Università di Roma Tor Vergata

Tutoraggio
Union Find

Laura Trivelloni
4 Dicembre 2017

Ingegneria degli algoritmi
A.A. 2017/2018

Union Find

- Operazioni
  - **makeset**
  - **find**
  - **union**
- Strutture
  - **QuickFind**
  - **QuickUnion**
  - **QuickFind bilanciato**
  - **QuickUnion bilanciato**
  - **QuickUnion con euristica di path compression**

Ereditarietà in Python

Alcuni vantaggi:

- ➔ **Ereditare** tutti i metodi di una classe
- ➔ **Sovrascrivere** alcuni metodi
- ➔ **Richiamare** i metodi sovrascritti della classe ereditata

Ereditarietà in Python

Alcuni vantaggi:

- ➔ **Ereditare** tutti i metodi di una classe
- ➔ **Sovrascrivere** alcuni metodi
- ➔ **Richiamare** i metodi sovrascritti della classe ereditata

Ma perché tutto questo?

Ereditarietà in Python

- Per ereditare una classe:
  - **class ClasseFiglia(ClasseEreditata)**
- Per richiamare un metodo sovrascritto, all'interno di un metodo della classe figlio:
  - **super( ).metodo( )**
    - ***super( )*** è il riferimento alla classe ereditata detta anche *superclasse*

Ereditarietà in Python - Esempio

```
class Person:
    def __init__(self, first, last, age):
        self.firstname = first
        self.lastname = last
        self.age = age

    def __str__(self):
        return "Name: " + self.firstname + " "
                + self.lastname + " - Age: " + str(self.age)
```

Ereditarietà in Python - Esempio

```
class Student(Person):

    def __init__(self, first, last, age, number):
        super().__init__(first, last, age)          # riferimento a Person
        self.number = number

    # overwritten method
    def __str__(self):
        return super().__str__() + " - IDNumber: " +  self.number
```

Ereditarietà in Python - Esempio

```
def main() :

    x = Person("Jon", "Snow", 30)
    y = Student("Samwell", "Tarly", 28, "0123457")

    print(x)            # cosa stampa?

    print(y)            # e qui?
```

Profiling in Python

- ottenere una serie di **statistiche** che descrivano quanto spesso e per quanto tempo varie parti del codice siano in esecuzione
- profiling **non invasivo**: senza aggiungere codice esterno al nostro script Python

cProfile e pstats

- opzione ***-m*** dell'interprete Python che esegue un modulo come fosse uno script!
  - **python -m cProfile -o <output_file.prof> <input_file.py>**
- profiler eseguito sul file di input e risultati salvati sul file di output
- script per **pstats** che prenda un file di profiling e stampi alcune statistiche di base

cProfile e pstats

- prendere in input il nome (o il percorso) di un file:
- stampare statistiche di base da un file di profiler:

```
import pstats
...
p = pstats.Stats(<file.prof>)

p.strip_dirs( ).sort_stats('time').print_stats( )
```