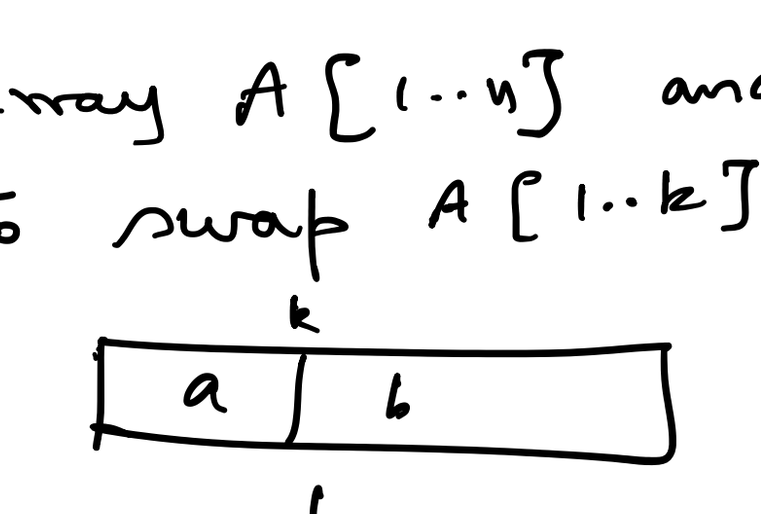
Assignment 1.

Thursday, 8 September 2022 6:58 PM

Given an array $A[1..n]$ and a k, $1 \le k \le n-1$, we need to swap $A[1..k]$ with $A[k+1..n]$.



Eg: $A \rightarrow$ a b c d e f g h i j k.

$A[1..11]$. $k = 3$.

After swapping

A = d e f g h i j k a b c

We outline two ideas. Design and analyse the Algorithms based on these ideas.

[a] if $k = n-k$, Swap $A[1..k]$ with $A[k+1..n]$ element by element & return.

if $k < (n-k)$, Swap $A[1..k]$ with $A[n-k+1..n]$. Recursively proceed with $A[1..n-k]$ with an appropriate k.

if $k > (n-k)$ Swap $A(n-k+1..n)$ with $A(1..k)$. Recursively proceed with $A[k+1..n]$ (with appropriate k).

(a) Prove the correctness.

(b) Write down a recursive pseudocode implementing the idea stated above.

(c) Count the number of swap operations carried out by the algorithm. From the way in which the size is reducing, you may guess that GCD $(k, n-k)$ must figure in some way in the complexity expression.
(Analyse the algorithm by setting up appropriate recurrence equation and solving the same).

The second idea is using the fact that

$$(XY)^R = Y^R X^R$$ where X^R is reverse of X.

(a) Write a (3 step) iterative algorithm based on the above idea.

(b) Analyse the complexity of your algorithm. (Count the number of swap operations).

(c) Which idea is better? Why?

---

(2) Complexity of Euclid Algorithms:

(a) Let $x = qy + r$. Show that complexity of finding q and r, given x and y is $\Theta(\log q \cdot \log y)$ bit operations.
(x, y, q, r are all positive integers.
$q = \lfloor x/y \rfloor = x \text{ Div } y$. $\{x, q, r \ge 0, y > 0\}$.
$r = x \bmod y$.)

(b) Consider the iterative GCD algorithm as specified below:

```
GCD (a, b)
   r0 = a;
   r1 = b;
   i = 1;
  while (ri > 0)
     i = i+1;
     ri = ri-2 mod ri-1
     qi-1 = ri-2 div ri-1.
  Return (ri-1).
```

(Note q_{i-1} is not needed for computation of GCD but needed only for extended Euclid Algorithm).

Assume that the While loop is executed l times ($l \ge 0$). Then $i = l+1$, and r_l will be the GCD. When $l \ge 1$, $q_1, q_2, \dots, q_l$ are all defined.

i) Show that $q_1 q_2 \dots q_l \le a$ (product of all q's is $\le a$).

ii) . Show that $q_2 q_3 \dots q_l \le b$.

iii) . Prove that the bit complexity of the GCD algorithm is $\Theta(\log a \cdot \log b)$.
(We may define $len(a)$ = No of bits in the binary representation of a. ($a \ge 0$, int).
Then, $len(a) = 1 + \lfloor \log_2 a \rfloor$, $a > 0$
$= 1$ $a = 0$.)
The complexity is $\Theta(len(a) \cdot len(b))$

---

(3) Binary GCD.

Consider the following version of Binary GCD.

```
B-GCD (a, b).
    r = a;
    r' = b;
    e = 0;
    while (2|r) and (2|r')
        r = r/2 ; r' = r'/2 ; e = e+1;
    Repeat
        while (2|r)
            r = r/2
        while (2|r')
            r' = r'/2 .
        if (r' < r) then
            (r, r') = (r', r)
        r' = r' - r
    until (r' = 0).
    d = 2^e . r.
    Return d.
```

(a) Prove the correctness.

(b) Prove that the complexity is $\Theta(l^2)$ where $l = \text{Max}\{len(a), len(b)\}$.

(c) Extend this algorithm to compute $x, y \ni ax + by = d$.

Note: For 1 (b) "memory efficient" version may be written as follows..

```
GCD (a, b)
   r = a ; r' = b ;


   while (r' > 0)
      q = ⌊r/r'⌋
      r'' = r mod r'
      r = r' ; r' = r''.
   Return r .
```

(This is a general remark)..

---

4. Consider the problem of computing $\lfloor n^{1/2} \rfloor$ for a given non-negative integer n

(a) Using binary search, give an algorithm for this problem that runs in time $O(\log(n))^3$. Your Algorithm should discover bits of $\lfloor n^{1/2} \rfloor$, one-at-a-time, from high to lower order bits.

(b) Refine your algorithm in part (a) so that it runs in time $O(\log n)^2$.

5. Given n points $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$

(a) Show that there is a unique polynomial of degree $(n-1)$, $P_n(x) = a_0 + a_1 x + \dots + a_{n-1} x^{n-1}$ passing through these n points ($n \ge 2$).

(b) Use Lagranges interpolation formula to construct the polynomial.
Compute the EXACT number of multiplications done in the algorithm. (Division is also treated as multiplication).

(c) Let $G_{j-1}(x)$ be the $(j-2)$ degree polynomial that interpolates (x_k, y_k), $1 \le k \le j-1$. Let
$D_{j-1}(x) = (x - x_1)(x - x_2) \dots (x - x_{j-1})$.
Define
$$G_j(x) = (y_j - G_{j-1}(x_j)) \frac{(D_{j-1}(x))}{D_{j-1}(x_j)} + G_{j-1}(x).$$
Show that $G_j(x)$ interpolates (x_k, y_k), $1 \le k \le j$.
Find the EXACT complexity of this incremental algorithm.
How does it compare with the complexity of (b)?

---

(6) i) Is $\lceil \log n \rceil!$ polynomially bounded?

ii) Is $\lceil \log \log n \rceil!$ polynomially bounded?

iii) If $f(n), g(n) \in O(n^k)$
does $f(n) + g(n) \in O(n^k)$
does $f(n) - g(n) \in O(n^k)$.

iv) if $f(n), g(n) \in \Theta(n^k)$
does $f(n) + g(n) \in \Theta(n^k)$?
$f(n) - g(n) \in \Theta(n^k)$?

v) if $f(n) \in \Theta(g(n))$
does $2^{f(n)} \in \Theta(2^{g(n)})$?
Does $2^{f(n)} \in O(2^{g(n)})$?
