Examen TP

Mihaila Andra
214
Subiect 16
pag 1

1) def partition(arr, start, end, comp_func):
    pivot = arr[start]
    low = start + 1
    high = end
    while True:
        while low <= high and comp_func(arr[high], pivot):
            high = high - 1
        while low <= high and not comp_func(arr[low], pivot):
            low = low + 1
        if low <= high:
            arr[low], arr[high] = arr[high], arr[low]
        else:
            break
    arr[start], arr[high] = arr[high], arr[~~low~~ start]
    return high

def quickSort(arr, start, end, comp_func):
    if start >= end:
        return
    p = partition(arr, start, end, comp_func)
    quickSort(arr, start, p-1, comp_func)
    quickSort(arr, p+1, end, comp_func)

~~f~~ quickSort(array, 0, len(array)-1, lambda x, y: x[1] < y[1])

Mihăilă Andra
214
Subiect 16

2.

```
def f(n):
    """
    Funcția creează o listă cu (n+1) elemente, pe poziția 1 se
    află elementul 1.
    Funcția calculează elementul de pe poziția i ca fiind
    suma dintre elementele de pe i-1 și i-2, adică primele
    două elemente din fața lui i.
    Funcția ridică ValueError dacă se introduce un nr.
    negativ și returnează n dacă se introduce un
    număr între 0 și 1 inclusiv.
    Funcția returnează numărul din listă de pe poziția n
    """
```

```
def test_f(n):
    try:
        assert f(2) == 1
        assert f(1) == 1
        assert f(3) == 2
        assert f(-1) == 5
    except ValueError():
        pass
```

Mihăilă Andra
214
Subiect 16
pag 3

(3) Complexitate de timp:

$\sum_{i=1}^{n}$

Pentru cele 2 for: $\sum_{i=1}^{n} \sum_{j=1}^{i} 1 = \sum_{i=1}^{n} i \in n \cdot i$

Dar $i \to n$ $\Rightarrow T(n) = n^2 \in \Theta(n^2)$

Pentru while: $T(n) = \log_2 n$

$\Rightarrow \Theta(n^2 \log_2 n)$

Complexitate de spațiu: $T(1)$ (nu avem memorie alocată pentru variabile auxiliare)

(4)

```
def findMin(l):
    if len(l) == 1: return l[0]
    mid = len(l)//2
    min1 = findMin(l[:mid])
    min2 = findMin(l[mid:])
    if min1 > min2:
        return min2
    return min1
```

5. Candidat: Pentru ca o secvență să fie candidat, aceasta trebuie să fie formată din paranteze și acolade și să nu aibă mai multe elemente decât n.

Consistent: Pentru a fi consistent, un candidat trebuie să respecte cerința: fiecare paranteză și acolada trebuie să se închidă corect. și să aibă n elemente.

Soluție: O soluție este candidatul care este consistent, adică îndeplinește toate condițiile: lungimea unei secvențe să fie n, aceasta să fie alcătuită din paranteze și acolade, și să se închidă corect.

Mihăilă Andrea
214
Subiect 16
pag 4