Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή ΗΜ&ΜΥ
Προγραμματιστικές Τεχνικές
2ο εξάμηνο

Λύσεις Παλαιών Θεμάτων

Γερακάρης Βασίλης
<vgerak@gmail.com>

1 Σεπτέμβριος 2007

1.1 Θέμα 1 ο

Κάθε βασική λειτουργία (tfetch, tstore, tC, tH, κ.λ.π.) = 1 μονάδα χρόνου, να υπολογιστεί χρόνος εκτέλεσης και άνω ασυμπτωτικό όριο O() για τον παρακάτω κώδικα:

```
int K = 0;
for (int i = 0; i < n; ++i) //"i-O"
    for (int j = 1; j < n; ++j)
        ++K;
```

Είναι $O(n^2)$ -> n (εξωτερική for) * n (εσωτερική for)

1.2 Θέμα 2 ο

Θεωρώ ότι η isEmpty επιστρέφει 1 αν η στοίβα είναι άδεια, αλλιώς επιστρέφει 0. Επιπλέον, η pop έχει ως τιμή επιστροφής την τιμή του στοιχείου που αφαιρείται.

```
int product (stack s)
{
    int acc = 1;    //1 is the neutral element for multiplication
    while (isEmpty(s))
        acc *= pop(s);
    return acc
}
```

```
//Easier to comprehend
int product (stack s)
{
    int acc = 1;
    int temp;
    while (isEmpty(s) != 1) {
        temp = pop(s);
        acc = acc * temp
    }
    return acc
}
```

1.3 Θέμα 3 ο

Cheat answer: Δε θα τυπωθεί τίποτα, θα φάει syntax error στην "*printf(“ΤΕΛΟΣ \n”;*

Serious answer:
A)

- f(10) -> τυπώνει 10, μπαίνει στο "n%2 == 0" (c)
- f(5) -> τυπώνει 5, μπαίνει στο "else" (c)
- f(16) -> τυπώνει 16, μπαίνει στο (b)
- f(8) -> τυπώνει 8, μπαίνει στο (b)
- f(4) -> τυπώνει 4, μπαίνει στο (b)
- f(2) -> τυπώνει 2, μπαίνει στο (b)
- f(1) -> τυπώνει 1, μπαίνει στο "n==1", τυπώνει "ΤΕΛΟΣ" και τερματίζει

Β) Θα τυπωθούν ακριβώς τα ίδια μηνύματα, με αντίστροφη σειρά:
ΤΕΛΟΣ, 1, 2, 4, 8, 16, 5, 10

1.4 Θέμα 4 ο

Υποθέτουμε ότι οι stdlib.h και string.h έχουν γίνει include στο αρχείο μας.
Υποθέτουμε επίσης ότι η δομή που χειριζόμαστε είναι η παρακάτω.

```
struct node {
    char *word;
    struct node *next;
};
```

```
struct node *remDupl(struct node *myList)
{
    struct node *ptr;
    struct node *temp;
    if (myList == NULL)
        return myList;  //check if empty list
    ptr = myList;
    while (ptr->next != NULL) {
        if (strcmp(ptr->word, ptr->next->word) == 0) { //if we find string match
            temp = ptr->next->next; //keep where the 2nd node points
            free(ptr->next);        //delete node
            ptr->next = temp;       //restore pointer
        } else
            ptr = ptr->next;        //check next 2 nodes
    }
}
```

1.5 Θέμα 5 ο

A)

```
int c = 0;  //global

int traverseCount(tree t)
{
    if(t == NULL)
        return 0;
    if(t->left != NULL && t->right != NULL)
        ++c;
    traverseCount(t->left);
    traverseCount(t->right);
    return c;
}
```

B) 5 -> 4 -> 2 -> 8 -> 6 -> 7 -> 9

1.6 Θέμα 6 ο

Αρχικά : [2,1,4,1,6,8,3,5]
i = 7:
j = 1 : [1,2,4,1,6,8,3,5] (swap 1,2)
j = 2 : [1,2,4,1,6,8,3,5]
j = 3 : [1,2,1,4,6,8,3,5] (swap 1,4)
j = 4 : [1,2,1,4,6,8,3,5]
j = 5 : [1,2,1,4,6,8,3,5]
j = 6 : [1,2,1,4,6,3,8,5] (swap 3,8)
j = 7 : [1,2,1,4,6,3,5,**8**] (swap 5,8)
i = 6:

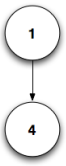
j = 1 : [1,2,1,4,6,3,5,**8**]
j = 2 : [1,1,2,4,6,3,5,**8**] (swap 1,2)
j = 3 : [1,1,2,4,6,3,5,**8**]
j = 4 : [1,1,2,4,6,3,5,**8**]
j = 5 : [1,1,2,4,3,6,5,**8**] (swap 3,6)
j = 6 : [1,1,2,4,3,5,**6**,**8**] (swap 5,6)

i = 5:
j = 1 : [1,1,2,4,3,5,**6**,**8**]
j = 2 : [1,1,2,4,3,5,**6**,**8**]
j = 3 : [1,1,2,4,3,5,**6**,**8**]
j = 4 : [1,1,2,3,4,5,**6**,**8**] (swap 3,4)
j = 5 : [1,1,2,3,4,**5**,**6**,**8**]

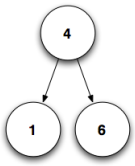
i = 4: Καμία αντιμετάθεση, ο πίνακας είναι ταξινομημένος.

1.7 Θέμα 7 ο

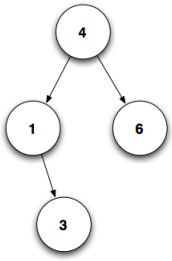
Εισαγωγή 1,4:



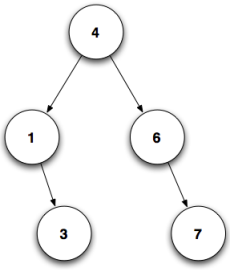
Εισαγωγή 6 (rebalance):



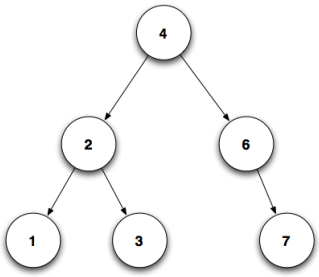
Εισαγωγή 3:



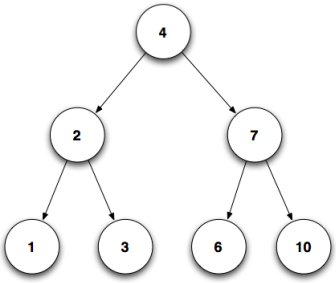
Εισαγωγή 7:



Εισαγωγή 2 (rebalance):



Εισαγωγή 10 (rebalance):



2 Ιούλιος 2011

2.1 Θέμα 1 ο

Κάθε βασική λειτουργία (tfetch, tstore, tC, tH, κ.λ.π.) = 1 μονάδα χρόνου, να υπολογιστεί χρόνος εκτέλεσης και άνω ασυμπτωτικό όριο O() για τον παρακάτω κώδικα:

```
int K = 0;
for (i = 0; i < n; i++)
    for (int j = 0; j < n; ++j) // "j<m"
        f(n);   //Π          f(n): O(n)
```

Είναι $O(n^3)$ -> n(εξωτερική for) * n(εσωτερική for) * O(n))

2.2 Θέμα 2 ο

```c
#define ARRSIZE 100

int evenElementsSum (int* array)
{
    int sum = 0;
    int i;

    for (i = 0; i < ARRSIZE; i += 2)
        sum += *(array + i);
    return sum;
}
```

2.3 Θέμα 3 ο

Λύση γραμμικού χρόνου:

```c
node* removeNodes(node* list, int num)
{
    node **toDelete;
    node *ptr = list
    int count = 0
    toDelete = (node *) malloc (num * sizeof(node *));
    while (ptr != NULL) {
        toDelete[count % n] = ptr;
        ptr = ptr->next;
        ++count;
    }
```