

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



Λύσεις Παλαιών Θεμάτων

Γερακάρης Βασίλης
<vgerak@gmail.com>

1 Σεπτέμβριος 2007

1.1 Θέμα 1 ° - Πολυπλοκότητα

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:

```
1 int K = 0;
2 for (int i = 0; i < n; ++i) /"i-0" typo mistake
3     for (int j = 1; j < n; ++j)
4         ++K;
```

Είναι $O(n^2)$ -> n (εξωτερική for) * n (εσωτερική for) * $O(1)$

1.2 Θέμα 2 ° - Στοίβα

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.

```
1 int product (stack s)
2 {
3     int acc = 1; //1 is the neutral element for multiplication
4     while (isEmpty(s))
5         acc *= pop(s);
6     return acc
7 }
```

```
1 //Easier to comprehend
2 int product (stack s)
3 {
4     int acc = 1;
5     int temp;
6     while (isEmpty(s) != 1) {
7         temp = pop(s);
8         acc = acc * temp
9     }
10    return acc
11 }
```

1.3 Θέμα 3 ° - Κλήση Αναδρομικής Συνάρτησης

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

B) Θα τυπωθούν ακριβώς τα ίδια μηνύματα, με αντίστροφη σειρά:

ΤΕΛΟΣ, 1, 2, 4, 8, 16, 5, 10

1.4 Θέμα 4 ° - Συνδεδεμένη Λίστα

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

```
1 struct node {
2     char *word;
3     struct node *next;
4 };

1 struct node *remDupl(struct node *myList)
2 {
3     struct node *ptr;
4     struct node *temp;
5     if (myList == NULL)
6         return myList; //check if empty list
7     ptr = myList;
8     while (ptr->next != NULL) {
9         if (strcmp(ptr->word, ptr->next->word) == 0) { //if we find string match
10             temp = ptr->next->next; //keep where the 2nd node points
11             free(ptr->next); //delete node
12             ptr->next = temp; //restore pointer
13         } else
14             ptr = ptr->next; //check next 2 nodes
15     }
16 }
```

1.5 Θέμα 5 ° - Δυαδικό Δέντρο

A)

```
1 int c = 0; //global
2
3 int traverseCount(tree t)
4 {
5     if(t == NULL)
6         return 0;
7     if(t->left != NULL && t->right != NULL)
8         ++c;
9     traverseCount(t->left);
10    traverseCount(t->right);
11    return c;
12 }
```

Ή (καλύτερα)

```
1 int traverseCount(tree t)
2 {
3     if(t == NULL)
4         return 0;
5     if(t->left != NULL && t->right != NULL)
6         return 1 + traverseCount(t->left) + traverseCount(t->right);
7     else
8         return traverseCount(t->left) + traverseCount(t->right);
9 }
```

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

1.6 Θέμα 6 ° - Ταξινόμηση

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]

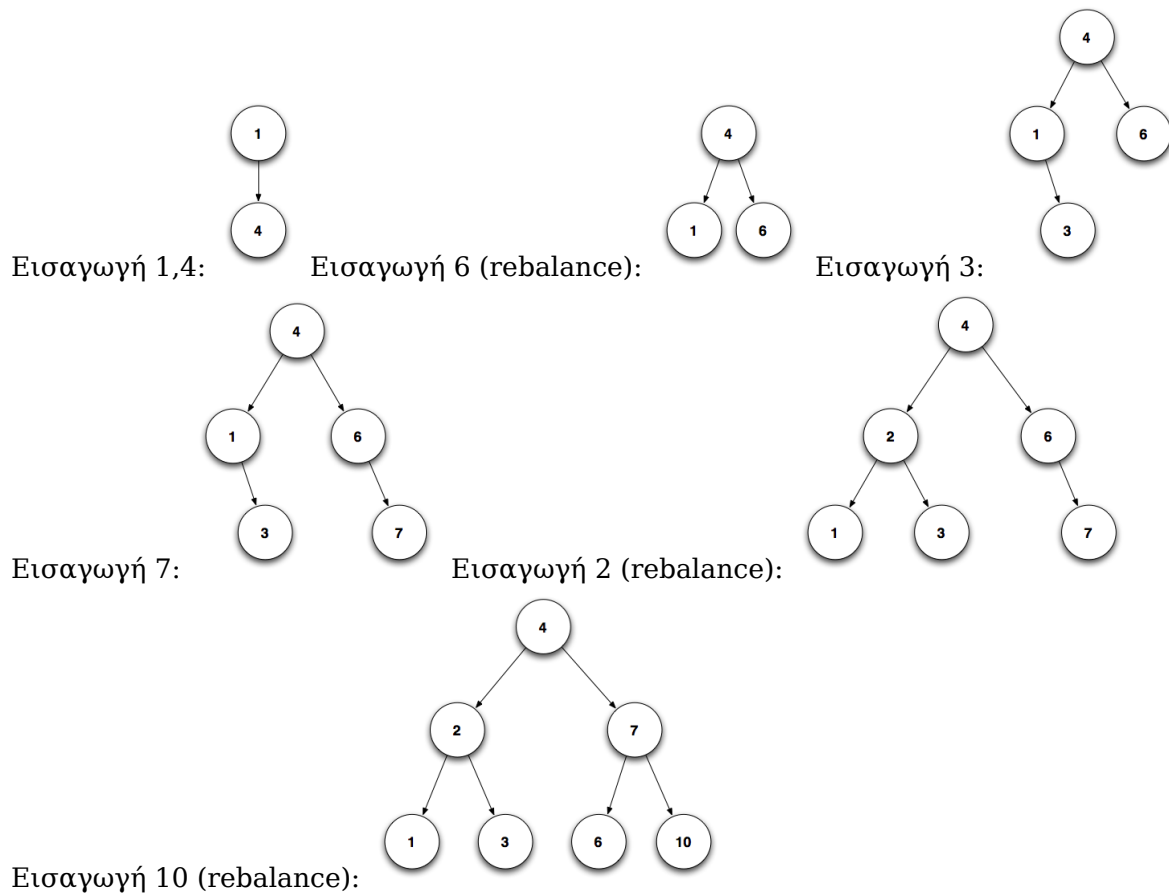
j = 3 : [1,1,2,4,3,5,6,8]

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 ° - AVL Δέντρα



2 Ιούλιος 2011

2.1 Θέμα 1 ° - Πολυπλοκότητα

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:

```
1 int K = 0;
2 for (i = 0; i < n; i++)
3     for (int j = 0; j < n; ++j) // "j<m" typo
4         f(n); // f(n) Complexity: O(n)
```

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

2.2 Θέμα 2 ° - Πίνακες

```
1 #define ARR_SIZE 100
2
3 int evenElementsSum (int* array)
4 {
5     int sum = 0;
6     int i;
7
8     for (i = 0; i < ARR_SIZE; i += 2)
9         sum += *(array + i);
10    return sum;
11 }
```

2.3 Θέμα 3 ° - Λίστες

Λύση γραμμικού χρόνου:

```
1 struct node* removeNodes(struct node* list, int num)
2 {
3     struct node *ptr = list
4     struct node *temp;
5     int count = 0;
6     /* Count the nodes */
7     while (ptr != NULL) {
8         ptr = ptr->next;
9         ++count;
10    }
11    ptr = list;
12    --count; //needed to stop at the node before the ones to be deleted
13    /* Find desired node */
14    while (count != num) {
15        ptr = ptr->next;
16        --count;
17    }
18    /*Make new last node point to NULL */
19    temp = ptr->next;
20    ptr->next = NULL
21    ptr = temp;
22    /* Delete all nodes */
23    while (ptr != NULL) {
24        temp = ptr->next;
25        free(ptr);
26        ptr = temp;
27    }
```

```
28  /* Return the list after deleting nodes */
29  return list;
```

2.4 Θέμα 4 ° - Δέντρα

```
1  int treeAdder(tree t, int min, int max)
2  {
3      if(t == NULL)
4          return 0;
5      if((t->value < min) || (t->value > max))
6          return t->value + treeAdder(t->left) + treeAdder(t->right);
7      else
8          return treeAdder(t->left) + treeAdder(t->right);
9  }
```