HW2 - key

4.2.1

b) $x(n) = 3x(n-1)$ for $n>1$ $x(1)=4$

$$x(n) = 3x(n-1)$$
$$= 3.[3x(n-1-1)] = 3^2 x(n-2)$$
$$= 3^2.[3x(n-2-1)] = 3^3 x(n-3)$$
$$\vdots$$
$$= 3^i x(n-i)$$
$$\vdots$$
$$= 3^{n-1} x(1) = 4.\ 3^{n-1}$$

c) $x(n) = x(n-1) + n$ for $n>0$ $x(0)=0$

$$x(n) = x(n-1)+n$$
$$= [x(n-2)+(n-1)]+n = x.(n-2)+(n-1)+n$$
$$= [x(n-3)+n-2]+n-1+n = x.(n-3)+(n-2)+(n-1)+n$$
$$\vdots$$
$$= x.(n-i)+(n-i+1)+(n-i-2)+\ldots+n$$
$$\vdots$$
$$= x(0)+1+2+3+\ldots+n = \frac{n.(n+1)}{2}$$

d) $x(n) = x(n/2)+n$ for $n>1$ $x(1)=1$ solve for $n=2^k$

$$x(2^k) = x.(2^{k-1})+2^k$$
$$= [x(2^{k-2})+2^{k-1}]+2^k = 2^{k-1}+2^k$$
$$= [x(2^{k-3})+2^{k-2}]+2^{k-1}+2^k = x.(2^{k-3})+2^{k-2}+2^{k-1}+2^k$$
$$\vdots$$
$$= x.(2^{k-i})+2^{k-i+1}+2^{k-i+2}\ldots 2^k$$
$$= x(2^{k-k})+2^1+2^2+2^3\ldots 2^k = 2^{k+1}-1 = 2.2^k-1 = 2n-1$$

2.5.3 Find smallest value of n : $F(n) > 2^{31}-1$

a.) $F(n) = \frac{1}{\sqrt{5}} \phi^n$ // round to nearest integer we get:

$$\frac{1}{\sqrt{5}} \phi^n > 2^{31}-1 \equiv \phi^n > \sqrt{5}(2^{31}-1) \quad / \ln$$

$$\ln \phi^n > \ln(\sqrt{5}(2^{31}-1))$$

$$n > \frac{\ln(\sqrt{5}(2^{31}-1))}{\ln \phi} \approx 46.3 \doteq \underline{47}$$

b.) $F(n) > 2^{63}-1$

$$\frac{1}{\sqrt{5}} \phi^n > 2^{63}-1 \equiv \phi^n > \sqrt{5}(2^{63}-1) \overset{\ln}{\equiv} n > \frac{\ln(\sqrt{5}(2^{63}-1))}{\ln \phi} \equiv$$

$$\equiv n > 92.4 \doteq \underline{93}$$

3.1.4

a.) Algorithm Brute Force Alg Eval ($P[0..n], x$)

```
p ← 0
for i ← n to 0 step -1 do
    power ← 1
    for j ← 1 to i do
        power ← power * x
    p ← p + P[i] * power
return p
```

} number of multiplications - most frequent op.

$$M(n) = \sum_{i=0}^{n} \sum_{j=1}^{i} 1 = \sum_{i=0}^{n} i = \frac{n(n+1)}{2} \in O(n^2)$$

b.)

```
p ← P[0]
power ← 1
for i ← 1 to n do
    power ← power * x
    p ← p + P[i] * power
return p
```

} $m_{fo} = M(n) = \sum_{i=1}^{n} 2 = 2n$

3.1.6

not stable

3.1.7

Yes. Both ops. (find smallest element and swap) can be done as efficiently

3.1.10

Yes stable. Swap makes it stable w/ only constant space required for extra variable.

3.1.11

Algorithm DSort (A[1...2n]

for (i = 1 to n step 2)

swap (A[i], A(2n-i+1))

This algorithm will do n/2 swaps thus $\Theta(n)$

3.2.4

43 comparisons

for i=0 To n-1

0 to 47 - 5 = 43

3.2.9 (a)

Algorithm CountSubstringsAB(A[0...n-1])

count ← 0

for i=0 to n-2

if A[i] = 'A'

for (j = i+1 to n-1

if A[j] = 'B' $\Theta(n^2)$

count ← count + 1

return count

3.3.2

a.) - find average distance
- place the post office to closest city to average value

3.1

place the post office to a city closest to $\frac{c_1 + c_n}{2}$

8

3.4.8

generate all permutations of elements. For each permutation check if it is ordered according to requirement (sequential comparison of consecutive elements). Stop when first permutation succeed your check, o.w. check next permutation

$$T(n) = O(n! \cdot (n-1)) = O((n+1)!)$$