

# Data Structures and Algorithms

## Exercise 2

Markus Innerebner, Romans Kasperovics  
dsa(a)inf.unibz.it

Free University of Bozen - Bolzano, Italy

Thursday, March 26rd, 2009

a)  $T(n) = 15T(n/4) + n^2$

## Solution using Master method

$$n^{\log_b a} = n^{\log_4 15} \approx n^{1.95} \implies \text{case 3}$$

$$f(n) = n^2 \in \Theta(n^{1.95+0.05}) \implies T(n) \in \Theta(n^2)$$

$$b) T(n) = bT(n-1) - (b-1)$$

### Solution using Substitution method

$$\begin{aligned}T(n) &= bT(n-1) - b + 1 = \\&= b(bT(n-2) - b + 1) - b + 1 = b^2T(n-2) - b^2 + 1 \\&= b^2(bT(n-3) - b + 1) - b^2 + 1 = b^3T(n-3) - b^3 + 1\end{aligned}$$

$$T(n) = b^i T(n-i) - b^i + 1$$

Upper bound of  $i = n - 1$ . We assume  $T(1) = d + 1$ , then

$$T(n) = b^{n-1}(d+1) - b^{n-1} + 1 = b^{n-1}d + 1 = O(b^n)$$

Our guess:  $T(n) \leq cb^n$

$$bT(n-1) - b + 1 \leq bcb^{n-1} - b + 1 \stackrel{?}{\leq} cb^n$$

$$-b + 1 \stackrel{?}{\leq} 0, \text{ holds for any } c, b \geq 1$$

c)  $T(n) = 6T(n/2) + n^2$

### Solution using Master method

$$n^{\log_b a} = n^{\log_2 6} \approx n^{2.585} \implies \text{case 1}$$

$$f(n) = n^2 \in \Theta(n^{2.585-0.585}) \implies T(n) \in \Theta(n^{2.585})$$

$$d) T(n) = T(n-1) + \lg n$$

### Solution using Substitution method

$$\begin{aligned} T(n) &= T(n-1) + \lg n = T(n-2) + \lg(n-1) + \lg n \\ &= T(n-3) + \lg(n-2) + \lg(n-1) + \lg n \end{aligned}$$

$$T(n) = \lg(n!)$$

Our guess:  $T(n) = \Theta(n \lg n)$

## Upper bound

$\exists c, n_0 > 0 : \forall n \geq n_0 : T(n) \leq cn \lg n$

$$T(n) = T(n-1) + \lg n \leq c(n-1) \lg(n-1) + \lg n \stackrel{?}{\leq} cn \lg n$$

$$c(n-1) \lg(n-1) + \lg n = cn \lg(n-1) - c \lg(n-1) + \lg n$$

- For  $n \geq 2 : \lg(n-1) \geq \lg(\frac{n}{2})$ , from this follows

$$cn \lg(n-1) - c \lg(n-1) + \lg n \leq cn \lg(n-1) - c \lg \frac{n}{2} + \lg n$$

$$cn \lg(n-1) - c \lg \frac{n}{2} + \lg n = cn \lg(n-1) - c \lg n + c \lg 2 + \lg n = cn \lg(n-1) - c \lg n + c + \lg n$$

- $\lg(n-1) \leq \lg n$ , from this follows

$$cn \lg(n-1) - c \lg n + c + \lg n \leq cn \lg n - c \lg n + c + \lg n$$

- Now,  $cn \lg n - c \lg n + c + \lg n \leq cn \lg n$  iff  $-c \lg n + c + \lg n \leq 0$

$$c \leq (c-1) \lg n, \frac{c}{c-1} \leq \lg n$$

holds for  $c = 2$  and  $n \geq 4$

### Lower bound

$\exists c, d, n_0 > 0 : \forall n \geq n_0 : T(n) \geq cn \lg n + dn$

$$T(n) = T(n-1) + \lg n \geq c(n-1) \lg(n-1) + d(n-1) + \lg n = cn \lg(n-1) - c \lg n - 1 + dn - d + \lg n \stackrel{?}{\geq} cn \lg n + dn$$

- For  $n \geq 2 : \lg(n-1) \geq \lg(\frac{n}{2})$ , from this follows

$$cn \lg(n-1) - c \lg n - 1 + dn - d + \lg n \geq cn \lg \frac{n}{2} - c \lg(n-1) + dn - d + \lg n$$

$$= cn \lg n - cn \lg 2 - c \lg(n-1) + dn - d + \lg n = cn \lg n - cn - c \lg(n-1) + dn - d + \lg n \geq cn \lg n$$

- Now:  $cn \lg n - cn - c \lg(n-1) + dn - d + \lg n \geq cn \lg n$  iff  $cn - c \lg(n-1) + dn - d + \lg n \geq 0$

- $\lg n > \lg(n-1)$ , from this follows:

$$-cn - c \lg(n-1) + dn - d + \lg n > -cn - c \lg(n-1) + dn - d + \lg(n-1)$$

$$\text{find condition when: } -cn - c \lg(n-1) + dn - d + \lg(n-1) \geq 0$$

$$(d-c)n \geq (c-1) \lg(n-1) + d$$

holds for  $c = 1, d = 2$  and  $n \geq 2$

e)  $T(n) = 9^c T(n/3) + n^{2c}$

### Solution using Master method

$$n^{\log_b a} = n^{\log_3 9^c} = n^{2c} \implies \text{case 2}$$

$$f(n) = n^{2c} \in \Theta(n^{2c}) \implies T(n) \in \Theta(n^{2c} \lg n)$$

Write a C program that uses the Hoare partitioning setting the pivot element to the most left side.

```
int main() {
    char A[] = "QUICKSORT";
    quicksort(A,0,strlen(A)-1);
    return 0;
}

void quicksort(char* A, int l, int r) {
    if(l<r) {
        printf("Quicksort(%d,%d)",l,r);
        printf("%s\n",A);
        int m = partition(A,l,r); printf("%s m= %d \n", A,m);
        quicksort(A,l,m); quicksort(A,m+1,r);
    }
}
```

```
#define swap(x, y) {char tmp = x; x = y; y = tmp; }
```

```
typedef enum { FALSE, TRUE } bool;
```

```
int partition(char A[], int l, int r) {  
    char x = A[l];  
    int i = l-1; int j = r+1;  
    while(TRUE){  
        do j=j-1;  
        while(A[j]>x);  
        do i=i+1;  
        while(A[i]<x);  
        if(i<j) swap(A[i],A[j]);  
        else return j;  
    }  
}
```

Array after each call of PARTITION :

m	Q	U	I	C	K	S	O	R	T
3	O	K	I	C	U	S	Q	R	T
2	C	K	I	O	U	S	Q	R	T
0	C	K	I	O	U	S	Q	R	T
1	C	I	K	O	U	S	Q	R	T
7	C	I	K	O	T	S	Q	R	U
6	C	I	K	O	R	S	Q	T	U
4	C	I	K	O	Q	S	R	T	U
5	C	I	K	O	Q	R	S	T	U

### Call stack

Quicksort (0, 8)	QUICKSORT
Quicksort (0, 3)	OKICUSQRT
Quicksort (0, 2)	CKIOUSQRT
Quicksort (0, 0)	CKIOUSQRT
Quicksort (1, 2)	CKIOUSQRT
Quicksort (1, 1)	CIKOUSQRT
Quicksort (2, 2)	CIKOUSQRT
Quicksort (3, 3)	CIKOUSQRT
Quicksort (4, 8)	CIKOUSQRT
Quicksort (4, 7)	CIKOTSQRU
Quicksort (4, 6)	CIKORSQTU
Quicksort (4, 4)	CIKOQSRTU
Quicksort (5, 6)	CIKOQSRTU
Quicksort (5, 5)	CIKOQRSTU
Quicksort (6, 6)	CIKOQRSTU
Quicksort (7, 7)	CIKOQRSTU
Quicksort (8, 8)	CIKOQRSTU

Runtime analysis:

### Asymptotic Complexity

$$T_{\text{worst}}(n) = \Theta(n^2)$$

$$T_{\text{best}}(n) = \Theta(n \log n)$$

Running time depends on the distribution of splits

### Special cases

Case	Runtime	Example
Worst	$O(n^2)$	(1, 2, 3, 4, 5, 6, 7, 8)
Best	$O(n * \log n)$	(5, 7, 1, 4, 2, 8, 3, 6) pivot element splits subarray in the m

Modification in non increasing order:

```
int partition(char A[], int l, int r) {
    char x = A[l];
    int i = l-1; int j = r+1;
    while(TRUE){
        do {j=j-1;}
        while(A[j]<x);
        do {i=i+1;}
        while(A[i]>x);
        if(i<j) {swap(A[i],A[j]);}
        else {return j;}
    }
}
```