Design and Analysis of Algorithms

Assignment-02

Dr. Ram P Rustagi
Sem IV (2020-Even)
Dept of CSE, KSIT
rprustagi@ksit.edu.in

Assignment Logistics

- Same group as that of HA01. To change, make a request.
- All assignments are to be submitted online in github.
  – Program (java/C/C++/python) should run on Linux.
- 1 day late submission: 20% penalty
- 2 days late submission: 50% penalty.
- Early submission may yield bonus marks
  - Early submission by 2 days: 20% bonus marks.
- Program should adhere to following
  - Use command line arguments for parameter passing
    - Avoid any hard coding of parameter values.
  - Program should not crash under any circumstances
  - It should have proper indentation for readability
  - Use proper variable names to indicate meaning
    - Avoid use of cryptic variable names e.g. `a,b,c,x,y`

Assignment Logistics

- There are total of 12 programming questions.
  - All Qs requires use of Divide/Conquer or Decrease/ Conquer apporach. (No brute force)
- Each group/team is assigned one of the questions and need to submit the same question. The assignment question number is same as that for HA01.
- A team is encouraged to do other non-assigned questions to help improve learning.
  - Team may be given bonus marks (as per the discretion of the instructor). The bonus marks, if given, may count towards previous/future assignments.
- Plagiarism (copy) will result in 0 marks for all
- Any partial code from net (googling) should be cited with URL along with explanation

Assignment Submission Details

- Each submission (on github) should include
  - The program name should preferably be as e.g.
    - Q01: `Asn02P01.java/c/cpp/py` …

      :
    - Q12: `Asn02P12.java/c/cpp/py` …
  - `Readme.txt`: should contain
    1. Team details (Names, USN)
    2. Contribution of each team member
    3. Instructions to run the program
    4. Challenges faced and how did you address these
    5. What did you learn from this assignment
  - `Output.txt`
    1. Output of program with your sample data
    2. Total number of basic (key) operations i.e. computation of time complexity.

Q01: Max Sum Subsequence

- Given an input sequence of numbers (both positive and negative), find the subsequence with maximum sum i.e. when values of the subsequence are added, the sum is maximum compared to any other subsequence. Note: The sequence of numbers will in a file, where filename is command line argument.
- For example, the sequence 2, -3, 1.5, -1, 3, -2, -3, 3
- The max sum subsequence is (1.5, -1, 3) = 3.5
- Hint: Look at max sub-sequence so far, and current sub-sequence

Max Sum Subsequence

arr=`[2, -3, 1.5, -1, 3, -2, -3, 3]`

`I1`: maxsubseq(arr) #8, last elem=`3`

  `I2`: maxsubseq(arr) #7, last elem=`-3`

    `I3`: maxsubseq(arr) #6, last elem=`-2`

      `I4`: maxsubseq(arr) #5, last elem=`3`

        `I5`: maxsubseq(arr) #4, last elem=`-1`

          `I6`: maxsubseq(arr) #3, last elem=`1.5`

            `I7`: maxsubseq(arr) #2, last elem=`-3`

              `I8`: maxsubseq(arr) #1, last elem=`2`

                last elem=`2`

                return max=`2`, suffix=`2`

Max Sum Subsequence

arr=[2, -3, 1.5, -1, 3, -2, -3, 3]

I7:last elem = -3

return max=2, suffix=0 (2-3=-1 is -ve)

I6:last elem=1.5

return max=2 (>0+0.5), suffix=1.5 = 0+1.5

I5:last elem=-1

return max=2 (>0.5+1), suffix=0.5 (1.5-1)

I4:last elem=3

return max=3.5 <0.5+3, suffix=3.5(=0.5+3)

I3:last elem=-2

return max=3.5 (>3.5-2), suffix=1.5 (3.5-2)

Max Sum Subsequence

arr= [2, -3, 1.5, -1, 3, -2, -3, 3]

I3:last elem=-2

return max=3.5 (>3.5-2), suffix=1.5 (3.5-2)

I2:last elem=-3

return max=3.5 (>1.5-3), suffix=0(1.5-3 is -ve)

I1:last elem=3

return max=3.5 (>0+3), suffix=3

answer 3.5

Q02: Left Rotation of String

- Given a string `S` of size `n`, and positive integer `m`, rotate left the string characters by `m`
- For example, if the string is "`abcdefghijkl`", and `m=5`, the output should be "`fghijklabcde`".
- Hint:
  - Consider the string as X_1X_2, where X_1 represents first `m` chracaters and X_2 remaining `n-m` characters.
  - Use the relation: $(X_1X_2)' = X_2'X_1'$
    - where X' implies reverse of X.
      - e.g. `(abcd)' = dcba`
  - Reverse(s) method should be implemented using decrease and conquer.

Reverse Method

- **str=**`abcde`

```
str=abcde
I1: abcde # size 5, [0..4]
  I2: bcd # size 3, [1..3]
    I3: c # size 1, [2..2] base case
    return c
  return dcb # swap 1st & last
return edcba # swap 1st & last
```

Q03: Hanoi Towers with 5 towers

- Implement Hanoi's tower using 5 towers to transfer `N` discs from tower A to tower B using tower C, D and E as temporary holding place.
  - Hint:
    - Move top `N/3` discs from A to C using E
    - Move middle `N/3` discks from A to D using E
    - Move top `N/3` discs from A to B using E
    - Move `N/3` discs from D to B using E
    - Move `N/3` discs from C to B using E
- Output: program should output status of each tower after each move
  - Compute time complexity and verifies it with various values of N e.g. N=6, 8, 9.

Q03: Hanoi

- Consider N=`7`
- `A=[D1,D2,D3,D4,D5,D6,D7]`
- `B=[], C=[], D=[], E=[]`
- `Algo`

Move top 3 (=Ceiling(7/3) from A to E using C

`A=[D4,D5,D6,D7], B=[], C=[], D=[]`

`E = [D1,D2,D3]`

Move Middle 2 (=Floor(7/3) from A to D using C

`A=[D6,D7], B=[], C=[],`

`D=[D4, D5]`

`E=[D1,D2,D3]`

Q03: Hanoi

So far:

`A=[D6,D7], B=[], C=[], D=[D4, D5]`

`E=[D1,D2,D3]`

Move Bottom 3 (=Floor(7/3) from A to B using C

`A=[], B=[D6,D7], C=[], D=[D4, D5]`

`E=[D1,D2,D3]`

Moves = 7 = $2^{\lceil n/3 \rceil}-1=7$

Move all (=2) from D to B using C

`A=[], B=[D4,D5,D6,D7], C=[], D=[]`

`E=[D1,D2,D3]`

Moves = 3 = $2^{\lfloor n/3 \rfloor}-1=3$

Q03: Hanoi

So far:

`A=[], B=[D4,D5,D6,D7], C=[], D=[]`

`E=[D1,D2,D3]`

Moves = 3 = $2^{\lfloor n/3 \rfloor}-1=3$

Move all (=3) from E to B using C

`A=[], B=[D1,D2,D3,D4,D5,D6,D7], C=[],`

`D=[], E=[]`

Moves = $7+3+3=10 = 3*2^{\lceil n/3 \rceil} = O(2^{\lceil n/3 \rceil})$

Moves with 3-tower Hanoi= $2^7-1=127$

Q04: CounterFeit Coin

- Detecting a counterfeit coin of a different weight using $\log_2 N$ comparisons.
  - You are given a bag with `N>0` coins and told that at most one of these coins is counterfeit.
  - Further, you are told that counterfeit coin can be either lighter or heavier than genuine ones.
  - Your task is to find if bag contains a counterfeit coin.
  - Available to you is a balance machine that compares the weights of two sets of coins and tells you if they are of equal weight of which set is lighter than the other
- logistics: Implement a library method which mimics balance and returns comparison result of two arrays.
  - -1 (less), 0 (Equal), +1 (greater than)
  - Assume compare method knows the counterfeit coin and hence can correctly provide the result.

A04: CounterFeit coin

- Consider N =50, coin=23, type =lighter

I1: Compare(1-25, 26-50) returns -1

I2: Compare(1-12, 13-24) returns +1

Since I1 gives -1, and I2 gives +1, implies all balls in 26-50 are identical and defective ball is lighter.

I3: Compare(1-6, 7-12) returns 0

Defective ball is between 13-24.

I4: Compare(13-18, 19-24) returns +1

Defective ball is between 19-24.

I5: Compare(19-21, 22-24) returns +1

Defective ball is between 22-24.

A04: CounterFeit coin

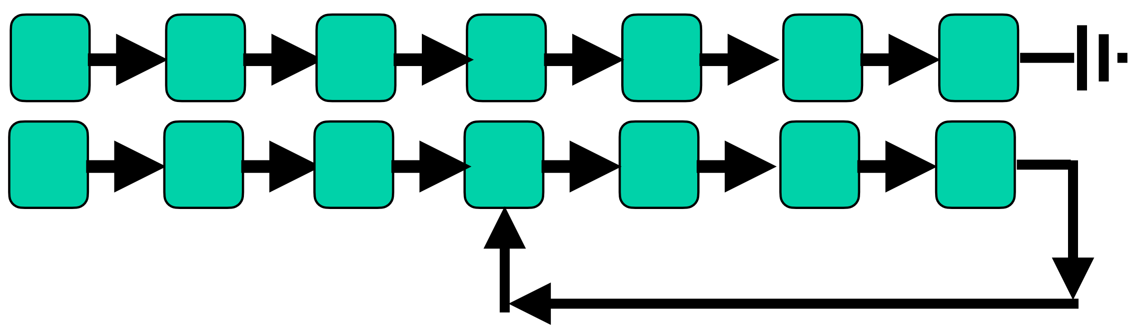
- Consider N =50, coin=23, type =lighter

`I6:` Compare(`22, 23`) returns `+1`

Defective ball is `23`

Q05: Traversing Circular Linked List

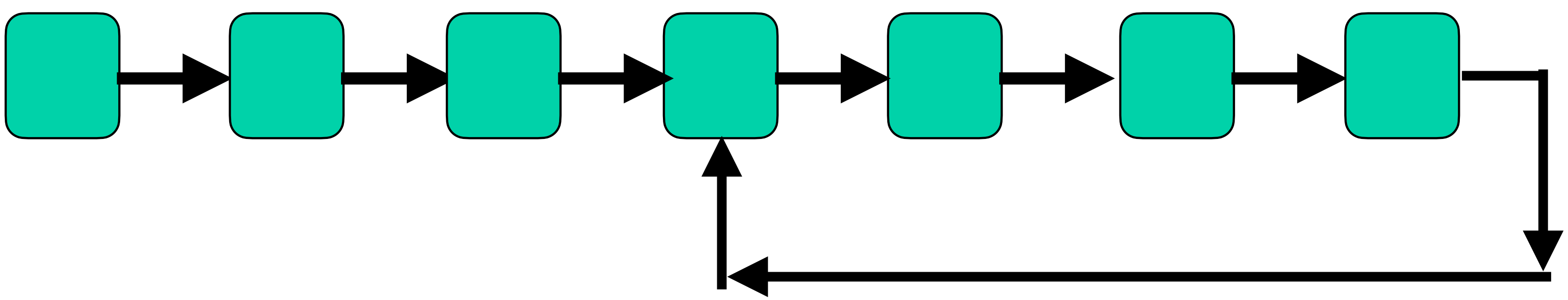
- Given a linked list (e.g. array of indexes), where each item points to next index. Some of the index may point to some earlier index. This may form a circular or a terminating linked lists as shown below.



- Implement a libary function that implements such a linked list of at least 10K elements and returns head of the list.
  - Use a random number to make it circular/terminating.
- Write a program to determine if the list (as returned by library function call) is circular or terminating in `O(n)` time.
  - Hint: consider two concurrent traversals one at double speed of of other (a variant of binary search or Divide)

A05

- Given head : head of the list
- Keep two pointers
  - P1=head; P2=head->next
  - both points to first element
- Run the following loop forever
  - Run following loop two times
    - if P2==P1 or P2==Null
      - it is circular link list, return
    - P2=P2->next
  - P1 = P1->next



Q06: Karatsuba Algorithm

- Given `2` decimal positive numbers, each consisting of `N` digits, perform multiplication of these two numbers using Karatsuba algorithm (i.e. divide by half and conquer)
  - Read the two decimal numbers from a file. The filename is to be taken as command line argument.

A06: Karatsuba

- Please go thru lecture L13.

Q07: Divide and Conquer

- Computation of Fibonacci number.
- Consider the following matrix multiplication

$$[F_{n-1} \; F_n] = [F_{n-2} \; F_{n-1}] * \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$$

- F_n is the 2^{nd} component of this matrix multiplication
- Taking the value of $F_0=0$, and $F_1=1$,
  - compute the value of F_n using Divide and Conquer approach in $O(\log_2 n)$ matrix multiplications.

Q07-Fibonacci

$[F_0\ F_1]=[0\ 1]$

$[F_1\ F_2]=[F_0\ F_1]*\begin{bmatrix}0 & 1\\ 1 & 1\end{bmatrix}=[0\ 1]*\begin{bmatrix}0 & 1\\ 1 & 1\end{bmatrix}=[1\ 1] \Rightarrow F_2=1$

$[F_2\ F_3]=[F_1\ F_2]*\begin{bmatrix}0 & 1\\ 1 & 1\end{bmatrix}=[1\ 1]*\begin{bmatrix}0 & 1\\ 1 & 1\end{bmatrix}=[1\ 2] \Rightarrow F_3=2$

Alternatively

$[F_2\ F_3]=[F_1\ F_2]*\begin{bmatrix}0 & 1\\ 1 & 1\end{bmatrix}=[F_0\ F_1]*\begin{bmatrix}0 & 1\\ 1 & 1\end{bmatrix}*\begin{bmatrix}0 & 1\\ 1 & 1\end{bmatrix}$

$=[F_0\ F_1]\begin{bmatrix}1 & 1\\ 1 & 1\end{bmatrix} = [0\ 1]\begin{bmatrix}1 & 1\\ 1 & 1\end{bmatrix} = [1\ 2] \Rightarrow F_3=2$

$[F_n\ F_{n+1}]=[F_{n-1}\ F_n]*\begin{bmatrix}0 & 1\\ 1 & 1\end{bmatrix}=[F_0\ F_1]*\begin{bmatrix}0 & 1\\ 1 & 1\end{bmatrix}*\begin{bmatrix}0 & 1\\ 1 & 1\end{bmatrix}*\dots*\begin{bmatrix}0 & 1\\ 1 & 1\end{bmatrix}$

$=[F_0\ F_1]*\begin{bmatrix}0 & 1\\ 1 & 1\end{bmatrix}^n$

Q07-Fibonacci

- Any computation of power(x,n) which computes x**n can be done in `log n` time
  - Note: x can be anything e.g. number, matrix etc.

```
def pow(x,n)
  if n=0
    return 1
  if n is even
    y = pow(x,n/2)
    return y * y
  else # n is odd
    y = pow(x, (n-1)/2)
    return x * y * y
```

Q07-Fibonacci

- Let M is matrix:

```
M=[0 1]
  [1 1]
```

- To find `Fn`
  - Compute `P=pow(M,n)`
    - can be done in log n time
  - Compute
    `Q=[0 1]*P`
- Element Q_{12} is `Fn`

Q08: Find Median

- Given an array of odd number of unsorted positive integers, find the median using divide and conquer without performing sort.
- For example,
  - if the given numbers are `3,10,2,9,5,7,4,6,8`
    - The median is 6.
- Hint: Choose a pivot, find the place of pivot in the array. If the pivot position is = `(N+1)/2`, then pivot is median
  - If pivot position is < `(N+1)/2`, then median is in right half
  - If pivot position is > `(N+1)/2`, then median is in left half

A08: Find Median

arr= `3,10,2,9,5,7,4,6,8`

Array size:9, median is 5th element in sorted array

`I1:` pivot=3 (=`A[1]`) #assuming index 1 to 9

partition using the pivot 3

arr= `[2],3,[10,9,5,7,4,6,8]`

pivot position=`2`, median in 2nd half, at 3rd (5-2) pos

`I2:` pivot=`10` (=`A[3]`) #assuming index 1 to 9

partition using the pivot 10

arr= `2,3,[8,9,5,7,4,6],10`

pivot position=`9`, median in 1st half, at 3rd (5-2) pos

A08: Find Median

arr= 2,3,[8,9,5,7,4,6],10

I3: pivot=8 (=A[3]) #assuming index 1 to 9

partition using the pivot 8

arr= 2,3,[4,6,5,7],8,[9],10

pivot position=7, median in 1st half, at 3rd (5-2) pos

I4: pivot=4 (=A[3]) #assuming index 1 to 9

partition using the pivot 4

arr= 2,3,4,[6,5,7],8,9,10

pivot position=3, median in 2nd half, at 2nd (5-3) pos

I5: pivot=6 (=A[4]) #assuming index 1 to 9

partition using the pivot 6

arr= 2,3,4,[5],6,[7],8,9,10

pivot position=5, the desired answer

Q09: Find Duplicate Numbers

- Given input file containing `N+1` numbers between `1` and `N` (both inclusive) in random order, find the repeating number. Each number is on one line. You can use only `O(N)` memory space. The finding of duplicate numbers should be done in `O(N)` time. You are not permitted to compute the sum of numbers and. The only operation allowed is compare.
- Hint:
  - Count the number of elements between `1` and `N/2`, and `N/2+1` and `N`. One of them will have `N/2+1` and other will have `N/2` count. The duplicate number will be in the partition having more numbers.

A09

arr= [8,10,19,5,17,20,4,14,18,2]

I1: 8 is sorted, Sorted arr = 8

arr= [10,19,5,17,20,4,14,18,2]

I2:10 Find pos using binary search

searches=1, sorted = 8,10

rem= [19,5,17,20,4,14,18,2]

I3:19, Find pos using binary search, # searches=2

sorted = 8,10,19, rem= [5,17,20,4,14,18,2]

I4:5 Find pos using binary search, # searches=2,

sorted = 5, 8,10,19, rem= [17,20,4,14,18,2]

I5:17 find pos using binary search, # searches=2,

sorted arr = 5, 8,10,17,19 arr= [20,4,14,18,2]

A09

`I5:17` is not in sorted order, Find pos using binary search

  \# searches=2, sorted arr = `5, 8, 10, 17, 19`

  arr= `[20, 4, 14, 18, 2]`

`I6:20` is in sorted order, Find pos using binary search

  \# searches=3, sorted arr = `5, 8, 10, 17, 19, 20`

  arr= `[4, 14, 18, 2]`

`I7:20` is in sorted order, Find pos using binary search

  \# searches=3, sorted arr = `5, 8, 10, 17, 19, 20`

  arr= `[4, 14, 18, 2]`

  :

  :

  :

Q10: Gifts/Box Matching Problem

- You are given a collection of N gifts of different sizes and N corresponding boxes. You are allowed to try a gift and box together, from which you can determine whether the gift is larger than the box, smaller than the box, or fits in the box exactly. However, there is no way to compare two gifts together or two boxes together for their sizes. The problem is to match each gift to its box. Design an algorithm for this problem with average-case efficiency in $O(n \log n)$
- Hint:
  - Take a box (as a pivot) to partition the gifts. Use the partition point of gifts to partition the box. Continue this way.

A10: Gifts/Box

G: 19,15,5,16,20,2,11,17,4,10,8,18,12,13,1,9,7,3,6,14

B: 5,20,13,6,16,4,2,3,9,15,18,11,1,10,14,8,7,19,12,17

`I1`: Take a random element of B as pivot for G, e.g. 8

partition G with this pivot.

Since each box has a match, this value will exist.

G is partitioned as

G_L=5,2,4,1,7,3,6; G_R=19,15,16,20,11,17,10,18,12,13,9,14

Box 8 is matched with Gift.

Partition the boxes with this pivot 8

B_L=5,6,4,2,3,1,7; B_R=20,13,16,9,15,18,11,10,14,19,12,17

B_L can only be matched G_L, and B_R can only be match G_R

`I2`: Solve the sub problem (B_L,G_L) and (B_R, G_R)

Q11: Alternating A/B Problem

- There are `2N` students (assume `N=2M` for some positive integer `M`), standing next to each other in a row, the first `N` of them from section `A`, while the remaining `N` students are from section `B` (. Make the students alternate in `ABAB...BABA...ABAB` pattern, where 1st part `ABAB...` is of size `M`, 2nd part of `BABA...` is of size `2M`, and 3rd part is size `M` similar to first part. This should be achieved in the minimum number of student moves. Use Decrease and conquer approach and using in-place movement i.e. use `O(1)` space
- Hint: Consider the smallest size problem you can solve in 1 move

A11: Alternating A & B

- Identify the reduction size and the smallest problem that can be solved brute force.
- Example: middle part
  - AB —> BA (swap the two)
  - AABB —>BABA (swap the first and last)
  - AA…BB—>BA…BA (swap 1^{st} A with last B)
- Example first and 3rd part
  - AABB —> ABAB (2nd and 3rd)
  - AA…BB —> AB…AB (Swap 2^{nd} A with 2^{nd} last B)

A11: Alternating A & B

- consider N=6, and the input=AAAAAABBBBBB
  - Split into 1st, 2nd and 3rd part as
    AAA AAABBB BBB
- First solve the middle (2nd) part: AAABBB
  - Reduce by 4:
    combine(AA+solve(AB)+BB)
    = combine(AA + BA + BB)
    = BABABA (swap 1st A with last B)
- Solve 1st and 3rd part: AAA (2nd part) BBB
  - Reduce by 4
    Combine(AA + solve(A (2nd part) B)+BB)
    = Combine(AA + (A (2nd part) B)+BB)
    =AB + (A (2nd part) B)+ AB (swap 2nd A with2nd last B)
    =ABA (2nd part)BAB = ABA BABABA BAB

A11: Alternating A & B

- Let N=8, and the input=AAAAAAAABBBBBBBB
  - Split into 1st, 2nd and 3rd part as
    AAAA AAAABBBB BBBB
- First solve the middle (2nd) part: AAAABBBB
  - Reduce by 4: combine(AA+solve(AABB)+BB)
    = combine(AA + BABA + BB)
    = BABABABA (swap 1st A with last B)
- Solve 1st and 3rd part: AAAA (2nd part) BBBB
  - Reduce by 4
    Combine(AA + solve(AA (2nd part) BB)+BB)
    = Combine(AA + (AB (2nd part) AB)+BB)
    =AB+(AB(2nd part) AB)+AB (swap 2nd A with2nd last B)
    =ABAB (2nd part)ABAB
    = ABAB BABABABA ABAB

Q12: Binary Insertion Sort

- General Insertion sort find an appropriate position to insert `A[i]` among the previously sorted array `A[0] <= A[1] <=...<=A[i-1]`.
- Binary insertion sort uses binary search to find an appropriate position to insert `A[i]` among the previously sorted `A[0]<=A[1]<=...<=A[i-1]`.
- Implement Binary Insertion Sort.

A12: Binary Insert

- arr=8,10,19,5,17,20,4,14,18,2
  - Sorted = []
- `I1`: elem: 8. Bin search steps =0
  - sorted = 8
  - unsorted: 10,19,5,17,20,4,14,18,2
- `I2`: elem: 10. Bin search steps =1
  - sorted = 8,10
  - unsorted: 19,5,17,20,4,14,18,2
- `I3`: elem: 19. Bin search steps =1
  - sorted = 8,10, 19
  - unsorted: 5,17,20,4,14,18,2

A12: Binary Insert

- `I3`: elem: 19. Bin search steps =1
  - sorted = 8,10, 19
  - unsorted: 5,17,20,4,14,18,2
- `I4`: elem: 5. Bin search steps =2
  - sorted = 8, 10, 19, 5
  - unsorted: 17,20,4,14,18,2
- `I5`: elem: 17. Bin search steps =2
  - sorted = 8, 10, 19, 5, 17
  - unsorted: 20,4,14,18,2
- `I6`: elem: 20. Bin search steps =3
  - sorted = 8, 10, 19, 5, 17, 20
  - unsorted: 4,14,18,2

… …

Epilogue

- Work with different problem sizes and similar other problems to improve your understanding.