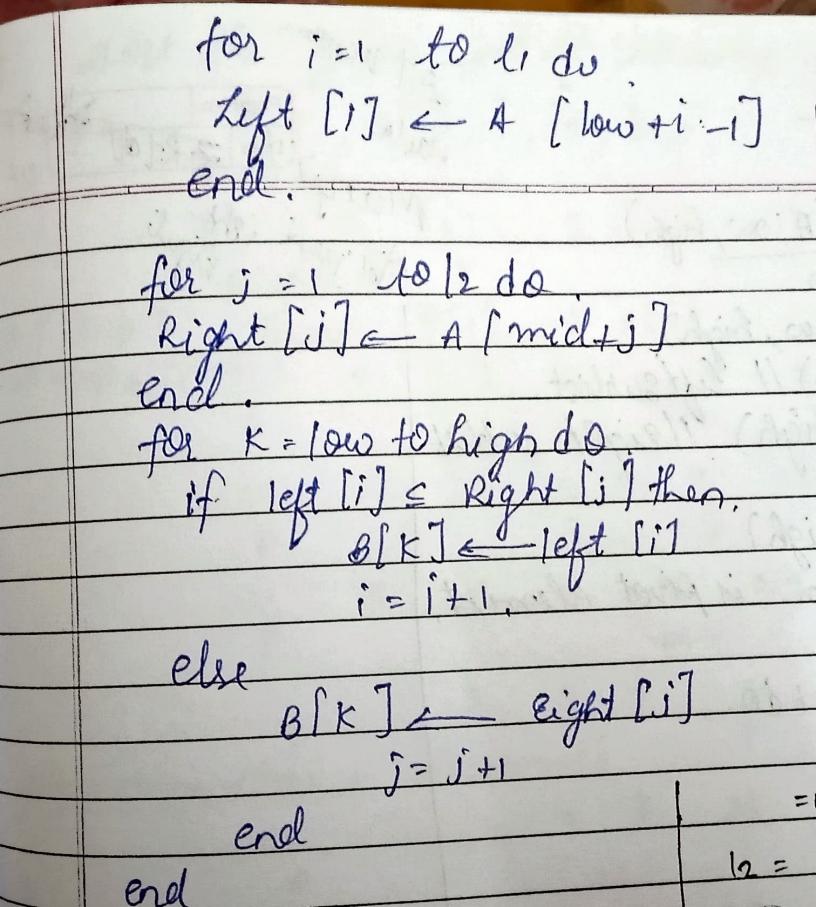
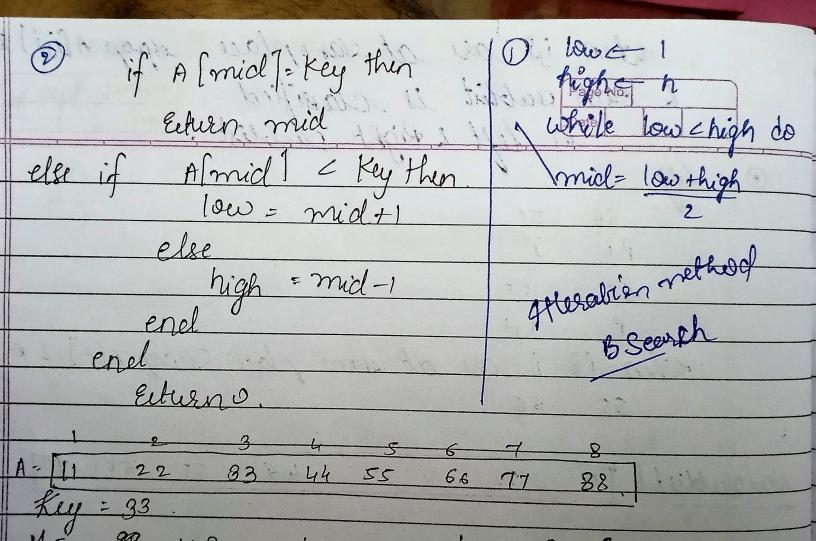


o Theodien Stat: Algerithm Insertion Sort (A) for j=2 to A Length Key = A[j] while i > 0 and A[i]> Key Ali+17 = Ali7 1=1-1 end Ali+1]= Key

# Algorithm (Merge Stort) sorted Algolithm merge-sort (A, low, high) mid & flow ( low high)/2. merge børt (A, mid+1, high) combine (A, low, mid, high) combine (A, low, mid, high) illemid-low+1 12= high- mid. for is! tole do





ding Minimum and Maxinum Algorithm man-min CA [1....n], man, rin)
man-min -A [1] for (i=2 ton) do if (A[i] > max) then man - Ali I 11 Obtaining ranvalue Asil = min) then min = A[i] II Altaining nin value

Fractional-Knapsack P(1...n), w) weight =0 weight + w[i] elsef 19-60

Algorithm JOB\_SCHEDULING(J, D, P) // Description: Schedule the jobs using greedy approach which maximizes the profit // Input: J: Array of N jobs D: Array of deadline for each job P: Array of profit associated with each job Sort all jobs in J in decreasing order of profit If job does not miss its deadline  $S \leftarrow \emptyset // S$  is set of scheduled jobs, initially it is empty // Sum is the profit earned SP ← 0 Add job to solution set far i ← 1 to N do if Job J[i] is seasible then Schedule the job in latest possible free slot meeting its deadline. S-SUJII  $SP \leftarrow SP + P[i]$ Add respective profit end

```
Algorithm FLOYD_APSP (L)
II L is the matrix of size n X n representing original graph
AD is the distance matrix
D \leftarrow l_*
for k - 1 to n do
 for i ←1 to n do
           for j - 1 to n do
              D[i,j]^k \leftarrow \min(D[i,j]^{k,1},D[i,k]^{k,1} + D[k,j]^{k,1}
           end
```

```
Algorithm:
```

```
sumofsubset(s,k,r)
            X[k]=1;
            if (s+W[k]=m) then write(X[1:k]);
            else if (s+W[k]+W[k+1] \le m)
            then sumofsubset(s+W[k], k+1,r-W[k]);
            if ((s+r-W[k]>=m)and(s+W[k+1]<=m)) then
                        X[k]=0;
                        sumofsubset(s, k+1, r-W[k]);
```

```
described below:
Algorithm NAÏVE_STRING_MATCHING(T, P)
// T is the text string of length n
// P is the pattern of length m
for i ← 0 to n - m do
     if P[1...m] == T[i+1...i+m] then
        print "Match Found"
end
         --- laulte analycic
```