UNIVERSITATEA BABEŞ-BOLYAI
Facultatea de Matematică şi Informatică

ALGORITMI ŞI PROGRAMRE

Testarea și inspectarea programelor

Laura Dioşan

Conținut curs

- Programming in the large
  - Introducere în procesul de dezvoltare software
  - Programare procedurală
  - Programare modulară
  - Tipuri definite de utilizator
  - Principii de dezvoltare a softului
  - **Testarea și inspectarea programelor**
- Programming in the small
  - Recursivitate
  - Complexitatea algoritmilor
  - Algoritmi de căutare
  - Algoritmi de sortare
  - Metode de rezolvare a problemelor
- Recapitulare

Sumar

- Testare
  - Concept
  - Alegerea datelor de testare
  - Nivele de testare
  - Testare în Python

Faze în viața unui program (produs soft)

- Stabilirea cerințelor beneficiarului
- Analiza (descompunerea problemei în subprobleme)
- Proiectare (stabilirea TAD-urilor și a structurii/nivelelor aplicației)
- Implementare
- **Testare**

Testarea

- Testarea
  - Observarea comportamentului unei aplicații pentru diferite date de test
  - Verificarea programului
    - ≠ demonstrarea corectitudinii
  - Poate indica incorectitudinea programului
  - Activitate continuă
- Probleme ale testării
  - Spațiul datelor de intrare ale unei funcții nu poate fi acoperit complet
  - Anumite elemente nu pot fi controlate (SO, hardware, bibliotecti, etc.)
- Întrebări posibile
  - Cum se aleg datele de intrare?
  - Câte teste trebuie executate?
  - Cum știm dacă un program a lucrat corect la un test?
- Determinarea datelor de test
  - Testare exhaustivă
  - Testare cu ajutorul cutiei negre
  - Testare cu ajutorul cutiei transparente

Metode de testare

- Testare exhaustivă
  - Verificarea unui program pentru toate posibilele date de intrare
  - Imposibil de aplicat în practică → trebuie ale un număr finit de cazuri de testare

- Testare incompletă
  - Verificarea unui program doar pentru anumite date de intrare
  - Posibil de aplicat în practică
  - Cum se aleg aceste date de test?
    - Testare cu ajutorul cutiei negre
    - Testare cu ajutorul cutiei transparente

Metode de testare

- Testare cu ajutorul cutiei negre (black-box testing)
  - Alegerea cazurilor de testare în funcție de specificarea algoritmilor (date și rezultate), fără a analiza codul sursă
  - Se testează dacă aplicația face ceea ce ar trebui să facă
    - Valori obișnuite
    - Valori limită
    - Condiții de eroare
  - Se verifică dacă aplicația respectă specificarea

- Testare cu ajutorul cutiei transparente (white-box testing)
  - Alegerea cazurilor de testare pe baza codului sursă al algoritmului
    - Toate ramurile de execuție trebuie acoperite prin diferite date de test

Testare – exemple

```
def isPrime(x):
    '''
    checks if a number is prime or not
    Data:    x - a potivie integer number
    Results: True, if x is prime,
             False if x is composed
             raise ValueError if x < 0
    '''
S1    if (x < 0):
S2        raise ValueError("give a
           positive number to be
           tested...")
S3    else:
S4        if (x < 2):
S5            return False
S6        else:
S7            d = 2;
S8            while (d * d <= x):
S9                if (x % d == 0):
S10                   return False
S11               d = d + 1
S12           return True
```

```
def testIsPrime():
    #black box testing
    #test a prime number
    assert isPrime(5) == True
    #test a composed number
    assert isPrime(15) == False
    #test 0
    assert isPrime(0) == False
    #test a negative number
    try:
        isPrime(-3)
        assert False
    except ValueError as ex:
        print("some errors: " + str(ex))
        assert True
    #white box testing (cover all paths)
    # x < 0 => S1, S2
    try:
        isPrime(-5)
        assert False
    except ValueError as ex:
        print("some errors: " + str(ex))
        assert True
    #0 <= x < 2    => S3, S4, S5
    assert isPrime(0) == False
    assert isPrime(1) == False
    #x = 2 or x = 3 => S3, S6, S7, S12
    assert isPrime(2) == True
    assert isPrime(3) == True
    #x = 11    => S3,S6,S7,S8,S11,S8,S11,S12
    assert isPrime(11) == True
    #x = 15    => S3,S6,S7,S8,S11,S8,S9,S10
    assert isPrime(15) == False

testIsPrime()
```

Testare – exemple

```
def sumOfEvenValues(l):
    '''
    computes the sum of even values from a list
    Data: l - a list of integers
    Results: sum of even values of l
    '''
    s = 0
    for i in range(0, len(l)):
        if (l[i] % 2 == 0):
            s = s + l[i]
    return s
```

```
def changeElement(l, pos, el):
    '''
        change the pos-th element of list to el
        Data: a list l, a position, a new element
        Results: list l' with l[pos] == el
                 raise IndexError if pos < 0 or
                 pos >= len(l)
    '''
    if (pos < 0):
        raise IndexError("pos must be positive")
    else:
        if (pos >= len(l)):
            raise IndexError("position must be
                smaller to the length of list")
        else:   #a valid position
            l[pos] = el
            return l
```

```
def testSumOfEvenValues():
    #empty list
    assert sumOfEvenValues([]) == 0
    #no even value
    assert sumOfEvenValues([5,1,7,3]) == 0
    #one even value
    assert sumOfEvenValues([5,2,7,3]) == 2
    #more evne values
    assert sumOfEvenValues([5,2,7,4,6,3])==12
    #all values are even
    assert sumOfEvenValues([4,8,2,6]) == 20

testSumOfEvenValues()
```

```
def testChangeElement():
    #black box testing and white box testing
    #negative position
   try:
        changeElement([1,3,9,2], -2, 5)
        assert False
   except IndexError as ex:
        print("some errors: " + str(ex))
    #position > len(l)
   try:
        changeElement([1,3,9,2], 6, 5)
        assert False
   except IndexError as ex:
        print("some errors: " + str(ex))
    #valid data
   assert changeElement([1,3,9,2],2,5)==[1,3,5,2]

testChangeElement()
```

Testare – comparare metode

White-box testing	Black-box testing
+ cunoașterea codului = descrierea și testarea lui mai ușoară	+ eficientă pentru coduri sursă mari
+ poate ajuta la identificarea unor defecte ascunse	+ nu necesită acces la cod
+ poate ajuta la optimizarea codului	+ separarea programator vs. Tester
+ Acoperiri mari	
- Probleme cu codul care lipsește	- Acoperiri reduse
- Necesită cunoșterea bine a codului sursă	- Teste ineficiente
- Necesită acces la codul sursă	

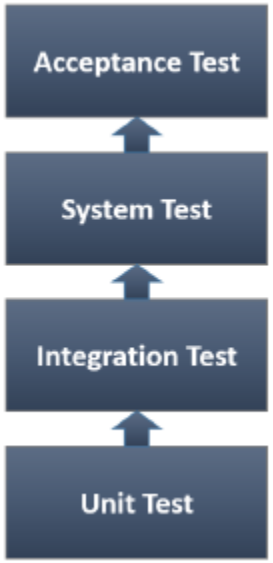
Nivele ale testării

- Testele se pot grupa
  - Pe baza locului în care au fost dezvoltate
  - Pe baza specificității lor

- Tipuri de testare
  - Testare manuală
  - Testare automată
    - Scrierea unui program care să efectueze testarea
      - Controlul execuției testelor
      - Compararea valorilor obținute cu cele așteptate

Nivele ale testării

- Nivele de testare
  - Testare la nivel de unitate
    - Teste care verifică funcționarea unei anumite păr (eg. a unei funcții)
    - Testare izolată –părțile de cod se testează indepe unele de altele
  - Testare de integrare
    - Testarea părților (combinate ale) unui sistem
    - Abordare bottom-up, bazată pe unit testing
  - Testare de sistem
    - Consideră modul în care un program funcționează ca un întreg
    - După ce toate modulele au fost testate (și corectate)
  - Testare de acceptare
    - Se verifică dacă sistemul compilează cu cerințele clientului și este gata de utilizare



Testare în Python

- Modulul UnitTest
  - Testare automată
  - Acces la codul setup și shutdown pentru teste
  - Agregarea testelor în colecții
  - Independența testelor față de framework

Recapitulare

- Testare
  - Concept
  - Alegerea datelor de testare
  - Nivele de testare
  - Testare în Python

Cursul următor

- Programming in the large
  - Introducere în procesul de dezvoltare software
  - Programare procedurală
  - Programare modulară
  - Tipuri definite de utilizator
  - Principii de dezvoltare a softului
  - Testarea și inspectarea programelor
- Programming in the small
  - **Recursivitate**
  - **Complexitatea algoritmilor**
  - Algoritmi de căutare
  - Algoritmi de sortare
  - Metode de rezolvare a problemelor
- Recapitulare

Materiale de citit şi legături utile

1. Limbajul Python
http://docs.python.org/3/reference/index.html

2. Biblioteca standard Python
http://docs.python.org/3/library/index.html

3. Tutorial Python
http://docs.python.org/3/tutorial/index.html

4. Frentiu, M., H.F. Pop, Fundamentals of Programming, Cluj University Press, 2006, 220 pagini

5. Kent Beck.Test Driven Development: By Example. *Addison-Wesley* Longman, 2002
http://en.wikipedia.org/wiki/Test-driven_development

6. Martin Fowler. Refactoring. Improving the Design of Existing Code. *Addison-Wesley*, 1999
http://refactoring.com/catalog/index.html

- Informaţiile prezentate au fost colectate din diferite surse de pe internet, precum şi din cursurile de Fundamentele Programării ţinute în anii anteriori de către:
  - Lect. Dr. Adriana Guran – www.cs.ubbcluj.ro/~adriana
  - Prof. Dr. Istvan Czibula - www.cs.ubbcluj.ro/~istvanc
  - Conf. Dr. Andreea Vescan -www.cs.ubbcluj.ro/~avescan
  - Lect. Dr. Ioan Lazăr -www.cs.ubbcluj.ro/~ilazar
  - Lect. Dr. Molnar Arthur – www.cs.ubbcluj.ro/~arthur