Lecture 11: Matchings

Definitions. Hall's Theorem and SDRs. Perfect matchings

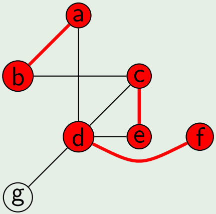
Outline of this lecture

- Matchings
  - perfect, maximum, and maximal matching
  - M-alternating patg, M-augmenting path
  - Berge's Teorem. Hall's Theorem.
- Systems of distinct representatives (SDRs)
- Weighted bipartite matchings
  - The Hungarian algorithm
- Spanning trees
  - Kruskal's algorithm
- Enumerating all trees with n nodes
  - Prüfer sequences

Definitions (1)

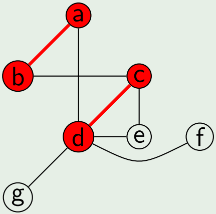
- **Assumption.** $G = (V, E)$ is a simple unoriented graph.
- A matching in G is a set M of edges in which no pair shares a vertex. The vertices belonging to the edges of M are said to be saturated by M (or M-saturated). The other vertices are M-unsaturated.

Example



$M_1 = \{(a,b),(c,e),(d,f)\}$

M_1-saturated nodes: a,b,c,e,d,f



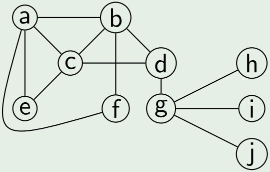
$M_2 = \{(a,b),(c,d)\}$

M_2-saturated nodes: a,b,c,d

Definitions (2)

- A perfect matching is a matching that saturates all nodes of G.
- A maximum matching of G is a matching that has the largest possible number of edges.
- A maximal matching of G is a matching that can not be enlarged by the addition of any edge.

Example (Maximum and maximal matchings)

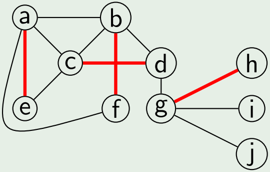


Maximum matchings?

Definitions (2)

- A perfect matching is a matching that saturates all nodes of G.
- A maximum matching of G is a matching that has the largest possible number of edges.
- A maximal matching of G is a matching that can not be enlarged by the addition of any edge.

Example (Maximum and maximal matchings)



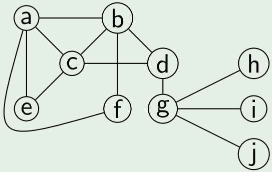
Maximum matchings?

$M_1 = \{(a,e),(b,f),(c,d),(g,h)\}$

Definitions (2)

- A perfect matching is a matching that saturates all nodes of G.
- A maximum matching of G is a matching that has the largest possible number of edges.
- A maximal matching of G is a matching that can not be enlarged by the addition of any edge.

Example (Maximum and maximal matchings)

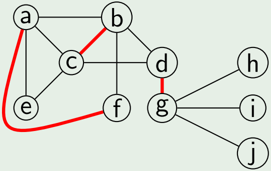


Maximal matchings?

Definitions (2)

- A perfect matching is a matching that saturates all nodes of G.
- A maximum matching of G is a matching that has the largest possible number of edges.
- A maximal matching of G is a matching that can not be enlarged by the addition of any edge.

Example (Maximum and maximal matchings)



Maximal matchings?

$M_2 = \{(d,g),(a,f),(b,c)\}$

Definitions (3)

Definition (M-alternating path, M-augmenting path)

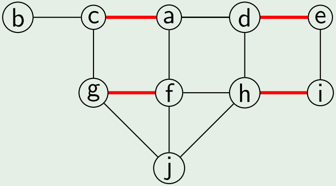
Given a graph G and a matching M, an M-alternating path is a path in G where the edges alternate between M-edges and non-M-edges. An M-augmenting path is an M-alternating path where both end vertices are M-unsaturated.

Definitions (3)

Definition (M-alternating path, M-augmenting path)

Given a graph G and a matching M, an M-alternating path is a path in G where the edges alternate between M-edges and non-M-edges. An M-augmenting path is an M-alternating path where both end vertices are M-unsaturated.

Example



M-alternating path: (c,a,d,e,i)

M-augmenting path: (j,g,f,a,c,b)

Berge's Theorem

Theorem

A matching M in a graph G is maximum if and only if G contains no M-augmenting paths.

PROOF OF "$\Rightarrow$". Suppose M is a maximum matching. We prove by contradiction that G has no M-augmenting paths. If $P : (v_1, v_2, \ldots, v_k)$ were an M-augmenting path then, by definition, k must be even and such that $(v_2, v_3), (v_4, v_5), \ldots, (v_{k-2}, v_{k-1})$ are edges in M, and $(v_1, v_2), (v_3, v_4), \ldots, (v_{k-1}, v_k)$ are not edges in M.

If so, we can define the matching M_1 of G to be

$$M_1 = (M \setminus \{(v_2, v_3), \ldots, (v_{k-2}, v_{k-1})\}) \cup \{(v_1, v_2), \ldots, (v_{k-1}, v_k)\}.$$

But M_1 contains one more edge than M, which contradicts the assumption that M is maximum.

Berge's Theorem
Proof of "$\Leftarrow$"

Assume G has no M-augmenting paths. Be prove by contradiction that M must be maximum. If M is not maximum then let M' be a matching of G larger than M, that is, $|M'| > |M|$. Let H be the subgraph of G defined as follows: $V(H) = V(G)$ and $E(H) =$the set of edges that appear exactly in one of M and M'. Since $|M'| > |M|$, H must have more edges in M' than in M.

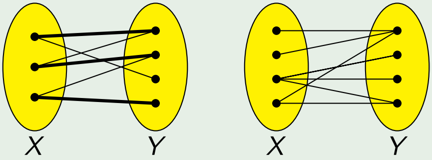
Every node of G lies on at most one edge from M and at most one edge from M' $\Rightarrow \deg_H(v) \leq 2$ for all $v \in V(H)$. This implies that every connected component of H is either a single node, a path, or a cycle. If it is a cycle, then it must be an even cycle, because the edges alternate between M-edges and M'-edges

$\Rightarrow$ the only connected components of H that may contain more M'-edges than M-edges are the paths. Since $|M'| > |M|$, there must be a path P in H that begins and ends with edges from M'. But P is an M-augmenting path, contradicting our assumption.

"Matching into"

- If G is a bipartite graph with partite sets X and Y, we say that X can be matched into Y if there exists a matching in G that saturates the nodes in X.

Example



(a) A bipartite graph where X can be matched into Y.

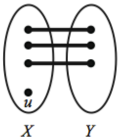
(b) A bipartite graph where X can not be matched into Y. *Why is this so?*

Hall's Theorem (a.k.a. Hall's Marriage Theorem)

Theorem

Let G be a bipartite graph with partite sets X and Y. X can be matched into Y if and only if $|N(S)| \geq |S|$ for all subsets S of X.

PROOF. Suppose X can be matched into Y and let $S \subseteq X$. Since S itself is also matched into Y, we learn that $|N(S)| \geq |S|$.
Now, suppose $|N(S)| \geq |S|$ for all $S \subseteq X$, and let M be a maximum matching. Suppose that $u \in X$ is not saturated by M.



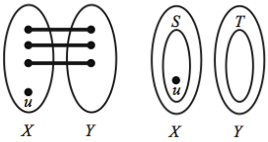
Let A be the set of nodes in G that can be joined to u by an M-alternating path, $S = A \cap X$, and $T = A \cap Y$.

Hall's Theorem (a.k.a. Hall's Marriage Theorem)

Theorem

Let G be a bipartite graph with partite sets X and Y. X can be matched into Y if and only if $|N(S)| \geq |S|$ for all subsets S of X.

PROOF. Suppose X can be matched into Y and let $S \subseteq X$. Since S itself is also matched into Y, we learn that $|N(S)| \geq |S|$.
Now, suppose $|N(S)| \geq |S|$ for all $S \subseteq X$, and let M be a maximum matching. Suppose that $u \in X$ is not saturated by M.



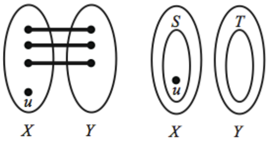
Let A be the set of nodes in G that can be joined to u by an M-alternating path, $S = A \cap X$, and $T = A \cap Y$.

Hall's Theorem (a.k.a. Hall's Marriage Theorem)

Theorem

Let G be a bipartite graph with partite sets X and Y. X can be matched into Y if and only if $|N(S)| \geq |S|$ for all subsets S of X.

PROOF. Suppose X can be matched into Y and let $S \subseteq X$. Since S itself is also matched into Y, we learn that $|N(S)| \geq |S|$.
Now, suppose $|N(S)| \geq |S|$ for all $S \subseteq X$, and let M be a maximum matching. Suppose that $u \in X$ is not saturated by M.



Let A be the set of nodes in G that can be joined to u by an M-alternating path, $S = A \cap X$, and $T = A \cap Y$.

By Berge's Theorem, all nodes in T are saturated by M, and u is the only unsaturated node of $S \Rightarrow |T| = |S| - 1$. It follows from Berge's Theorem and the definition of T that $N(S) = T$. But then we have that $|N(S)| = |S| - 1 < |S|$, and this is a contradiction.

System of distinct representatives (SDR)

Definition

Given a finite family of sets $X = \{S_1, \ldots, S_n\}$, a system of distinct representatives, or SDR, for the sets in X is a set of distinct elements $\{x_1, \ldots, x_n\}$ with $x_i \in S_i$ for $1 \leq i \leq n$.

Example

Let $S_1 = \{2, 8\}$, $S_2 = \{8\}$, $S_3 = \{5, 7\}$, $S_4 = \{2, 4, 8\}$, $S_5 = \{2, 4\}$.

- $X_1 = \{S_1, S_2, S_3, S_4\}$ does have SDR $\{2, 8, 7, 4\}$.
- $X_2 = \{S_1, S_2, S_4, S_5\}$ does not have an SDR.

Question. Under what conditions will a finite family of sets have an SDR?

SDRs of finite families of sets

Theorem

Let $S_1, S_2, \ldots, S_k$ be a collection of finite, nonempty sets. This collection has an SDR if and only if for every $t \in \{1, \ldots, k\}$, the union of any t of these sets contains at least t elements.

Proof. Let $S := S_1 \cup S_2 \cup \ldots \cup S_k$. Y is a finite set, say $Y = \{a_1, \ldots, a_n\}$, where $n = |S|$.
Let's consider the bipartite graph with partite sets $X = \{S_1, \ldots, S_k\}$ and Y. We place an edge between S_i and a_j if and only if $a_j \in S_i$.
Hall's Theorem implies that X can be matched into Y if and only if $|A| \leq |N(A)|$ for all subsets A of X. In other words, the collection of sets has an SDR if and only if for every $t \in \{1, \ldots, k\}$, the union of any t of these sets contains at least t elements.

Weighted bipartite matchings

A motivational problem

Three workers: John, Dan and Roy are supposed to perform three tasks: wash the bathroom, cook, and clean windows. Each of them asks a certain price to perform a task, for example:

	wash bathroom	cook	clean windows
John	20	30	30
Dan	30	20	40
Roy	30	30	20

We wish to assign a task to each of them, such that we spend minimum amount of money.

Weighted bipartite matchings

PROBLEM:

Assume $G = K_{n,n}$ is a complete bipartite graph between sets $S = \{x_1, \dots, x_n\}$, $T = \{y_1, \dots, y_n\}$, such that every $(x_i, y_j) \in E$ has a weight $w(i,j) \geq 0$.

Find a cost function $c : V \to \mathbb{R}$ which assigns to every node $v \in V$ a cost $c(v)$, such that

1. $c(x_i) + c(y_j) \geq w(i,j)$ for all $(x_i, y_j) \in E$
2. $\sum_{i=1}^{n} c(x_i) + \sum_{i=1}^{n} c(y_j)$ has smallest possible value.

A function $c : V \to \mathbb{R}$ satisfying condition 1 is called cover for the nodes of G. If it also satisfies condition 2, it is called minimal cover of the nodes of G.
The sum $\sum_{i=1}^{n} c(x_i) + \sum_{i=1}^{n} c(y_j)$ is called the cost of the cover c.

Weighted bipartite matchings

Well-known theoretical results

- For every perfect matching M between S and T, and cover c, we have

$$\sum_{i=1}^{n} c(x_i) + \sum_{i=1}^{n} c(y_j) \geq \sum_{(x_i,y_j)\in M} w(i,j).$$

Moreover:

$$\sum_{i=1}^{n} c(x_i) + \sum_{i=1}^{n} c(y_j) = \sum_{(x_i,y_j)\in M} w(i,j)$$

if and only if $c(x_i) + c(y_j) = w(i,j)$ for all edges $(x_i, y_j) \in M$. In this case, M is a maximal matching and c is a minimal cover of G.

Weighted bipartite matchings
The Hungarian Algorithm (1)

Computes a cover of a weighted graph $K_{n,n}$ in polynomial time $O(n^4)$:

- Initially, $c(x_i) = 0$ and $c(y_j) = \max\{w(i,j) \mid 1 \leq i \leq n\}$ for $1 \leq i \leq n$. (REMARK: c is a cover for all nodes of G.)
- The algorithm updates c in a finite number of steps, until c becomes a minimal cover. At every step, we take into account:
  1. the graph $G_c := \{(x_i, y_j) \mid c(x_i) + c(y_j) = w(i,j)\}$ and a matching M of G_c. Initially, $M = \emptyset$.
     - **The algorithm stops when M becomes a perfect matching** (it has n edges).
  2. $R_S \subseteq S$ are the M-unsaturated nodes from S, and $R_T \subseteq T$ are the M-unsaturated nodes from din T.
  3. Z :=the set of nodes reacheable by M-alternating paths starting from R_S:
     - We distinguish 2 cases: if $R_T \cap Z = \emptyset$ or $R_T \cap Z \neq \emptyset$

Weighted bipartite matchings

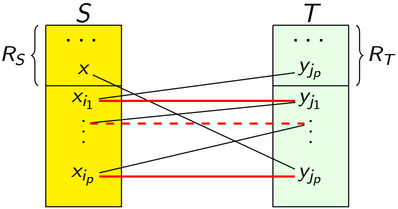
The Hungarian Algorithm (2)

If $R_T \cap Z \neq \emptyset$, let $(x_{i_1}, y_{j_1}, x_{i_2}, \ldots, x_{i_p}, y_{j_p})$ be an M-alternating path form $x_{i_1} \in R_S$ to $y_{j_p} \in R_T$. This means that

$(y_{j_k}, x_{i_{k+1}}) \in M$ for $1 \leq k < p$ and $(x_{i_k}, y_{j_k}) \notin M$ for $1 \leq k \leq p$. In this case, we modify M to be

$$M := (M - \{(y_{j_k}, x_{i_{k+1}}) \mid 1 \leq k < p\}) \cup \{(x_{i_k}, y_{j_k}) \mid 1 \leq k \leq p\}.$$

Graph G_c:



REMARK: $|M|$ GROWS BY 1

Weighted bipartite matchings

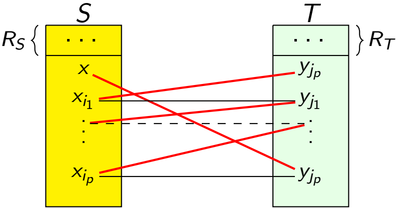
The Hungarian Algorithm (2)

If $R_T \cap Z \neq \emptyset$, let $(x_{i_1}, y_{j_1}, x_{i_2}, \ldots, x_{i_p}, y_{j_p})$ be an M-alternating path form $x_{i_1} \in R_S$ to $y_{j_p} \in R_T$. This means that

> $(y_{j_k}, x_{i_{k+1}}) \in M$ for $1 \leq k < p$ and $(x_{i_k}, y_{j_k}) \notin M$ for $1 \leq k \leq p$. In this case, we modify M to be

$$M := (M - \{(y_{j_k}, x_{i_{k+1}}) \mid 1 \leq k < p\}) \cup \{(x_{i_k}, y_{j_k}) \mid 1 \leq k \leq p\}.$$

Graph G_c:

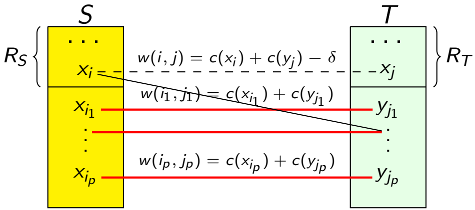


REMARK: $|M|$ GROWS BY 1

Weighted bipartite matchings
The Hungarian Algorithm (3)

If $R_T \cap Z = \emptyset$, let
$\delta = \min\{c(x_i) + c(y_j) - w(i,j) \mid x_i \in Z \cap S, y_j \in T \setminus Z\}$

Graph G_c:



Note that $\delta > 0$. We modify c as follows:

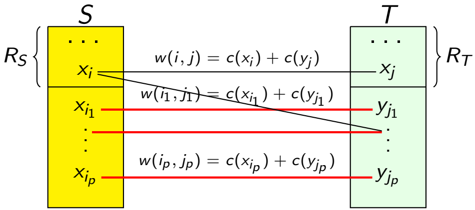
- $c(x_i) := c(x_i) - \delta$ for all $x_i \in Z \cap S$, and
- $c(y_j) = c(y_j) + \delta$ for all $y_j \in Z \cap T$

$\Rightarrow$ G_c gets modified: $|R_T \cap Z|$ grows, and $M \subseteq G_c$ continues to hold

Weighted bipartite matchings
The Hungarian Algorithm (3)

If $R_T \cap Z = \emptyset$, let
$\delta = \min\{c(x_i) + c(y_j) - w(i,j) \mid x_i \in Z \cap S, y_j \in T \setminus Z\}$

New graph G_c:
(after c gets modified)



Note that $\delta > 0$. We modify c as follows:

- $c(x_i) := c(x_i) - \delta$ for all $x_i \in Z \cap S$, and
- $c(y_j) = c(y_j) + \delta$ for all $y_j \in Z \cap T$

$\Rightarrow$ G_c gets modified: $|R_T \cap Z|$ grows, and $M \subseteq G_c$ continues to hold

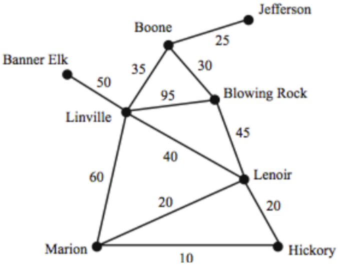
Remark

We can solve the motivational problem with the Hungarian algorithm as follows:

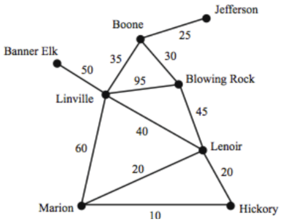
- We consider the weighted bipartite graph $K_{3,3}$ between sets $S = \{\text{John, Dan, Roy}\}$ and $T = \{\text{WB,C,CW}\}$ where WB: wash bathroom, C: cook, CW: clean windows and label every edge (x, y) from $x \in S$ to $y \in T$ with the cost requested by worker x to perform task y.
- A run of the Hungarian algorithm on this graph computes a perfect matching M, which indicates what task to assign to each worker.

Spanning trees
Motivating problem

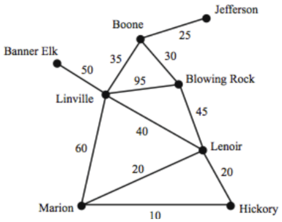
The North Carolina Transportation Department (NCDOT) decided to realize a fast railway network between 8 cities from west of the state. Some cities are already connected by roads, and the plan is to place railways beside the existing roads. Different terrain forms imply different costs to build the rail connections. NCDOT employed a consultant to compute the construction costs of a railway beside every existing road between 2 cities. The consultant produced the graph illustrated below, where the costs of building every rail connection are indicated. The objective is to build the railway network with minimum costs, and to ensure the connection between every two cities.



Spanning trees
Motivating problem

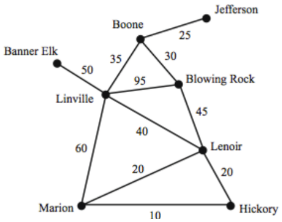


Spanning trees
Motivating problem



- A spanning tree of a graph G is a tree containing all nodes of G.

Spanning trees
Motivating problem



- A spanning tree of a graph G is a tree containing all nodes of G.
- We wish to a spanning tree T with minimum total cost, that is, a minimum spanning tree:
  - The sum of costs of the edges of $T \leq$ the sum of costs of the edges of any other spanning tree of G.

Spanning trees
Motivating problem (continued)

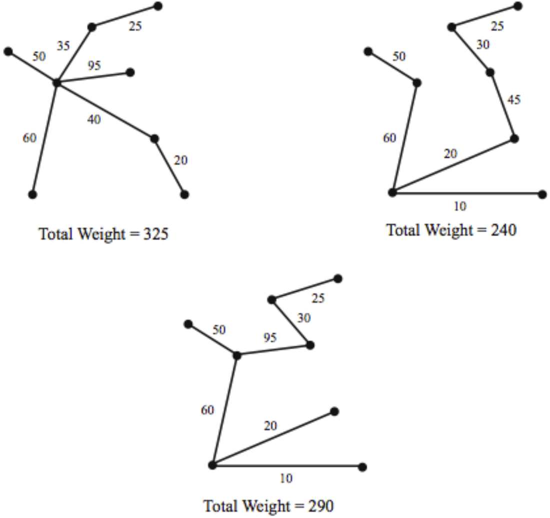


FIGURE 1.42. Several spanning trees.

Finding a minimum weight spanning tree
Kruskal's algorithm

Given a connected weighted graph G

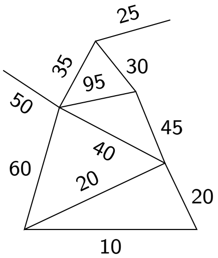
(1) Find an unmarked edge with minimum weight, and mark it.

(2) Take into account only the unmarked edges which do not produce a cycle with the other marked edges.
   - ▷ choose such an unmarked edge with minimum weight, and
   - ▷ mark it.

(3) Repeat step (2) until the marked edges form a spanning tree of G.

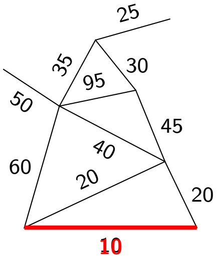
Kruskal's Algorithm

Illustration for the weighted graph from the motivating problem



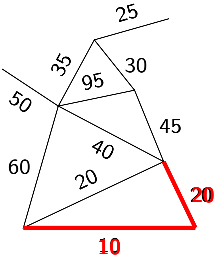
Kruskal's Algorithm

Illustration for the weighted graph from the motivating problem



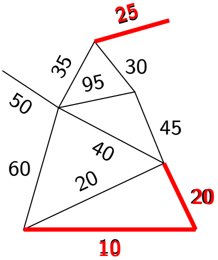
Kruskal's Algorithm

Illustration for the weighted graph from the motivating problem



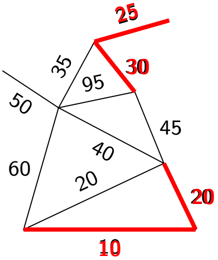
Kruskal's Algorithm

Illustration for the weighted graph from the motivating problem



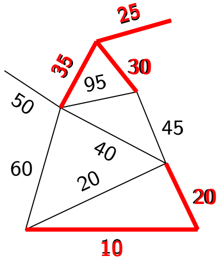
Kruskal's Algorithm

Illustration for the weighted graph from the motivating problem



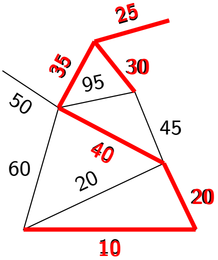
Kruskal's Algorithm

Illustration for the weighted graph from the motivating problem



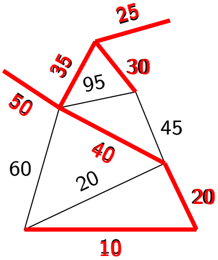
Kruskal's Algorithm

Illustration for the weighted graph from the motivating problem



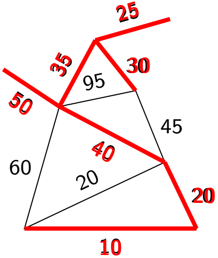
Kruskal's Algorithm

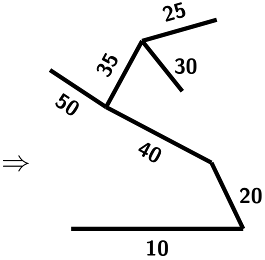
Illustration for the weighted graph from the motivating problem



Kruskal's Algorithm

Illustration for the weighted graph from the motivating problem





Total weight: 210

Enumeration of trees

We wish to enumerate all trees with n labeled nodes.
(Remark: a tree is a connected graph without cycles)

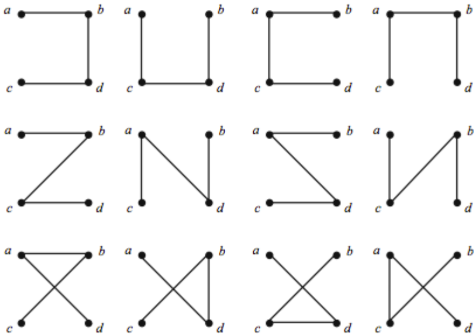
- Assume that the node positions are fixed; we wish to enumerate all the possibilities to draw a tree whose nodes are the given n nodes.
  For example, for $n = 4$ we have 16 different trees:

Enumeration of trees

We wish to enumerate all trees with n labeled nodes.
(Remark: a tree is a connected graph without cycles)

- Assume that the node positions are fixed; we wish to enumerate all the possibilities to draw a tree whose nodes are the given n nodes.

  For example, for $n = 4$ we have 16 different trees:



Enumeration of trees

Cayley's Theorem. Prüfer's enumeration method

Theorem (Cayley's Theorem)

There are n^{n-2} distinct trees with n labeled nodes.

- Prüfer found a way to enumerate all trees with nodes labeled by numbers from to, n. His method is based on the definition of a bijective correspondence between these trees and the set of sequences of numbers

$$\underbrace{v_1, v_2, \ldots, v_{n-2}}_{n-2 \text{ numbers}} \quad \text{where } 1 \leq v_i \leq n$$

REMARK: There are n^{n-2} such sequences.

Enumeration of trees

The computation of the Prüfer sequence for a tree T with nodes 1,2,. . . ,n

Given a tree T with nodes $1, \ldots, n$

(1) Initially, the sequence is empty. Let $i = 0$ and $T_0 = T$.

(2) Find the leaf of T_i with smallest label; assume the label is v.

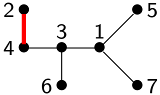
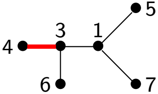
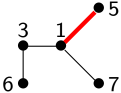
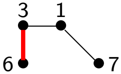
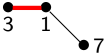
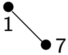
(3) Add the label of the parent of v to the Prüfer sequence.

(4) Remove leaf node v from $T_i \Rightarrow$ a smaller tree T_{i+1}.

(5) If T_{i+1} is K_2, stop. Otherwise, increment i by 1 and goto (2).

Prüfer's method

The computation of the Prüfer sequence of a tree

<table>
<tr><th>Current tree</th><th>current Prüfer sequence</th></tr>
<tr><td>$T = T_0$ </td><td></td></tr>
<tr><td>T_1 </td><td>4</td></tr>
<tr><td>T_2 </td><td>4,3</td></tr>
<tr><td>T_3 </td><td>4,3,1</td></tr>
<tr><td>T_4 </td><td>4,3,1,3</td></tr>
<tr><td>T_5 </td><td>4,3,1,3,1</td></tr>
</table>

Enumeration of trees

Computing the tree T corresponding to a given Prüfer sequence $a_1, \ldots, a_k$

Given a sequence $\sigma = a_1, a_2, \ldots, a_k$ of numbers from the set $\{1, \ldots, k+2\}$

(1) Draw $k+2$ nodes labeled with numbers $1, 2, \ldots, k+2$. Let $S = \{1, 2, \ldots, k+2\}$.

(2) Let $i = 0$, $\sigma_0 = \sigma$ and $S_0 = S$.

(3) Let j be the smallest number from S_i which does not appear in the sequence σ_i.

(4) Draw an edge between j and the first node from σ_i.

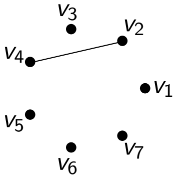
(5) Remove the first element from the sequence $\sigma_i \Rightarrow$ the sequence σ_{i+1}. Remove j from $S_i \Rightarrow$ the set S_{i+1}.

(6) When the sequence σ_{i+1} gets empty, draw an edge between the nodes left in S_{i+1}, and stop. Otherwise, increment i by 1 and goto step (3).

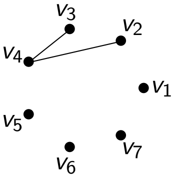
Prüfer's method

The construction of a labeled tree (1)

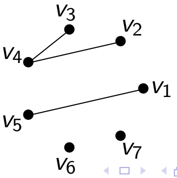
$$\sigma = \sigma_0 = \mathbf{4}, 3, 1, 3, 1$$
$$S = S_0 = \{1, \mathbf{2}, 3, 4, 5, 6, 7\}$$



$$\sigma_1 = \mathbf{3}, 1, 3, 1$$
$$S_1 = \{1, 3, \mathbf{4}, 5, 6, 7\}$$



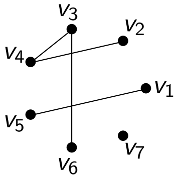
$$\sigma_2 = \mathbf{1}, 3, 1$$
$$S_2 = \{1, 3, \mathbf{5}, 6, 7\}$$



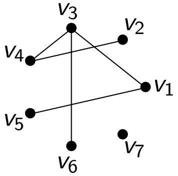
Prüfer's method

The construction of a labeled tree (2)

$$\sigma_3 = \mathbf{3}, 1$$
$$S_3 = \{1, 3, \mathbf{6}, 7\}$$

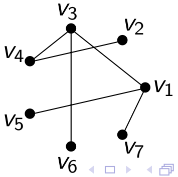


$$\sigma_4 = \mathbf{1}$$
$$S_4 = \{1, \mathbf{3}, 7\}$$



σ_5 is the empty sequence

$$S_5 = \{1, 7\}$$



References

Section 1.7 from

John M. Harris, Jeffry L. Hirst, Michael J. Mossinghoff. *Combinatorics and Graph Theory*. Second Edition. Springer 2008.