Data Structures and Algorithms
(資料結構與演算法)

Lecture 8: Tree

Hsuan-Tien Lin (林軒田)

htlin@csie.ntu.edu.tw

Department of Computer Science
& Information Engineering

National Taiwan University
(國立台灣大學資訊工程系)

intuition

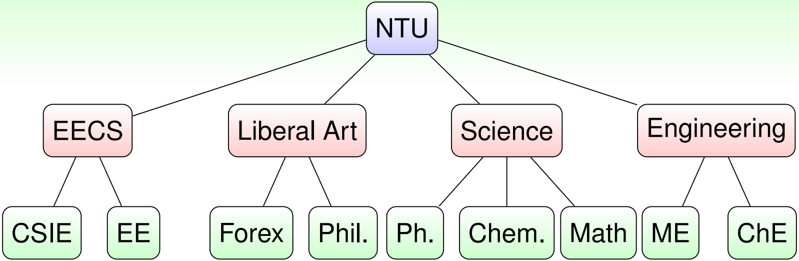
Nature of Data Structures

data structure	nature
array	indexed access
linked list	sequential access
stack/queue/deque	restricted (boundary) access
tree	hierarchical access

rotated from figure by Mysticsartdesign, licensed for free via Pixabay

next: tree

Visual Intuition of Tree



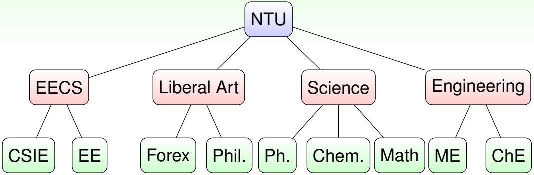
modified from tikz code by Stefan Kottwitz, licensed under CC-BY-SA 2.0 via TeXample.net

hierarchical (parent-child) relationship

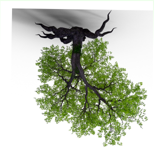
- organization structure
- file system
- document object model (e.g. HTML)

general-purpose data structures:
array ⟶ linked list ⟹ tree

Formal Definition of (Rooted) Tree



modified from tikz code by Stefan Kottwitz, licensed under CC-BY-SA 2.0 via TeXample.net



rotated from figure by Mysticsartdesign, licensed for free via Pixabay

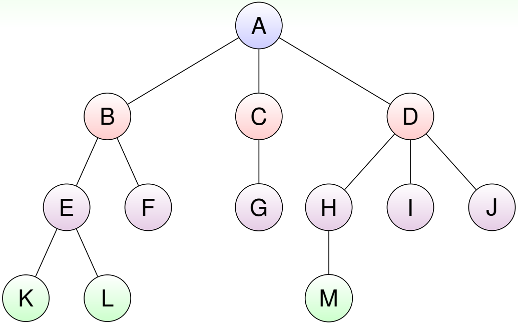
$$T \equiv (root; T_1, T_2, \ldots, T_n)$$

- recursive definition
- $T.root$ for starting tree access (like $L.head$)
- disjoint sub-trees $(T_1, \ldots, T_n)$
- recursion termination: T_{LEAF} with no sub-trees

rooted tree: usually illustrated with root at the top

terminologies

Height of Tree



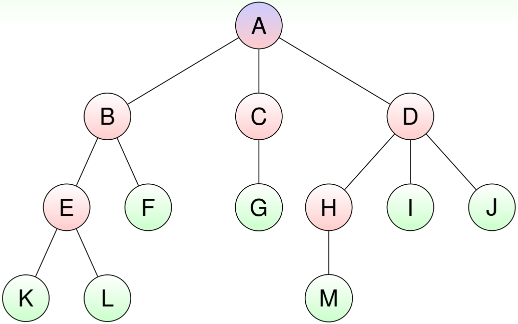
node depth
= # edges from root

- level 0: A (root)
- level 1: BCD
- level 2: EFGHIJ
- level 3: KLM

height = max depth + 1

usually want small height to access efficiently from root

Node of Tree



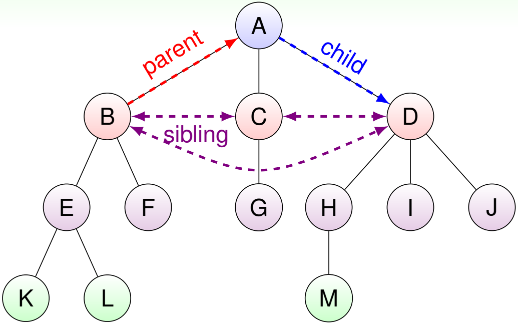
node degree

= # sub-trees

- internal nodes (degree $>$ 0): ABCDEH
- external nodes (degree $=$ 0): others [leaves]

# leaves sometimes called breadth of tree

‘Family Relatives’ of Tree



- ancestors of L: EBA (path to root)
- descendants of D: HIJM (sub-tree nodes)

‘family tree’ metaphor:
for illustrating tree operations lively

implementations

Basic Algorithms (Operations) for Tree

$$T \equiv (root; T_1, T_2, \ldots, T_n)$$

linked list

```
GET-DATA(L.node)
    return L.node.data
```

⟹

tree

```
GET-DATA(T.node)
    return T.node.data
```

```
GET-NEXT(L.node)
    return L.node.next
```

⟹

```
GET-SUBTREE(T.node, index)
    return T.node.subtree.GET(index)
```

```
INSERT-AFTER(L.node, data)
    newNode = NODE(data, L.node.next)
    L.node.next = newNode
```

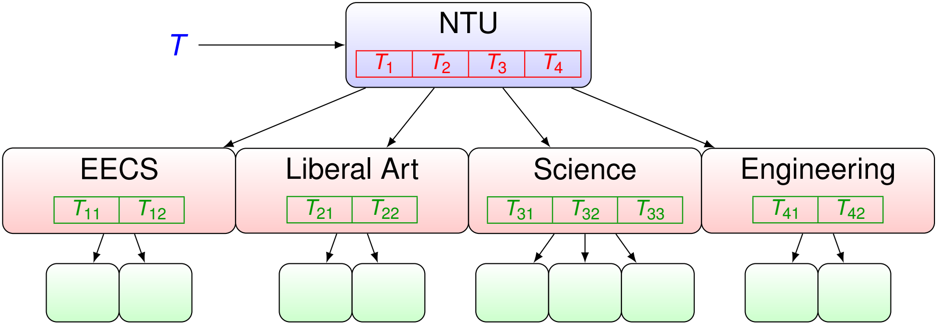
⟹

```
INSERT-CHILD(T.node, data)
    newNode = NODE(data, []) // no child
    T.node.subtree.INSERT(newNode)
```

(general-purpose) tree: extension of linked list

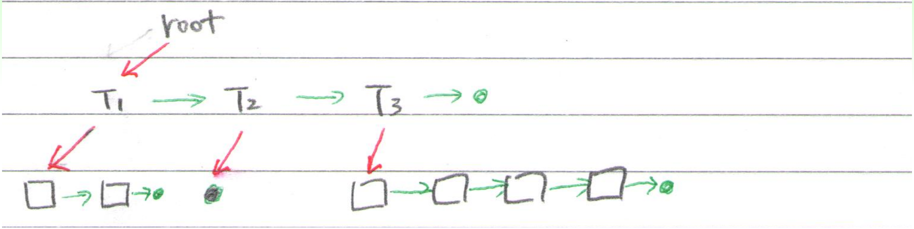
Representing Trees with Arrays

$$T \equiv (root; T_1, T_2, \ldots, T_n)$$



can also have parent link (like doubly linked list)

Representing Trees with Linked Lists



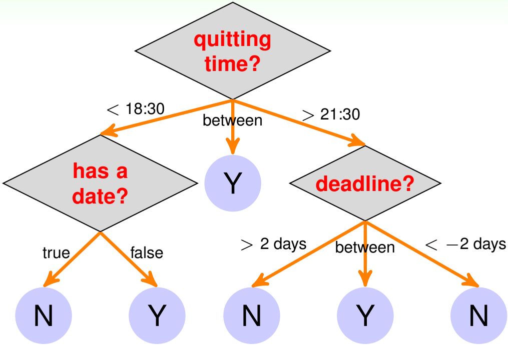
called left-child right-sibling
# link per node?

application: decision tree

Decision Tree for Watching MOOC Lectures

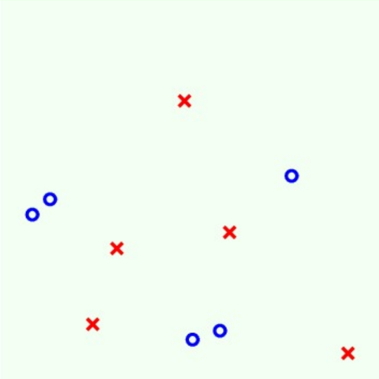
figure taken from Lecture 209 of ML Techniques

- **base decision**: leaf at end of path *t*
- **condition** on **internal nodes**



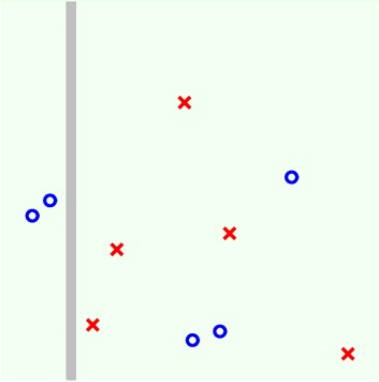
decision tree: arguably one of the most **human-mimicking models** in machine learning

C&RT Algorithm for Decision Tree



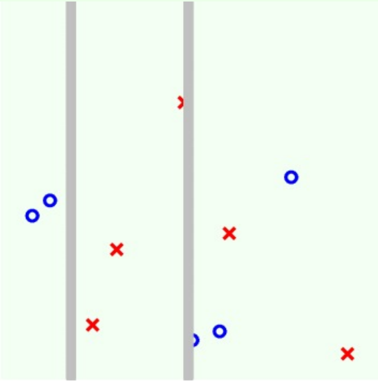
```
function DecisionTree(data)
if cannot branch anymore
        return best color
else
    1 learn branching criteria to cut the plane
    2 split data to 2 parts
    3 build sub-tree from each part
    4 return (branching criteria; sub-trees)
```

C&RT Algorithm for Decision Tree



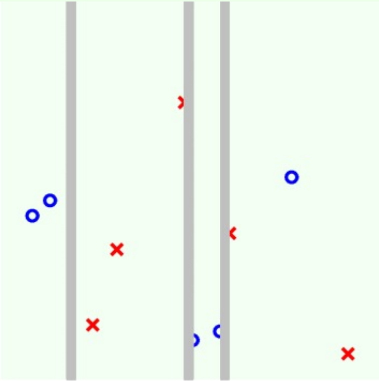
```
function DecisionTree(data)
if cannot branch anymore
        return best color
else
    1 learn branching criteria to
      cut the plane
    2 split data to 2 parts
    3 build sub-tree from each
      part
    4 return (branching criteria;
      sub-trees)
```

C&RT Algorithm for Decision Tree



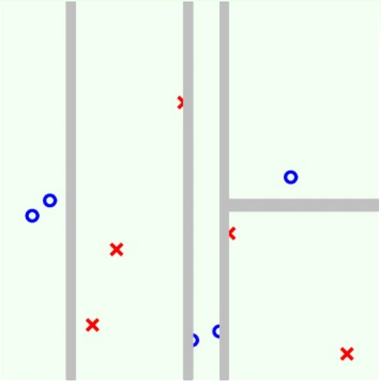
```
function DecisionTree(data)
if cannot branch anymore
        return best color
else
    1 learn branching criteria to
      cut the plane
    2 split data to 2 parts
    3 build sub-tree from each
      part
    4 return (branching criteria;
      sub-trees)
```

C&RT Algorithm for Decision Tree



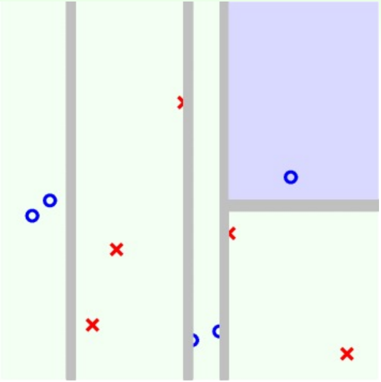
```
function DecisionTree(data)
if cannot branch anymore
        return best color
else
    1 learn branching criteria to
      cut the plane
    2 split data to 2 parts
    3 build sub-tree from each
      part
    4 return (branching criteria;
      sub-trees)
```

C&RT Algorithm for Decision Tree



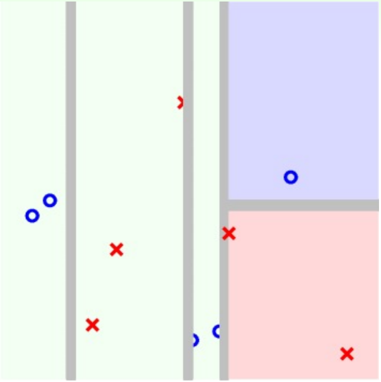
```
function DecisionTree(data)
if cannot branch anymore
        return best color
else
    1 learn branching criteria to
      cut the plane
    2 split data to 2 parts
    3 build sub-tree from each
      part
    4 return (branching criteria;
      sub-trees)
```

C&RT Algorithm for Decision Tree



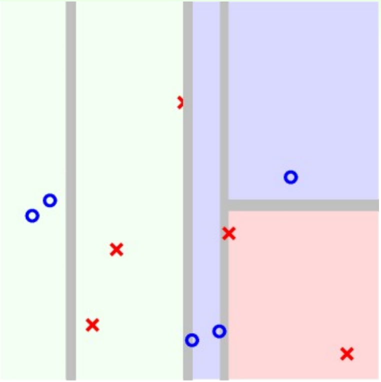
```
function DecisionTree(data)
if cannot branch anymore
        return best color
else
    1 learn branching criteria to cut the plane
    2 split data to 2 parts
    3 build sub-tree from each part
    4 return (branching criteria; sub-trees)
```

C&RT Algorithm for Decision Tree



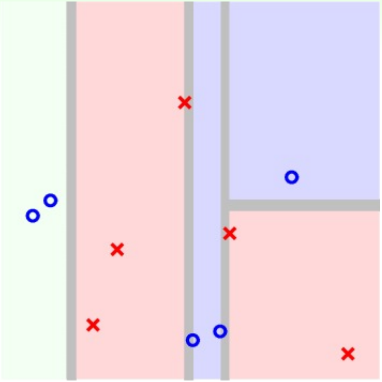
```
function DecisionTree(data)
if cannot branch anymore
        return best color
else
    1 learn branching criteria to cut the plane
    2 split data to 2 parts
    3 build sub-tree from each part
    4 return (branching criteria; sub-trees)
```

C&RT Algorithm for Decision Tree



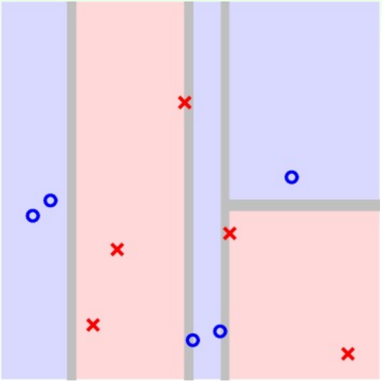
```
function DecisionTree(data)
if cannot branch anymore
        return best color
else
    1 learn branching criteria to
      cut the plane
    2 split data to 2 parts
    3 build sub-tree from each
      part
    4 return (branching criteria;
      sub-trees)
```

C&RT Algorithm for Decision Tree



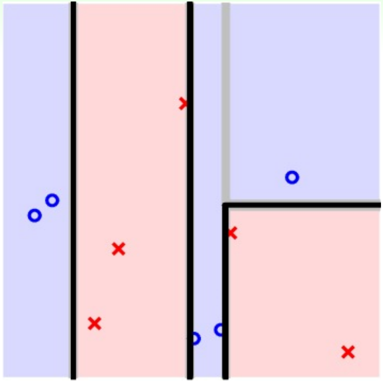
```
function DecisionTree(data)
if cannot branch anymore
        return best color
else
    1 learn branching criteria to
      cut the plane
    2 split data to 2 parts
    3 build sub-tree from each
      part
    4 return (branching criteria;
      sub-trees)
```

C&RT Algorithm for Decision Tree



```
function DecisionTree(data)
if cannot branch anymore
        return best color
else
    1 learn branching criteria to cut the plane
    2 split data to 2 parts
    3 build sub-tree from each part
    4 return (branching criteria; sub-trees)
```

C&RT Algorithm for Decision Tree



C&RT

```
function DecisionTree(data)
if cannot branch anymore
        return best color
else
    1 learn branching criteria to
      cut the plane
    2 split data to 2 parts
    3 build sub-tree from each
      part
    4 return (branching criteria;
      sub-trees)
```

C&RT: ‘divide-and-conquer’

(based on **selected components**

of **CART^{TM} of California Statistical Software**)

Summary

Lecture 8: Tree

- intuition

  hierarchical access from root of tree
- terminologies

  family relatives useful for describing node relations
- implementations

  more complicated links than array/linked list
- application: decision tree

  divide-and-conquer model in machine learning