

Seonhye Yang (sy3420) Homework 3

Problem 1

- Given input array,  $A = [1, 2 \dots n]$  ( $\text{len}(A) = n$ ) and positive integer  $k = 1 \leq k \leq n$ .
- Algorithm must find 1<sup>st</sup> smallest, 2<sup>nd</sup> smallest... and  $k^{\text{th}}$  smallest element.

Select( $A[1 \dots n]$ ,  $k$ ):

choose pivot  $i$  from  $1 \dots n$

$j := \text{Partition}(A[1 \dots n], \text{pivot} = A[i])$

if  $k = A[i]$ :

return pivot

else:

return select( $j$ ,  $k$ )

---

### Problem 2

Master method only works for the following type of recurrences.

$$T(n) = aT(n/b) + f(n) \text{ where } a \geq 1 \text{ and } b > 1$$

- a. If  $f(n) = O(n^c)$  where  $c < \log_b a$ , then  $T(n) = \Theta(n^{\log_b a})$
- b. If  $f(n) = \Theta(n^c)$  where  $c = \log_b a$ , then  $T(n) = \Theta(n^c \log(n))$
- c. If  $f(n) = \Omega(n^c)$  where  $c > \log_b a$ , then  $T(n) = \Theta(f(n))$

a.  $T(n) = 4T(n/2) + n^2$      $a = 4$  ,  $b = 2$  ,  $c = 2$

We have  $a = 4$ ,  $b = 2$ , and  $f(n) = n^2$

- $n^{\log_b a}$  is  $n^2$  and  $f(n)$  is therefore  $\Theta(n^2)$ .
- So case b is applied here.

Thus,  $T(n) = 4T(n/2) + n^2$  is  $\Theta(n^2 \log(n))$

b.  $2T(n/2) + \log(n)$

$a = 2$  ,  $b = 2$  ,  $f(n) = \log(n)$

Although  $a$  and  $b$  are satisfied,  $f(n)$  must be a polynomial to use the masters theorem.

c. First, we know that c,  $T(n) = 0.2T(n/2) + n$  will not work with master's theorem because  $a = 0.2$  and according to the theorem,  $a \geq 1$ .

d.  $T(n) = 8T(n/2) + 2^n$

$a = 8$  ,  $b = 2$  ,  $f(n) = 2^n$

Although  $a$  and  $b$  are satisfied,  $f(n)$  is an exponential function.  $f(n)$  must be a polynomial for the masters theorem to work.

### Problem 3

Fibonacci Sequence Formula.

•  $F_n = F_{n-1} + F_{n-2}$ , where  $n > 1$

```
Fib(n)
    memo[1...n] = [1, ..., 1]  n times base case
    for i=3 to n
        memo[i] = memo[i-1] + memo[i-2]
    return memo[n]
```

} Given to use during lecture.

$TC = \Theta(n)$

$SC = \Theta(n)$  space needed for memo[1...n]

Modified Fibonacci Sequence:

Def BottomUpFib(n):

if  $n == 0$  or  $n == 1$ :  
return n

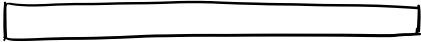
# lets define some variables here


A = 0  
B = 1

```
for i=1 in n:
    fib = A + B
    A = B
    B = fib
return B
```

In this bottom-up approach, we solved the subproblem only once. Therefore, the time complexity is  $O(n)$ . We used 2 variables to track our intermediate results, our space complexity is  $O(1)$ .

### Problem 4

a. Rod   $\rightarrow$   $n$ -inches

P   $\rightarrow$   $P[1 \dots n]$  Price

$P[i]$  = price of  $i$  inches piece of rod.

Two arrays to this problem:

① cut the rods into length of  $\text{length}[i]$ . Then add the profits of \$1 and check for the  $\text{length}[i]$ .

② Don't cut rod into  $\text{length}[i]$  and into  $i+1$ .

$$\text{MaxPrice}(n) = \max(P[1] + \text{MaxPrice}(n-1) + P[2] + \text{MaxPrice}(n-2) + P[3] + \text{MaxPrice}(n-3) \dots P[n] + \text{MaxPrice}(n)).$$

This is assuming we cut the rod of length  $n$  into  $i$  inches. Then we add the max price of the sizes of rod  $n-i$ .

b. The base case is when it's 0,  $\text{MaxPrice}[0]$ . This is because length 0 of the rod doesn't have a price.