ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Discrete Mathematics

Course preparation group:
Nguyễn Khánh Phương
Đỗ Phan Thuận
Phạm Quang Dũng
Huỳnh Thanh Bình
Trần Vĩnh Đức
Bùi Quốc Trung
Đinh Viết Sang
Ban Hà Bằng

Content of Part 2

PART 1
COMBINATORIAL THEORY
(Lý thuyết tổ hợp)

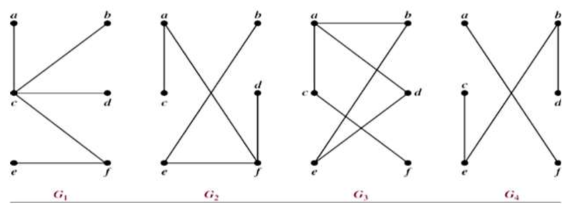
PART 2
GRAPH THEORY
(Lý thuyết đồ thị)

Contents

1. Tree and its properties

A **tree** is an undirected connected graph with no cycles.

Example 1. Which of the graphs are trees?

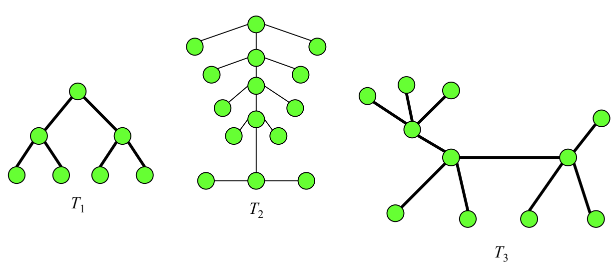


Solution: G_1, G_2

Note. G3: contains cycle {a,b,e,d,a}
G4: not connected

1. Tree and its properties

Forest: Graph containing no cycles that are not connected, but each connected component is a tree.



Forest *F* consists of 3 trees: T_1, T_2, T_3

1. Tree and its properties

Theorem. Given an undirected graph $G = (V,E)$, the following conditions are equivalent:

(1) G is a connected graph with no cycles. (Thus G is a tree by the above definition).

(2) For every two vertices $u, v \in V$, there exists exactly one simple path from u to v.

(3) G is connected, and removing any edge from G disconnects it (each edge of G is a bridge).

(4) G has no cycles, and adding any edge to G gives rise to a cycle. (Thus G is a maximal acyclic graph).

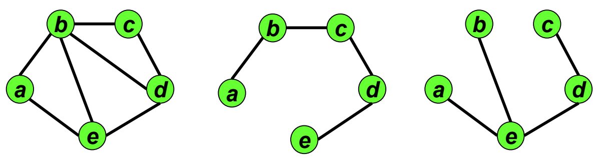
(5) G is connected and $|E| = |V| - 1$.

Contents

2.The spanning tree

Let $G=(V, E)$ be an undirected connected graph with vertex set V.
Tree $T=(V,F)$ *where* $F \subseteq E$ *is called* ***spanning tree*** *of* G

Undirected Connected graph without cycle

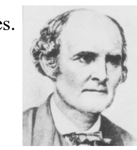


G   *Spanning tree* T_1 *Spanning tree* T_2

Graph G and its 2 spanning trees T_1 and T_2

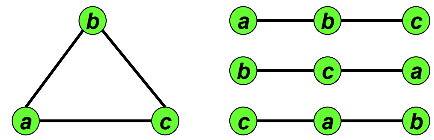
2. The spanning tree

Theorem (Cayley). A complete graph K_n has n^{n-2} spanning trees.



Arthur Cayley (1821 – 1895)

(A **complete graph** is a simple undirected graph in which every pair of distinct vertices is connected by a unique edge)



K_3 *3 spanning trees of K_3*

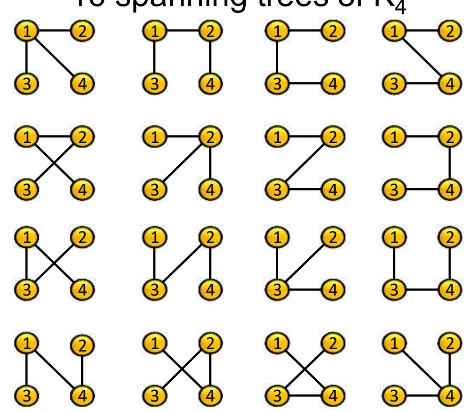
2. The spanning tree

Theorem. Every undirected connected graph contains a spanning tree.

Proof. Let G be an undirected connected graph.

- If G contains no cycle then G is its own spanning tree.
- If G contains a cycle: Removing any edge from the cycle gives a graph which is still connected. If the new graph contains a cycle then again remove one edge of the cycle. Continue this process until the resulting graph T contains no cycles. We have not removed any vertices so T has the same vertex set as G, and at each step of the above process we obtain a connected graph. Therefore T is connected and it is a spanning tree for G.

16 spanning trees of K_4



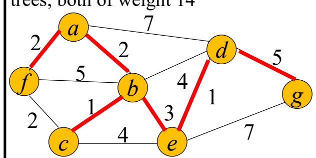
Contents

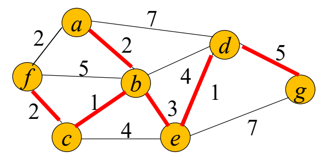
Weighted Graphs and Minimum Spanning Trees

Every undirected connected weighted graph G has a minimum spanning tree. Since G has only a finite number of spanning trees, one of them must have minimum weight.

Note: a given undirected connected weighted may have more than one minimum spanning tree.

Example: An undirected connected weighted graph with two minimal spanning trees, both of weight 14





As the number of spanning trees of G is very large (see Cayley's theorem), we could not solve this problem by brute force.

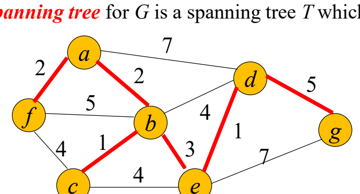
Weighted Graphs and Minimum Spanning Trees

Let $G=(V, E)$ be an undirected connected graph with vertex set V:

- For each edge (u,v) in E, we have a weight $w(u,v)$ specifying the cost (length of edge) to connect u and v.

For any subgraph H of G, we define the *weight of H*, denoted by $w(H)$, to be the sum of its edge weights:
$$w(H) = \sum_{e \in E(H)} c(e)$$

A ***minimum spanning tree*** for G is a spanning tree T which has the smallest weight



The weight of the spanning tree = 14

General scheme of the algorithm to find MST

Initialize: The minimum spanning tree T = ∅

Each step of the algorithm: one edge e which is the "safe" edge is chosen, subject only to the restriction that if adding edge e into T then T is still a tree (no cycle is created).

```
Generic-MST(G, c)
  T = ∅
  //T  is the subset edges of some minimum spanning tree
  while T is not the spanning tree do
        Finding edge (u, v) is "safe" edge for T
        T = T ∪{(u, v)}
  return T
```

Edge with smallest weight and insert it into T does not create cycle

Set T is always a subset of edges of some minimum spanning tree. This property is called the **invariant Property**.

An edge (u,v) is a **safe edge** for T if adding the edge to T does not destroy the invariant.

General scheme of the algorithm to find MST

Initialize: The minimum spanning tree T = ∅

Each step of the algorithm: one edge *e* which is the "safe" edge is chosen, subject only to the restriction that if adding edge *e* into T then T is still a tree (no cycle is created).

```
Generic-MST(G, c)
  T = ∅
  //T  is the subset edges of some minimum spanning tree
  while T is not the spanning tree do
       Finding edge (u, v) is "safe" edge for T
       T = T ∪{(u, v)}
  return T
```

How to find the "safe" edge ???:

The criteria to choose an edge at each step decides the process of two following minimum spanning tree algorithms (both use *greedy* strategies):

1. Kruskal
2. PRIM

How to find the "safe" edge?

T : set of edges of some spanning tree
Initialize: T = ∅

Kruskal algorithm

- *T* is **forest.**
- The "safe" edge added to *T* at each iteration is the edge with smallest weight **among edges connecting its connected components**.

Prim algorithm

- *T* is **tree**.
- The "safe" edge added to *T* at each iteration is the edge with smallest weight **among edges connecting the tree *T* to other vertex not in the tree**.