CS 435 HW1 - Anosh Abraham

1) $T(n) = 2n^3 - 10n^2 + 100n - 50$

(a) Prove $T(n)$ is $O(n^3)$ $\Rightarrow$ $\leq 2n^3 + 100n$, $n \geq 0$ (drop negatives)

$\leq n^3(2 + 100/n^2)$,

$\leq n^3(2 + 100/100^2)$, $n \geq 100$

$\leq (2.01)n^3$, $n \geq 100$ ✓

Prove

(b) $T(n)$ is $\Omega(n^3)$ $\Rightarrow$ $\geq 2n^3 - 10n^2 - 50$, $n \geq 0$ (drop positive)

$\geq n^3(2 - 10/n - 50/n^3)$,

$\geq n^3(2 - 10/100 - 50/100^3)$, $n \geq 100$

$\geq n^3(2 - .1 - .00005)$,

$\geq (1.89995)n^3$ $n \geq 100$ ✓

2) (a) $n = 1, 2, 4, 8, 16, 32, 64$;

Growth rates →	$\log n$	n	$n \log n$	n^2	n^3	2^n
$n = 1$	0	1	0	1	1	2
$= 2$	.301	2	.602	4	8	4
$= 4$	.602	4	2.408	16	64	16
$= 8$	.903	8	6.324	64	512	256
$= 16$	1.204	16	25.324	256	4096	65536
$= 32$	1.505	32	80.32	1024	32768	4294967296
$= 64$	1.807	64	258.042	4096	2097152	18446744073709551616

(b) $n^2 \log n$, 5, $n \log^2 n$, 2^n, n^2, n, $\sqrt{n}$, $\log n$, $\frac{n}{\log n}$

small → large: $5 < \log n < \sqrt{n} < \frac{n}{\log n} < n < n \log^2 n < n^2 < n^2 \log n < 2^n$

3) when $k=1 \rightarrow X = X+1$ executes $n-1$ times

$k=2 \rightarrow$ " " $n-2$ times

⋮

$k=n-2$ " " $n-(n-2) = 2$ times

$k=n-1$ " " $n-(n-1) = 1$ times

$k=n$ " " 0 times

where total executions equals $X = \frac{n(n-1)}{2}$ of $\Theta(f(n)) = \Theta(n^2)$

(n=m) m' k
n=16 → 16 0
8 1
4 2
2 3
1 4

4) (a) →

(b) $m = n/2^k$ ⇒ Initial start where $m=n$, $k=0$ so $m=n/2^k$

Suppose @ start of any iteration, $m=n/2^k$ holds true. At end both sides are divided by 2, thus relation holds still. Now let m' & k' be of m & k at the end of the iteration. Then, $m'=m/2$ & $k'=k+1$.

SO → $m'=m/2 = n/2^{k+1} = n/2^{k'}$ ⇒ relation holds.

(c) After last iteration, $m=1=n/2^k$. Therefore, $n=2^k$, meaning $k = \log_2 n$. ✓

5) (a)

```
int sum(int A[], int n){
    if (n==1) return A[0];
    T = sum(A, n-1);
    return (T + A[n-1]);
}
```

(b) recurrence equation:

$$f(n) = \begin{cases} 0, & n=1 \\ f(n-1)+1, & n>1 \end{cases}$$

(c) Of base, $n=1$, $f(1)=0$ & $f(n)=n-1=1-1=0$ ✓

for any $n \geq 2$, suppose $f(n-1)=n-2$. Then,

$f(n) = f(n-1)+1 = n-2+1 = n-1$ ✓

6) (a)

```
int MAX(int A[], int n){
    if (n==1) return A[0];
    M = MAX(A, n-1);
    if (A[n-1] > M) M = A[n-1];
    return M;
}
```

(b) recurrence equation

$$f(n) = \begin{cases} 0, & n=1 \\ f(n-1)+1, & n>1 \end{cases}$$

(c) Of base, $n=1$, $f(1)=0$ & $f(n)=n-1=1-1=0$ ✓

for any $n \geq 2$, suppose $f(n-1)=n-2$. Then,

$f(n) = f(n-1)+1 = n-2+1 = n-1$ ✓

7) (a)

[5 4] 3 2 1
~~4 [5 3] 2 1~~
[4 3] 5 2 1
3 4 [5 2] 1
3 [4 2] 5 1
[3 2] 4 5 1
2 3 4 [5 1]
2 3 [4 1] 5
2 [3 1] 4 5
[2 1] 3 4 5
1 2 3 4 5

Swaps for worst case n=5 is 10.

(b)

```
void ISORT(dtype A[], int n){
    if(n==1) return;
    ISORT(A, n-1);

    j = n-1;
    while (j>0 and A[j] < A[j-1]){
        swap(A[j], A[j-1]);
        j = j-1
    }
}
```

7) ⓒ recurrence relation

$$f(n) = \begin{cases} 0, & n=1 \\ f(n-1)+n-1, & n>1 \end{cases}$$

ⓓ repeated substitution

$$\hookrightarrow f(n) = n-1+f(n-1)$$
$$= (n-1)+(n-2)+f(n-2)$$
$$= \text{"} + \text{"} + (n-3)+f(n-3)$$
$$\vdots$$
$$= (n-1)+(n-2)+\ldots+1+f(1)$$
$$= \text{"} + \text{"} + \ldots + 1$$
$$= ((n-1)(n))/2 \Rightarrow \frac{n^2-n}{2}$$

8) ⓐ complexity of $T(n) \Rightarrow$ outer loop runs to $(n-1)$ $\quad 1+2+3+\ldots(n-1) = \frac{n(n-1)}{2}$

$\Rightarrow$ inner loop " to $(n-1)$

$T(n) = \frac{n^2-n}{2}$ $\quad T(n) = O(n^2)$

ⓑ
```
void bubble(dtype A[], int n){
 if(n==1) return;
 for j=0 to n-2{
   if(A[j] > A[j+1])
     swap(A[j], A[j+1]); }
 bubble(A, n-1);
}
```

ⓒ $f(n) = \begin{cases} 0, & n=1 \\ n-1+f(n-1), & n>1 \end{cases}$

$$f(n) = n-1+f(n-1)$$
$$= (\text{"}) + (n-2)+f(n-2)$$
$$= (\text{"}) + (\text{"}) + (n-3)+f(n-3)$$
$$\vdots$$
$$= (\text{"}) + (\text{"}) + \ldots + 1 + f(1)$$
$$= (\text{"}) + (\text{"}) + \ldots + 1$$
$$= ((n-1)(n))/2 \Rightarrow \frac{n^2-n}{2}$$

9) ⓐ recurr. relation

$f(n)=\begin{cases}0, & n=1\\ 2f(n/2)+1, & n>1\end{cases}$ $\rightarrow$ $f(n) = 1+2f(n/2) = 1+2[1+2f(n/4)]$

$= 1+2+4f(n/4)$

$= 1+2+4+8f(n/8)$ $n=2^k$

$= " + " + " + \dots + 2^{k-1} + f(n/2^k)$,

$= 1+2+4+\dots+ " + f(1)$

$= " + " + " + \dots + "$

$= 2^k - 1 \Rightarrow n-1$

rep. sub (i)

(ii) Of base, $n=1$, $f(1) = 1 = A+B$

For any $n>1$, supposing $f(n/2) = A\cdot n/2 + B$,

prove $f(n) = An+B$

$\hookrightarrow f(n) = 2f(n/2)+1$ [from recurrence]

$= 2(An/2+B)+1$ [hypothesis]

$= An+2B+1$

$= An+B$ [for induction]

for the other equality holds iff: $2B+1=B$

$\hookrightarrow$ now 2 relations $\Rightarrow$ $\left.\begin{array}{l} A+B=1 \text{ (base case)}\\ 2B+1=B \text{ (induction step)}\end{array}\right\}$ determines unknowns: $A=1$, $B=-1$. Thus, $f(n)=n-1$

ⓑ recurr. rel.

$f(n)=\begin{cases}0, & n=1\\ f(\lfloor n/2\rfloor)+f(\lceil n/2\rceil)+1, & n>1\end{cases}$

$\hookrightarrow$ Of base, $n=1$, $f(1)=0=n-1$, ~~base~~ case ✓

For any $n>1$, suppose $f(m)=m-1$, $\forall m<n$ $\Rightarrow$ $f(n) = f(\lfloor n/2\rfloor)+f(\lceil n/2\rceil)+1$ recurr. re

$= (\lfloor n/2\rfloor - 1) + (\lceil n/2\rceil - 1) + 1$ hyp

$= \lfloor n/2\rfloor + \lceil n/2\rceil - 1$

$= n-1$ ✓

10) ⓐ if all elements in first half equal (T1), and all elements in second half equal (T2), & element of first half equal to element of second half, then all elements are equal and algo returns true.

ⓑ $n=1$, there's a single element that equals itself, thus algo correctly returns true val

W/ induction step, prove algo works for any $n\geq 2$, supposing it's valid for $n/2$. Through hypoth, the first recursive call returns true in T1 iff all elements of first half are equal. And second recursive call returns true in T2 iff all elements in second half are equal. And T3 is true iff elements of first half equals an element of second half. Thus, algo correctly returns true if all elements are equal.

10) c) $f(n) = \begin{cases} 0 & n=1 \\ 2f(n/2)+1, & n>1 \end{cases}$

d) rep. sub.

$\hookrightarrow f(n) = 1+2f(n/2)$

$= 1+2[1+2f(n/4)]$

$= 1+2+4f(n/4)$

$= " + " + 4 + 8f(n/8)$

$= " + " + " + \dots + 2^{k-1} + f(n/2^k)$, where $n=2^k$

$= " + " + " + \dots + " + f(1)$

$= 1+2+4+\dots+2^{k-1} = 2^k - 1 = n-1$ ✓