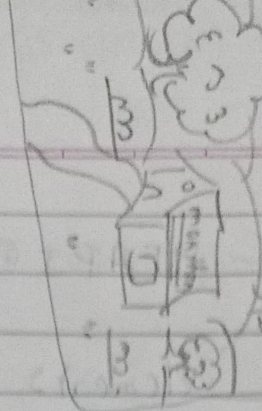


* I) Procedure INSERT (A, n)

```
    //
    //
    integer i, j, n
    j ← n , i ← ⌊n/2⌋ ; item ← A(n)
    while i > 0 & A(i) < item , do
        A[j] ← A(i)
        j ← i
        i ← i/2
    repeat
    A(j) ← item
end INSERT.
```

for value check

```
for i ← 2 to n do
    Call INSERT (A, i)
repeat
```

* ⌊ ⌋ = Lower floor = 1.

* ⌈ ⌉ = upper floor = 2.

1.5

* P: Left child ← i * 2
  Right -11- ← (i * 2) + 1
  Parent ← ⌊i/2⌋

II) ~~ADJUST~~ () — HEAPIFY()

Readjust the elements in a[1:n] to form heap.

```
Procedure HEAPIFY (A, n)
    integer n, i
    for i ← ⌊n/2⌋ to 1 by -1 do
        call ADJUST (A, i, n)
    repeat
END ------
```

adjust this subtree

The complete Binary trees with roots a(2*i) & a(2*i + 1) are combine with a[i] to form single heap, $1 \leq i \leq n$, no node has an address greater than n or less than 1.

j<n = Checking weather it has right child or not.

o Procedure ADJUST (A, i, n)

integer i, j, n
j ← 2 * i, item ← A(i)
while j ≤ n do
if j<n & A(j) < A(j+1), then (ASC)
// ——— > desc.
j ← j+1
end if
if item > A(j)
then exit loop.
else
A(⌊j/2⌋) ← A(j)
j ← 2*j
endif
repeat
A(⌊j/2⌋) ← item

END.

Procedure Heap Sort (A, n)

A : 1 to n contains n elements to be sorted Heap Sort rearranges them into non-decreasing order

integer i

Procedure HEAP SORT (A, n)
Call HEAPIFY (A, n)

// interchange the new maximum with the element at the end & the & adjust the route.

For i = n by 2 by -1 do

Call Exchange (A(i), A(1))
Call Adjust (A, 1, i-1)
Repeat

end - - - -

- Greedy Technology :- KNAPSACK Problem

(algorithm designing tech)

Procedure GREEDY_KNAPSACK (P, W, M, X, n)

Desc :- // P (1:n) & W (1:n) contains the profit & weight resp of n objects Ordered so that descending order of data

$$P(i)/W(i) \geq P(i+1)/W(i+1)$$

// M is the KNAPSACK size & x (1:n) is the soln vector.

Decl :- real P(1:n), W (1:n) x (1:n), M, cu.

* Assuming the data given is sorted according to P/W ratio.

```
Algo :- x ← 0  // initialize soln vector with ze
        cu ← M // remaining knapsack capacity.
        for i ← 1 to n do,
        |   if W (i) > cu, then
        |   |   exit loop
        |   else
        |   |   x(i) ← 1
        |   |   cu ← cu - W(i)
        |   endif
        repeat
```

```
if i ≤ n, then
    x(i) ← u / w(i)
endif
END GREEDY-KNAPSACK
```

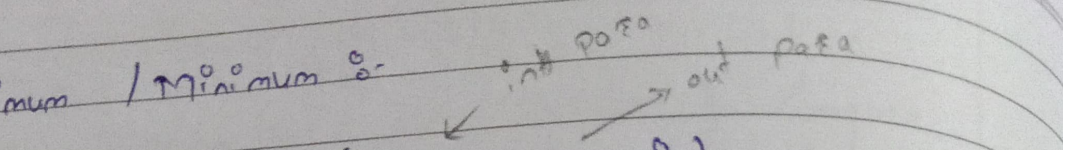
Maximum profit = $\sum_{i=1}^{n} x_i p_i$

Questions:

1) explain Greedy / fractional knapsack problem.
2) Solve following greedy knapsack problem.
3) write fractional knapsack algo.
4) Discuss & explain greedy approach for knapsack problem.
5) what is the greedy approach behind the S.S.S.P.

Maximum / Minimum :-

Procedure MaxMin (P, q, max, min)



Algo :-

```
if  P=q  ,then
;   max ← min ← A(P)
else
     if     P=q-1  ,then
     ;      if  A(P) > A(q) , then
     ;      ;       max ← A(P)
     ;      ;       min ← A(q)
     ;      else
     ;      ;       max ← A(q)
     ;      ;       min ← A(P)
     ;      endif
     else
     ;      m ← (P+q)/2
     ;      Call MaxMin (P, m, fmax, fmin)
     ;      Call  -11-   (m+1, q, hmax, hmin)
     ;      if fmax > hmax , then
     ;      ;       max ← fmax
```

min=1

max = $(\log_2 n) + 1$

$T(n) = O(\log_2 n) + 1$

```
        else
            max ← hmax
        endif
        if fmin < hmin , then
            min ← fmin
        else
            min ← hmin
        endif
    endif
endif
```