\tau(M) = \gamma(M)$

Proof: We have already shown that $\tau(M) \leq \gamma(M)$. With now need to show that $\tau(M) \geq \gamma(M)$. Suppose $\gamma(M) = r + c$, where r and c correspond to the number of rows and columns in our cover respectively. WLOG, we can assume that the cover used rows $1, 2, ..., r$ and columns $1, 2, ..., c$, since we can swap rows and columns. In the previous example, this would look like Figure 14(a), and in a general case, it would look like Figure 14(b)

$$\begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}$$

(a) Previous example



(b) General case

Figure 14: Swapping rows and columns so that we get a 0-submatrix in the bottom right

Note: no element of R is in a column line with an element of C.

Idea: combine a packing of 1's in R with a packing in C.

Claim: We can pack r 1's in R and c 1's in C, therefore we get a packing of size $r + c$. We'll show this for R, and the idea for C will be similar.

Create a bipartite graph G where X corresponds to rows $1, 2, ..., r$ and Y corresponds to columns $c + 1, c + 2, ..., n$. Put an edge between row-i-vertex and col-j-vertex if $M_{ij} = 1$. Packing r 1's in R corresponds to picking one 1 in each row of R, so we don't use the same column twice. This is equivalent to choosing a matching that matches every vertex in X (row vertices).

Due to Hall's Theorem, we know that there is such a matching as long as Hall's condition holds. Suppose Hall's condition fails for some $A \subset X = [r] = \{1, 2, ..., r\}$ (i.e., $|N(A)| < |A|$ where $N(A)$ is the set of columns $j \in \{c+1, ..., n\}$ such that $M_{ij} = 1$ for some $i \in A$). Hence, we get another line cover from $\{\text{col } 1, ..., c\} \cup \{row1, ..., r - A\} \cup \{\text{cols from } N(A)\}$, and this is smaller if $|N(A) < |A|$. $\Rightarrow\Leftarrow$

$\blacksquare$

2.4.4 Market Clearing Prices

(This section is very dodgy, needs to be reworked)

Consider n sellers, each with one house to sell and n buyers, each wanting a single house. Suppose buyer i values seller j's house at $V_ij \geq 0$. One approach is to match buyers to sellers to maximize total valuations. In graph theory, find a perfect matching M which maximizes $v(M) = \sum\limits_{i,j \in M} V_{ij}$.

Note that here, we ignore the sellers. How low of a price is the seller willing to accept?

Suppose seller j asks for price p_j for the house. What will buyers do in response to the price vector $(p_1, p_2, ..., p_n)$? Each buyer i views a payoff for each house j of $V_{ij} - p_j$. Call seller j_0 preffered for buyer i, if this house maximizes their payoff (i.e., $j_0 = \underset{j}{\text{argmax}}(V_{ij} - p_j)$)

Can we assign houses to buyers such that everyone buys from a preferred seller? Sometimes yes, sometimes no. Yes precisely when the preferred graph has a perfect matching.

Define $G_p = (X \cup Y, E)$ as the a preferred graph where $i, j \in E$ if j is preferred by seller of i. $X =$ buyers i and $Y =$ sellers. A vector of prices P is called *market-clearing* if G_p has a perfect matching.

Theorem: There exists market clearing prices

Proof (by algorithm):

while G_p *does not have a perfect matching* M **do**
  let $A \subseteq X$ such that $|N(A)| < |A|$;
  foreach $j \in N(A)$ **do**
    $P_j \leftarrow P_j + 1$;
  end
  if $P_{min} > 0$ **then**
    subtract P_{min} from all prices (to keep P_{min} at 0);
  end
end

The algorithm terminates: Define a potential function associated with each state of the algorithm. For each $i \in$ Buyers or Sellers:

$$\Phi(i) = \begin{cases} p_i & \text{if } i \in \text{ Sellers} \\ \max\limits_{j \in \text{ Sellers}} (V_{ij} - p_j) & \text{if } i \in \text{ Buyers} \end{cases}$$

Note:

(i) $\Phi(i) \geq 0$

(ii) Let $\Phi(P) = \sum\limits_{i \in \text{Buyers} \cup \text{Sellers}} \Phi(i)$, initially, since $P = (0, 0, ..., 0)$, $\Phi(P) = \sum\limits_{i \in \text{Buyers}} V_{i\,\max} < \infty$

Claim: On each iteration, $\Phi(P)$ decreases, and hence the algorithm terminates. This is true because in step (2), we subtract P_{min} from all sellers, so each of their potential decreases by P_{min}. Thus we have a total decrease of nP_{min}. By the same argument, all payoffs increased for buyers. Hence an increase of nP_{min}. This gives a net effect of 0.

In step (1), we increase the potential of each seller in $N(A)$ by 1, for a total increase of $|N(A)|$. But each buyer in A just had its maximum payoff decreased by 1, which gives a net decrease of $|A|$. Therefore, the new $\Phi(p)$ decreases by $|A| - |N(A)| > 0$.

■

3 Vertex Cover

For a graph $G = (V, E)$, a subset $C \subseteq V$ is a *vertex cover* if every edge has at least one endpoint in C.

NOTE: $V - C$ is a set of mutually non-adjacent vertices, called *independent set* or *stable set*.

OBSERVATION: For any matching M and vertex cover C, $|M| \leq |C|$, since every vertex can only be used once, and every edge in our matching will use one of the vertices of C.

Konig's Theorem: In a bipartite graph, the size of a maximum matching = the size of a minimum vertex cover.

Proof: We show this by reduction to the term rank problem. If $|X| = m$ and $|Y| = n$, then define an $m \times n$ $0-1$ matrix A where $A_{ij} = 1 \Leftrightarrow X_i Y_j \in E$. So rows of A correspond to vertices in X, columns of A correspond to vertices in Y, and 1's in A correspond to edges in G.

First, we can show that a maximum matching in G corresponds to picking the maximum number of 1's in A (edges in G) such that no two lie on the same row or column (not using the same vertex from X or Y more than once), and this value is $\tau(A)$. On the other hand, we can show that a minimum vertex cover corresponds to choosing the least number of vertices from X (rows) and Y (columns) so as to cover all the 1's (edges), and this corresponds to $\gamma(A)$. Lastly, from the previous theorem, we know that $\tau(A) = \gamma(A)$, therefore $|M| = |C|$.

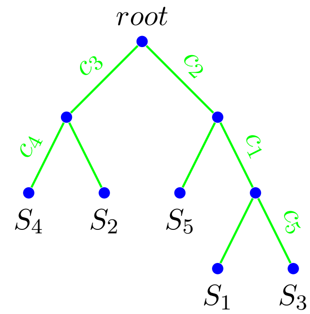
■

3.1 Perfect phylogenetic trees (PPT)

Given m species, each exhibiting some characteristics from a set of n characteristics, encode this information as an $m \times n$ $0-1$ matrix M, where $M_{ij} = 1$ if species i has characteristic j, and 0 otherwise.

Species	Chars.				
S_1	1	1	0	0	0
S_2	0	0	1	0	0
S_3	1	1	0	0	1
S_4	0	0	1	1	0
S_5	0	1	0	0	0

(a) Species-Chars Matrix



(b) PPT

A PPT is a rooted tree with the following properties:

i) Exactly m leaves, one for each species S_i

ii) Each characteristic labels one edge of the tree (some edges can be blank)

iii) For each species, the path from the root to S_i contains exactly the labels for S_i's characteristics

Let C_i be the set of species with characteristic i. The family $\mathcal{F} = \{C_1, ..., C_n\}$ is nested if $\forall i, j$:

a) S_i and S_j are disjoint
OR
b) They are comparable (i.e., $S_i \subseteq S_j$ or $S_j \subseteq S_i$)

Theorem: There exists a PPT $\Leftrightarrow$ $\mathcal{F}$ is nested

Proof: Beyond the scope of this course.

What if $\mathcal{F}$ is not nested? Try to find a large subset X of characteristics such that their corresponding family is nested. That is, restrict to the X-columns of M, then get a PPT.

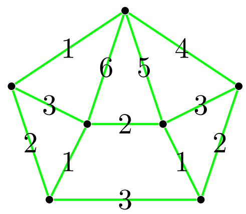
How to find X? Create a graph with a vertex for each C_i (i.e., each characteristic). Put an edge $ij \in E(G)$ if C_i and C_j are not nested.

Finding the set X corresponds to finding a largest stable set in G. This is equivalent to finding a minimum vertex cover (X is stable $\Leftrightarrow$ V-X is a cover).

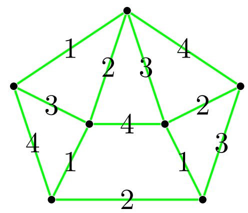
4 Graph coloring

4.1 Edge coloring

An *edge coloring* of a graph G is a function $c : E \to \mathbb{N}$ such that $c(e_1) \neq c(e_2)$ for any pair of edges e_1, e_2 with a common end point. The *edge-chromatic number* $\chi^E(G)$ is the minimum number of colors needed to edge-color G.



(a) 6 colors (bad)



(b) 4 colors (better)

OBSERVATION: $\chi^E(G) \geq \max_{v \in V} d(v) = \Delta$, since edges touching vertex v receive distinct colors.

NOTE: Any color class (edges with some color i) forms a matching.

Theorem: For any graph G, $\chi^E(G) \leq 2\Delta - 1$

Proof (by greedy algorithm): While there is an uncolored edge $e = (u, v)$, let C_u be the colors used already at u. Define C_v similarly. We know that $|C_u| \leq \Delta - 1$ and $|C_v| \leq \Delta - 1$. Hence, $\{1, 2, ..., 2\Delta - 1\} - (C_u \cup C_v)$ is not empty since $|C_u \cup C_v| \leq 2\Delta - 2$. We simply pick a color t in this set and give e that color. End while.

$\blacksquare$

Can we do better than $\chi^E(G) \leq 2\Delta - 1$?

Theorem (Vizing): For (simple) graph G, $\chi^E(G) \leq \Delta + 1$

Proof: Beyond this course

Theorem (König): For a bipartite graph G, $\chi^E(G) = \Delta$

Proof (by induction on $|E|$): We already have that $\chi^E(G) \geq \Delta$. We then show that $\chi^E(G) \leq \Delta$.

If $|E| = 0$, it's trivial to show.

Suppose $e = (u, v) \in E$ and consider $G' = G - e$. By induction, G' has an edge coloring C using at most Δ colors. Since $deg'_G(u) \leq \Delta - 1$, there is some color α not used by C at u. Similarly, there is a missing color β at v. If $\alpha = \beta$, then set $c(e) = \alpha$ to get a coloring of G. Else, if $\alpha \neq \beta$, WLOG, $\alpha = 1$, $\beta = 2$ (Figure 15).

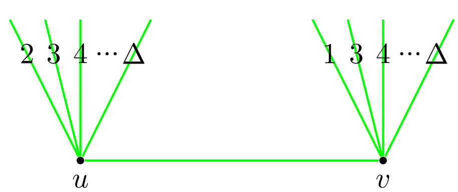
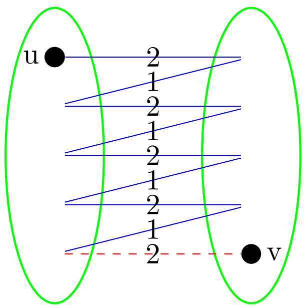


Figure 15: $\alpha = 1$ not used at u and $\beta = 2$ not used at v. $\Delta - 1$ colors used at each X and Y

Consider the subgraph H induced by the edges of color 1 and 2. Call M_i the color class for color i. By observing that in general, any color class forms a matching, M_1 and M_2 must be matchings, therefore, each component of H must be a path or a cycle (see Section 2.2).

Claim: The component containing u is a path and doesn't contain v.



This is because u is not incident to an edge of color 1, so $deg_H(u) = 1$. The path only has colors 1 and 2, alternating, starting at u with the color 2. If v is in the same component,, then since the edges in the path are alternating between 1 and 2, it would mean that v is touching an edge of color 2, but that can't be.

Then, all we need to do is swap the colors 1 and 2 on this path containing u, which gives us a new valid edge coloring for G, and allows us to now color e with the color 2.

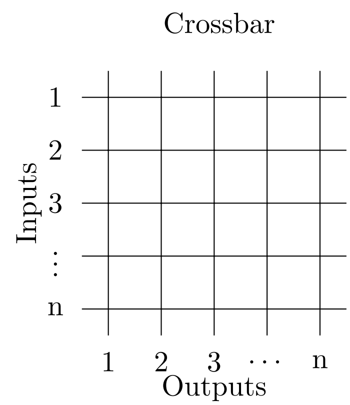
■

4.1.1 Applications

Sports Scheduling: Each pair of n teams should play each other on some Sunday. How many Sundays does it take? ANSWER: $\chi^E(K_n)$ where K_n is a complete graph on n vertices.

Job scheduling: We have m jobs $J_1, ..., J_m$ and n processors. Job J_i must be processed by every processor in set S_i (order doesn't matter). How to process the jobs in minimum time? ANSWER: bipartite edge coloring. Partitioning G into a minimum number of matchings $= \chi^E(G)$.

Switches: network design involves a graph G and making decisions about how "large" each edge and vertex should be. We have to realize that what looks like a node is often another network when you zoom in on it. We must choose how much to simplify. The basic building blocks are interconnected switches. GOAL: any pattern of requests from inputs to outputs should be "connectible" (i.e.: any matching from inputs to outputs).



Using a crossbar, we can achieve any communication, but the crosspoints are expensive, and there are $\mathcal{O}(n^2)$ of them. How to do better?

4.1.2 3-stage Clos network

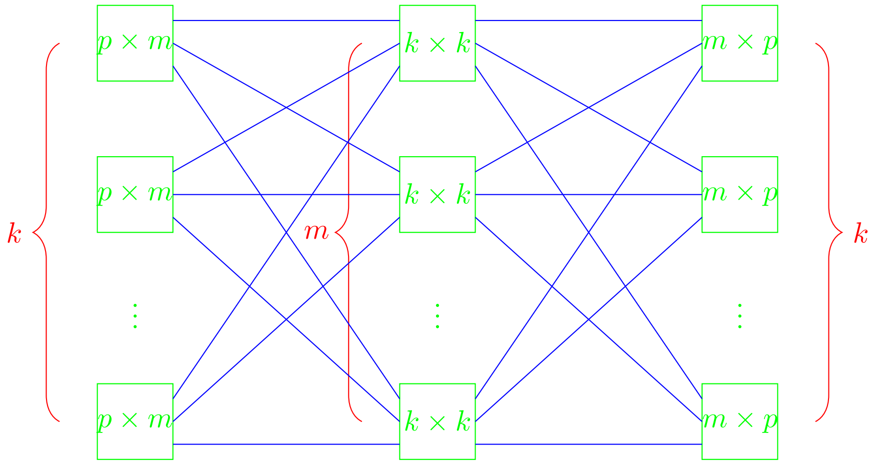


Figure 16: A 3-stage clos network. Each square is a crossbar.

Claim: For $p = m$, we can route any matching of size $n = pk$ from inputs to outputs.

Proof: Create a $k \times k$ bipartite graph G. X corresponds to the k input boxes and Y to the k output boxes. For each "demand" from a matching, we put an edge in G. If the demand goes from input box i to output box j, we put an edge from i to j. We allow multiple edges.

Note that G is $(p = m)$-regular, and that the maximum degree is p, since we are looking at the situation where we have a perfect matching from inputs to outputs. We can therefore find a p-edge coloring by König theorem (i.e., the edges of G partition into perfect matchings, one for each color class). $E(G) = M_1 \cup M_2 \cup ... \cup M_p$. we can now route all the demands in M_i via the middle stage box i.

Observation: if $k = m = p = \sqrt{n}$, then we have $3k = 3\sqrt{n}$ boxes, each with $\sqrt{n}\sqrt{n} = n$ crosspoints. This gives us a total of $\mathcal{O}(n^{\frac{3}{2}}) \ll \mathcal{O}(n^2)$.

4.1.3 Beneš network

Taking one step further, support $n = 2^i$ for some i. Take a clos network with $p = m = 2$ and $k = \frac{n}{2}$. Now, recurse on the two $\frac{n}{2} \times \frac{n}{2}$ boxes. The result is called a Beneš network and its number of crosspoints can be obtained via this recurrence: $f(n) = nf(2) + 2f\left(\frac{n}{2}\right)$. The number of crosspoints turns out to be $\mathcal{O}(n \log n)$, almost linear.