Sets and Bit-sets

27 Nov 2009
CMPT140
Dr. Sean Ho
Trinity Western University

TRINITY WESTERN UNIVERSITY

Set operations (theory, not Py)

- A set is an unordered collection of items
- Set membership: test if an item is in the set
- Set union: A ∪ B:
  - Anything that's in either A or B
- Set intersection: A ∩ B:
  - Those items which are in both A and B
- Set difference: A – B (or A \ B):
  - Those in A but not in B
- Set symmetric difference: A ^ B:
  - Those in exactly one of A or B

Sets in Python

- Python has a built-in type for sets (as does M2):
  - Instantiate with any iterable (e.g., a list):

    ```
    bagOfApples = set( [ 'Fuji', 'Gala', 'Red Delicious' ] )
    ```

  - Add an apple to the bag:

    ```
    bagOfApples.add( 'Rome' )
    ```

  - Remove an existing apple from the bag:

    ```
    bagOfApples.remove( 'Rome' )
    ```

  - Check if an apple is in the bag:

    ```
    if 'Fuji' in bagofApples:
    ```

- See Python documentation: http://docs.python.org/lib/types-set.html

Python set operators

- Operators for Python sets:
  - Union of two sets: .union() or |

```
bagOfApples.union( yourApples )
bagOfApples | yourApples
```

  - Intersection of two sets: .intersection() or &
  - Difference of two sets: .difference() or –
  - Symmetric diff: .symmetric_difference() or ^
  - Subset: .issubset() or <=
    - A <= B: everything in A is also in B
  - Superset: .issuperset() or >=

Bitsets

- Another way to implement sets is using a bitset: binary form of an integer represents flags:
- e.g., file permissions: a user may have permission to read, write, and/or execute
  - Let 4=read (r) , 2=write (w), 1=execute (x)
  - So the number 5 represents read+execute
- In Python, using bitwise shift operators:

```
readFlag = 1 << 2      # fancy way of saying 4
writeFlag = 1 << 1     # 2
execFlag = 1 << 0      # 1
```

Bitsets

- We can combine these flags using bitwise logical operators: or (|), and (&)

```
5 | 3 == 7          # (101) | (011) == (111)
5 & 3 == 1          # (101) & (011) == (001)
```

- Add read permission if not already there:

```
myPerms |= readFlag
```

- Check if we have write permission:

```
if myPerms & writeFlag:
```

- Need read perm on both file and directory:

```
if filePerms & dirPerms & writeFlag:
```