ASSIGNMENT-1

Write algorithm for following divide & Conquer applications

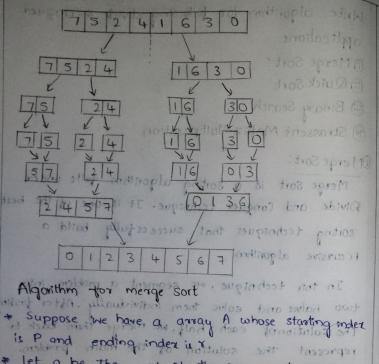
- 1) Merge Sort:
- @ Quick Sort
- 1 Binary Search
- 1 Strassen's Matrix Multiplication

OMerge Sort:

Merge Sort is a sorting algorithm talls under Divide and Conquer technique. It is one of the best sorting techniques that successfully build a recursive algorithm.

In this technique, we segment a problem into two halves and solve them individually. After finding the solution of each half, we merge them back to represent the solution of the main problem. The merge sort is a comparision sort and has an algorithmic complexity of 0 (nlogn). Elementary implementations not merge sort make use of two arrays - one for each half of the data set.

Marginally taster than the heap sort for large data set. It always does less number.



Algorithm for merge sort algorithm for merge sort algorithm for merge sort algorithm and act and suppose we have a array A whose starting index is P and ending index is Y let 2 be the mid of the array in between P, 8 we divide array into A[P,2] and A[2+1,8] be well as a soft and algorithm algorithm.

* We sort theres array and omerge them algorithm 2. Then 2 (P+x)/2 and soft and algorithm algorithm 3. MERGE SORT (A, P,2) and the soft and algorithm.

4. MERGE SORT (A, P,2) and the soft and algorithm algorithm.

```
5. MERGE (A, P,2,7) 130 / 2 2 (1) 75
 Functions: MERGE (A,P,217)
 1. 11=9-P+1 chartens c-0,3 ( see 1)
 2. n2=8-p 10-11 , c to many 11 11
 3. Create arrays [1,... n 1+1] and R[1,... n 2+1]
 4. tor i > 1 to 01 0 1 (010) TE (010)
 5. do[i] - A[P+in] - [o] as + (pla) ps] s
 6. tor j → 1 to n2
 7. dok(i) \rightarrow A[2+j]
                 100 x [ p(1) p 30
8. L[nI+I] → ∞
9. R[12+1] -100 10 10 100 100 100 100 100
10. 1 -> 4 3/000 mt 00 4+ (0) (0) + 4.
 11. 3 - 1
12. For k→ Pto 8
                  (0pa)) as + ao
 13. Do it LUS RIST and to postare and
 14. then A[K] -> L(1) apola ) 0 - last a 0 antister
15. 1->1+1
 16. else A[K] → R[j]
the thirty are bound and trople an it tros show the
 Computing time for merge soit
The time for the merging operation is proportional
to norther computing time for merge sort is
described by using recurrence relation
```

```
T(n) = a if n=1 (ns.4 A) 3843 m. 3
         2T(n/2) + cn it n>1
  Here c,a -> constants 1+9-9-10.1
   It n is power of 2, n=2k q-8-80 8
 Form recurrence relation lenning stores. 3
        T(n) = 2T(n/2) + (n) 10 of 1 < 1 rot. p
        2 [27(n/4)+cn/2]+cn/4)A (-li)0b.3
         4T[n/4] + 2cn co of 1 = 1 rot . 2
          22 T[n/4]+2cn 0 (1410) 3 .3
          25 T (n/e) +3ch for n= 1/4- [1-00] 2 . P
          24 T (1)(6)+4 CD for 1=1/6 1-1-1
          2k +(1) + k(n 1 -1 -1
           ant in (logn) 6019 CT CT
 By representing it by in the form of Asymptotic
 notation O is rin)= O (nlogn) (1) = [2] 4 aut 11
@ Quick Sort (1)A - (x)A sols at
 Quick sort is an algorithm based on divide and
canquer paradigm. that selects a pivot
element and reorders the given list in such a
way that all elements smaller to it are on
   described by any recurrence relation
```

```
One side and those bigger than it are on the
Other Then the sublists are recursively sorted until
the list gets completely sorted. The time complexity
of this algorithm is 0 (nlogn).
 Auxiliary space used in the average case for
 implementing recursive function calls is o(logn) and
 hence proves to be a bit space costly especially
 when it comes to large data sets.
 Its worst case has a time complexity of o(n2)
  which can prove very fatal for large data sets
  Algorithm for QuickSort
   Algorithm quicksort (a, low, high) i (and)
     if (high > low) then of
      m = partition (a, low, high);
      if (lowern) then quick (a low, m);
     if (M+1 < high) then Quick (a, m+1, high);
                     (5000
    Algorithm partition (a, low, high)
       i= low , j= high i
        mid = (low+ high)/2;
        pivot = a[mid];
```

of awhiles (ix=j) and finally (aff)= privat) shis say other the sublists are recorded it the and while acil & pivot) photograp del sot of this adjoint on a control of the overall for va (o pat) o tempe a [1]; outpany - evianisa particisars igan photografilealilings to a of the savary sound a[j] = tempion specif of monos 1: and Its worst case has a line complexitte on?) which can prove very fatal for law class sets return; (dpid, wol, o) trushing motinopla front (well apid) fi Time Complexity: Best Case in oflogin) and (assure) if (MH & High) then guick (a, mar brok); Average Case: o(nlogn) Worst Case: 0(n2) contarg and tropper idplay weigh colleged +well = bion ilbim o foria

Binary Search (room () 11 mon > part)

* This algorithm finds the position of aspectfied input value (the search "key") with an array sorted by key value.

* In each step, the algorithm compares the search key value with the key value of the middle element of the array.

* It key matches then matching element has been thound and its index, or position its returned.

* Otherwise, if the search key is less than the middle element's key then algorithm repeats its action on the Sub array to the left of the middle element or if the search key is greater than then the algorithm repeats on sub array to the right of the middle element.

the search element is less than the provincium position element or greater than maximum position element then this algorithm returns not found.

Algorithm for binary Search (recursive)

-Algorithm binary-search (Aikey; imin, iman)

if (imarkimin) then return "array is empty";

if (key < imin 11 k) iman) then return "element not in "array lut" porrelsers the eyest the search and Jaulov tugar sorted by key value. * in each step the (2) (samit nime) = bird seach ter towards sich [imid] > key) they all alm sulov return binary-search (A, key imin, mid-1); and relse it (Alimid) ckey) then part it return binary-search (A, key, insidted); man); * Otherwise, if the search key is less sile the middle elements key then adoptionmentments to action or the subjury of the left of the fordals element or if the search key is greater than fixen the alg For successful Search due an Mayor milworst case > Ollogn) or Ollogn) bross it is Average case > 0 (log n) or 0 (log n) 10 and 10 as unsuccessful Search amosquad authople o(logn) for all cases. "tylgars " porro" aruter

DStrassen's Matrix Multiplication

In matrix multiplication considering x, y, z

matrices z=xxy. Using Naive method two

matrices (x and y) can be multiplied if the order

of these matrices are pxy and (2xx)

Algorithm for Naive Method (0+3) M

for i=1 to p do (1+3) x (0-A)

Here we assume that, integer operations take O(1) time. There are three folloops in this algorithm and one is nested in other. Hence the algorithm takes $O(n^3)$ time to execute.

In this context, using Strassen's Matrix Multiplication algorithm, the time consumption can be improved a little bit would say to the consumption of the improved a

Strassen's Matrix Multiplication can be performed only ba on square matrices where n is a power of 2, Order of both of the matrices are nxn.

```
Divide X, Y and Z into four (0/2) X(0/2)
   matrices are represented below.

Z=[IJ] x=[AB] Y= [GXH]
 Using Strassen's Algorithmo computer the following
           M, = (AHC) X (TEXFX) 300 (3) Into (01 3 MIN)
           M2 = (B+D) x (G+H) swiph rot mathematically
           M3 = (A-D) x (E+H) ob q of 1=1 80%
           M4= AX(F-H) ob rot 1= 1 rot
           M5 = (C+D) XE 0 =: (1) 5
            MG = (A+B) XH Ob got 1= X rot
        [ : My & D x (G-E) (i) 5 - [ : 1] 5
    endlingto sidt di spool of serve are servet sont (1)0

andingto sidt di spool of servet are servet sont (1)0

andingto sidt di spool of servet are servet (1)0

andingto sidt di spool of servet are servet (1)0

K = Ms + Ms = of servet (20)0 what
a do Arialysis xixto 1 = M1-M3-M4-M5 103000 21 of all
  where cand dare constants
  Using this recurrence relation T(n) = 0 (n691)
   Hence, complexity of strasser's matrix multiplical
```