Must Do **Greedy Algorithm** problems for **Placement**

Greedy	Activity Selection Problem
Greedy	Job SequencingProblem
Greedy	Huffman Coding
Greedy	Water Connection Problem
Greedy	Fractional Knapsack Problem
Greedy	Greedy Algorithm to find Minimum number of Coins
Greedy	Maximum trains for which stoppage can be provided
Greedy	Minimum Platforms Problem
Greedy	Buy Maximum Stocks if i stocks can be bought on i-th day
Greedy	Find the minimum and maximum amount to buy all N candies
Greedy	Minimize Cash Flow among a given set of friends who have borrowed money from each other
Greedy	Minimum Cost to cut a board into squares
Greedy	Check if it is possible to survive on Island
Greedy	Find maximum meetings in one room
Greedy	Maximum product subset of an array
Greedy	Maximize array sum after K negations
Greedy	Maximize the sum of arr[i]*i
Greedy	Maximum sum of absolute difference of an array
Greedy	Maximize sum of consecutive differences in a circular array
Greedy	Minimum sum of absolute difference of pairs of two arrays
Greedy	Program for Shortest Job First (or SJF) CPU Scheduling
Greedy	Program for Least Recently Used (LRU) Page Replacement algorithm
Greedy	Smallest subset with sum greater than all other elements
Greedy	Chocolate Distribution Problem
Greedy	DEFKIN -Defense of a Kingdom
Greedy	DIEHARD -DIE HARD
Greedy	GERGOVIA -Wine trading in Gergovia
Greedy	Picking Up Chicks
Greedy	CHOCOLA –Chocolate

Greedy	ARRANGE -Arranging Amplifiers
Greedy	K Centers Problem
Greedy	Minimum Cost of ropes
Greedy	Find smallest number with given number of digits and sum of digits
Greedy	Rearrange characters in a string such that no two adjacent are same
Greedy	Find maximum sum possible equal sum of three stacks

Greedy Algorithm

Date 3/4/22

Ques → N meetings in one room (Activity Selection prob)

There is a meeting room in a firm. There are N in form of (start[i], end[i]) where start[i] is start time of meet i, & end[i] is finish time of meeting 'i'.
What is the maximum no. of meeting that can be accomplished, when one meeting held in meet room at a particular time?

⊛ Example :-

S[] = {1, 3, 0, 5, 8, 5} Ans → 4
F[] {2, 4, 6, 7, 9, 9} (1,2), (3,4), (5,7) (8,9)

Imp
(whenever a situation comes where we've to find max. from the given sequence then, there ∞ greedy approach comes into play.)

⊛ Intution :-

∵ we've to find max. from given sequence we'll apply sorting,
we can sort start & finish time, 2 type of sorting start time and end time sorting

In this case we'll use end time sorting
(1,2) (3,4) (0,6) (5,7) (5,9) (8,9)
Here end is same so small start will be considered.

Date: / / Page no:

⊛ Approach :-

⊙ Select all pair in sing order of finish time in each pair.

⊙ Select first meeting of sorted pair as the first meet in room & push into result vector & set variable say → time_limit with second value (F[]) of first selected meet.

⊙ Iterate from the second pair to last pair of array and if value of first ~~or~~ start time of current pair is greater than previously selected pair finish time (time_limit) then select the current pair and update result vector (push selected meet no. into vector) & var time_limit

⊙ print order of meeting from vector.

⊛ Code :-

```
for( i=1; i<n; i++) {
    if ( sa[i].second ] > time_limit ) {
        m.push_back (a[i].second+1);
        time_limit = a[i].first;
    }
}
```

Date: / / Page no:

Ques → Job Sequencing Problem

Given an array of job where every job has a deadline and associated profit if the job is finished before the deadline. it is given that job takes a single unit of time, so the min. possible deadline for any job is 1.

⊛ Example :-

Input :

JobID	a	b	c	d
Deadline	4	1	1	1
Profit	20	10	40	30

Output :- Maximum profit sequence : c, a

⊛ Intution :-

generate all subsets of given set of job and check individual subset for the fesibility of job in that subset.

keep track of maximum profict among all reasible subset.

(This is standard greedy approch.

TC- $O(n^2)$

we can optimize using priority queue (max-heap)

Date: __/__/____ Page no: ____

① Algorithm :-

Sort the job based on their deadlines
Iterate from end and calculate the available slot between every two consecutive deadlines. Include the profit, deadline, and job ID of ith job in the max heap.

While the slot are available and there are jobs left in max heap, include the job ID with max. profit and deadline in the result.

Sort the result array based on their deadlines.

② Code :-

```
for (int i=0; i<n; i++)
    for (int j = min(n, arr[i].dead) -1;
         j>=0; j--)
    if (slot[j] == false)
        result[j] = i;
        slot[j] = true;
        break;
```

Ques → Huffman Coding

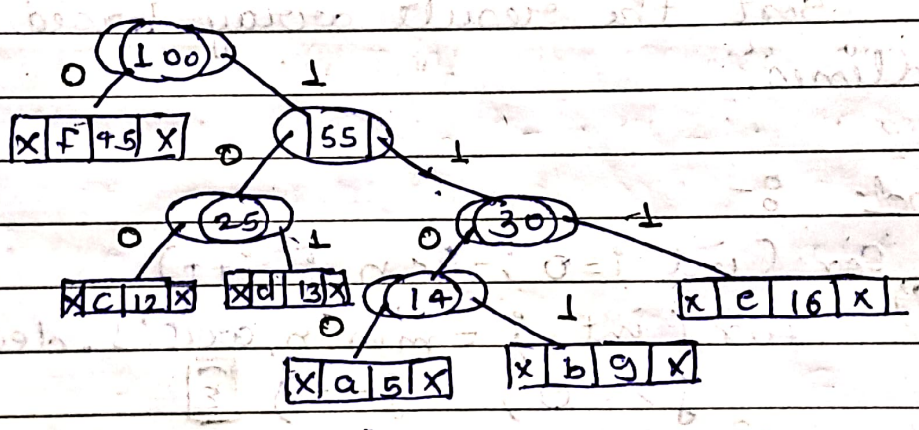
Given string S of distinct character of size N and their corresponding frequency f[].
i.e. character S[i] has f[i] frequency.

Task is to build the huffman tree print all the huffman codes in preorder traversal of the tree.

⊛ Example :-

S = 'abcdef' f[] = {5, 9, 12, 13, 16, 45}

o/p : 0, 100, 101, 1100, 1101, 111



Huffman Code will be :-

f : 0 , c = 100 , d = 101, a = 1100 , b = 1101, e = 111

(print them in preorder traversal)

NOTE:

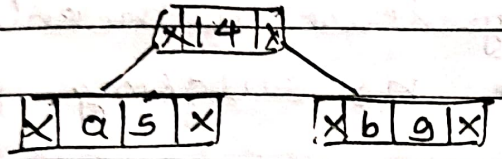
It is use in case where there is series of frequently occouring character.

Date: / / Page no:

Steps to create (min heap) Huffman tree

① Build a min heap that contain 6 nodes where each node repeat present root of tree with single node.

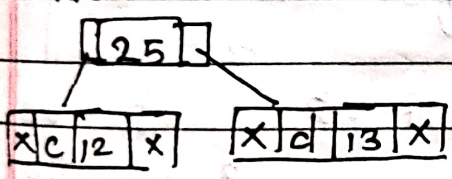
② Extract 2 min. freq. node from min. heap. Add a new internal node with freq. 5 + 9 = 14



Now min heap contain 5 node where 4 nodes are root of tree with single element each and 1 heap node is root with 3 elemen.

Character	c	d	Internal node	e	f
frequency	12	13		16	45

③ Extract two min. freq. nodes from heap. Add a new internal node with freq. 12+13=25.



④ Extract 2 min freq. node. Add new internal node with freq. 14+16=30

⑤ Extract 2 min freq. node. Add a new internal node with freq. 25+30 = 55

⑥ Extract 2 min freq. node. Add new internal node with freq. 45+ 55=100

Date 4/4/22 Page no:

Ques → Water Connection Problem

Every house in colony has atmost 1 pipe going into the house and 1 pipe atmost going out of it. Tanks and taps are to be installed in a manner such that every house with one outgoing pipe but no incomming pipe gets a tank installed on its roof and every house with only on incoming pipe and no outgoing pipe gets a tap.

Given 2 integer n and p denoting no. of houses & the number of pipes the connection of pipe among the house contain 3 input value : a_i, b_i, d_i denotes the pipe of diameter d_i from a_i to house b_i find out the efficient soln for the network

* Example

4 3
1 2 60
3 4 50
4 4 80

output : 3
1 2 60
3 4 50

connected components are :- 1→2 & 3→4
∴, our ans is 2 follow by 1 2 60 and 3 4 50

Date: __/__/____ Page no: ____

⊛ Intution

→ Perform DFS from appropriate houses to find all different connected components. The no. of diff. connected components in our answer T.

→ The next t line of output are the beginning of connected components, end of the connected components and the mono diameter from start to end of connected components in each line.

→ ∴, tank can be installed only on the house having outgoing pipe and no incoming pipe,

→ ∴, these are appropriate house to start DFS from i.e perform DFS from such unvisited houses.

Ques → Fractional Knapsack.

Given weights and values of N items, we need to put these items in a knapsack of capacity W to get maximum total value in the knapsack

Example

N = 3, W = 50

value[] = {60, 100, 120}

weight[] = {10, 20, 30}

output :-

240.00.

(total maximum value of item we can have is 240.00 from given capacity of sack).

Intution :- (greedy approch)

Idea is to calc ratio value/weight for each item & sort on the basis of this ratio.

Then take the item with the highest ratio and add them until we can't add the next item as a whole and at the end add the next item as much as we can, which will always be optimal soln to problem

Date: __/__/____ Page no: ____

Implementation

```
double fractionalknapsack(int w,
        struct Item arr, int n).
{
    sort(arr, arr+n, cmp);
int curWeight = 0;
double finalValue = 0.0;

for(int i=0; i<n; i++) {
    if(curWeight + arr[i].weight <= w)
    {
        curWeight += arr[i].weight;
        finalvalue += arr[i].value;
    }
    else {
        int remain = W - curWeight;
        final value += arr[i].value
                * ((double) remain /
                    (double) arr[i].weight);
        break;
    }
}
```

Date 5/4/22 Page no:

Ques → Greedy Algo to find min. no. of coins (choose and swap - gfg).

You are given a string s of lower case. You can choose any two char in the string & replace all the occurance of first char with second char and vice-versa. find lexicographically smallest string that can be obtained by doing this operation at most once.

* Example

A = "ccad" output = "aacd"

In ccad - we choose a and c & after doing replacement ofr once we get aacd.

* Intution

→ firstly store we're string "ccad", when we pick 'c' we check whether smaller than c exist in string or not, if yes i.e a, replace a and c => "a a c d".

→ string - d c b a → find smaller than d i.e a, b, c ∴ a is smallest ∴, d replace by a "a c b d"

→ String a d c b a = a it self is smallest so no replacement.
Now at d → a is smallest but a already check check for an 2nd smallest i.e b, so string will be d c => "a b c d a"

Date: / / Page no:

(*) Approch :- (S = ccad)

Take a set() s = a, c, d

Now iterate 1 by 1 each char in string, ccad, we don't need to ite replace c so remove c from set,

S = a, b next

Now find the smallest element after c i.e → a < c present after c.

replace a with c & c by a.

(*) Code :-

```
String chooseswap (string a) {
    str set <char> s;
for( i=0 ; i< a.length() ; i++ )  s. insert (a[i]) ;
for (int i=0 ; i<a.length(); i++) {
    s. erase (a[i]) ;
if ( s. empty ()) break;
char ch = * s. begin ();
if (ch< a[i]) {
    int ch2 = a[i];
    for(int j=0; j<a. length(); j++)
    {  if ( a[j] == ch)  a[j] = ch2;
    else if ( a[j] == ch) a[j] = ch;
    }
    break;
}
}
return a;
}
```

Date: __/__/____ Page no: ____

Ques → Maximum train for which stopage can be provided

n-platform & 2 main running railway track for both direct^n. Trains which need to be occupy stop at your station must occupy one platform for their stopage & the train which need not to stop at your station will run away through either of main track without stopping. Now, each train has 3 values, arrival time, departure time, and platform number. We are given 'm' such trains you have to tell max. no. of train for which you can provide stopkpage at your station.

⊛ Example

n = 3 , m = 6

Train no.	Arrival Time	Dept. time	ptt. no.
1	10:00	10:30	1
2	10:10	10:30	1
3	10:00	10:20	2
4	10:30	12:30	2
5	12:00	12:30	3
6	09:00	10:05	1

O/p ⇒ max. stoppage train = 5

If train no. 1 will left to go without stoppage then 2 and 6 can easily be accomplish on platform 1,
3 & 4 on ptt. 2 & 5 on platform 3.

Date: / / Page no:

Intution

→ For n platform we'll simply make n-vectors and put respective trains in those vector acc. to platform no.

→ After that by apply greedy approch, we can solve easily.

NOTE: We'll take input in form of 4 digit integer for arrival & departure time as 1030 will repres 10:30.

→ Also, we will choose a 2-D array for input as arr[n][3] where arr[i][0] denotes arrival time arr[i][1] denotes departure time and arr[i][2] denotes the platform for ith train

Date: / / Page no:

Ques → Minimum Platform problem

Given arrival and departure time of all trains that reach railway station. Find min. no. of platform required for the railway station so that no train is kept waiting.

Consider all train arrive at same day & leave on same day. Arrival & dept. time can never be same for a train but arrival time of 1 train can equal to dept time of another train. At a given instance of time, same platform can't be used for both dept of train & arrival of another train

(*) Example :-

n = 6

arr[] = {0900, 0940, 0950, 1100, 1500, 1800}

dep[] = {0910, 1200, 1120, 1130, 1900, 2000}

output = 3

min. 3 platform are required to safely arrive and depart all trains.

(*) Intuition :-

The idea is to consider all events in sorted order. Once the events are in sorted order, trace the no. of trains at any time keeping track of trains that have arrived, but not departed.

Date: / / Page no:

(*) Algorithm

(1) Sort arrival and departure times of trains.

(2) Creat 2 pointers i=0, and j=0 and a variable to store ans and current count (plat).

(3) Run a loop, while i<n and j<n & compare the ith element of arrival array and j^{th} element of departure array.

(4) If arrival time ≤ dept time, then 1 or more platform is needed so Inc count, i.e., plat ++, and increment 'i'

(5) else if arrival time > dept time. then, one less platform is needed so Dec the count i.e., plat -- and Inc j.

(6) update the ans,
i.e ans = max (ans, plat)

Date 6/4/22 Page no:

Ques → Buy max. stock if i stocks can be bought on i-th day

There is product with infinite stocks. The stock price given for n-days, where arr[i] denotes price of stock on i^{th} day. A rule that, customer can buy at most i stocks on i^{th} day. If customer has k amount of money initially, find out max. no. of stocks customer can buy

(*) EXAMPLE

price[] = {10, 7, 19} => 4 ans

K = 45, customer purchase 1 stock on day 1, 2 stock on day 2, 1 stock on day 3,

10 + (7+7) + 19 = 43 => 4 stock.

(*) Intuition :-

Greedely, we'll buy the stock whose price is lowest, the 2nd smallest lowest price stock, then 3rd lowest & so on

Sort the array, and arrang with pair of index. as :

[7, 2], [10, 1], (19, 3)

price max. stock.

Now we check for available money as 14 <= 45 (if less than available money) update K, i.e => 36.

Date: __/__/____ Page no: ____

maintain a variable 'a' as it hold no. of stock.

a = 2.

Now again check for 10 <= (K-updated)

If yes update a.

Now => a = 2+1 , K = 31 - 10 => 21.

Now again check 19 x 3 > 21.

divide $\frac{K}{\text{Price(per stock)}}$ => $\frac{21}{19} \simeq 1$.

∴ a = 2 + 1 + 1

return a.

(*) Code :-

```
for(int i=0; i<n; i++)  cin>>a[i];
vector<pair<int, int>> v;
for(int i=0; i<n; i++)
v.push_back({a[i], i+1});
sort(v.begin(), v.end());
int ans = 0;
for(int i=0; i<n; i++) {
int price = v[i].first;
int stock = v[i].second;
if( price * stock <= K) {
ans += stock;
K -= (price * stock);
}
else {
ans += (K/price);
K -= price * (K/price);
}
cout << ans;
```

Ques- Find min. and max. amount to buy all N candies.

In a candy store N diff. type of candies are available & the price of all the N different type of candies are provided to you. You are now provided with an attractive offer.

You can buy single candy & get atmost K other candies for free.

Now you've to answere 2 question, firstly find what is the min. amount of money you can spent to buy all N diff. candies. 2nd find max. amount you've to spend to buy all the N diff. candies.

Example :-

N = 4

K = 2

candies [] = {3, 2, 1, 4}

O/P => 3, 7.

Intution :-

We must use the offer & get max. candies back for every purchase. so if we want to minimize the money, we must buy candies at min. cost and get candies of max. cost for free. To maximize the money, we must do the revers

Date: / / Page no:

* for finding min amount :-
Start purchasing from starting and reduce K free candies from last with every single purchase.

* for finding max. amount :-
Start purchasing candies from the end and reduce K free candies from starting in every single purchase

Code :-

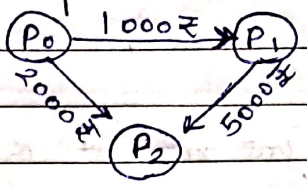
```
int findMin (int arr[], int n, int K)
{
    int res = 0;
for (int i=0; i<n; i++)
    res += arr[i];
    n = n - K;
return res;
}

int findMax (int arr[], int n, int K)
{
    int res=0; index = 0;
for (int i=n-1; i>=index; i--)
{
    res += arr[i];
    index += K;
}
return res;
}
```

07/01/22

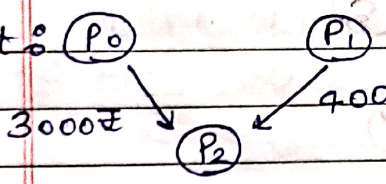
Ques → Maximize cashflow among given set of friend who borrowed money from each other.

Given no of friends who're to give or take some amount of money from one another design algo. by which total cash flow among all the friends is minimized.

Example

Input 

P0 has to pay 1000₹ to P1
P1 " " 5000₹ to P2
P0 also " " 2000₹ to P2.

output : 

P1 pay 4000₹ to P2
P0 pay 3000₹ to P2

Intution (greedy approch)

(1) Do following step for every person Pi where i is from 0 to n-1

(1) Compute net amount for every person. The net amount for person 'i' can be computed by subtracting sum of all debits by from sum of all credit.

amount[maxCredit] - = min;

(2) find 2 person that are max. creditor & debtor. Let max amount to be credited max. creditor be maxCredit & and max. amount to be debited from max. debtor be maxDebit. Let the max. debtor be pd and max. creditor be PC.

```
int mxCredit = getMax(amount);
int mxDebit = getMin(amount);
```

(3) Find minimum of maxDebit & max Credit. Let min of 2 be x. Debit 'x' from Pd & credit this amount to PC.

(4) If x = maxCredd, then remove Pc from person's & recurse for remaing (n-1) person.

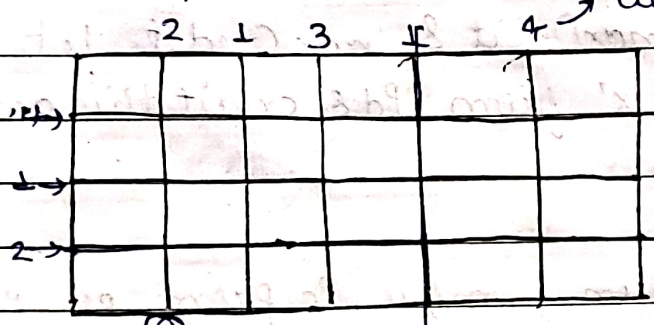
(5) If x = maxDebit, remove Pd from set of persons & recurse for remaing (n-1) persons.

Date: __/__/____ Page no: ____

Ques → Mini cost to cut a board into square

A board of length m & width n is given, we need to break this board into m*n squares such that cost of breaking is min. Cutting cost for each edge will be given for the board in short we choose a sequence of cutting so that cost is minimized.

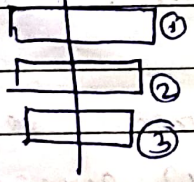
Example



Init value: total_cost = 0

total_cost = total_cost + edge_cost * total_pieces

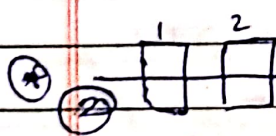
(*) Eg. →



⇒ price = 1, pieces cut = 3 ⇒ 1 × 3 = 3

(cost ↓, no. of pieces ↓)

when we cut vertically we'll see, how many vertical pieces being cut,



⇒ 2 × 3 ⇒ 6 (6 will be added in ans)

when we cut horizontall, we'll be make a count of horizontal pices being cut.

(*) How to think?

∵ we think greedy coz we need min. cost so, firstly we cut high cost edges becoz that time very few no. of pieces available

i.e - sort in descending order:-

(x) = 4, 3, 2, 1 , (y) = 4, 2, 1.

we main tain a count var for both horizontal as well as vertical & ans.

hz = 1; vz = 1; ans = 0;

→ Pick from 4 (from x axis) & vz

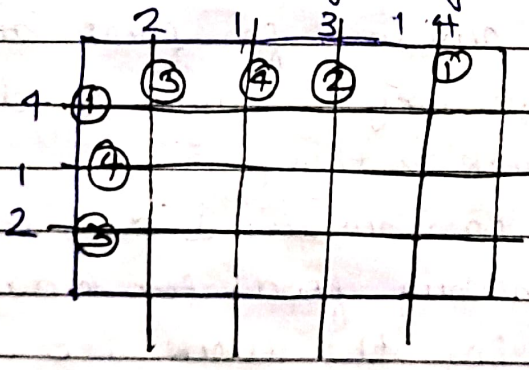
ans = 4 x 1 (when we cut hz, 1 se by 1)

→ Pick 3 (from x)

ans = 4x1 + 4x2 + 3x2 + 2x2 + 2x4

and so on for every cost,

this is a greedy approach to solve



(1, 2, 3, 4 are cutting priority)

Date 8/4/22 Page no:

Ques→ Check if it is possible to survive on Island

You are a poor person in an island, their is only 1 shop in an island, open all day of week except sunday. following conds.
N- Max. food you can buy each day
S- No. of days you are required to survive.
M- unit of food req. each day 2 to survive.

Right now, its monday & you need to survive for next S days. find min. no. of days you need to buy food from shop. so that you can survive next S days.
If not NO else say yes.

⊛ EXAMPLE -

I/P S=10, N=16 M=2
O/P - Yes ⇒ 2

⊛ Intuition :-

If we can survive for first 7 days then we can survive any number of days. for that we need to chick 2 things :
→ check whether we can survive 1 day or not
→ (S>=7) if we can buy food for 6 days & we can survive for the week i.e total food we can buy in a week (6*N) >= total food we required to survive in a week (7*6M) then we can survive.

⊛ Implementation

```
void service (int S, int N, int M)
{
    if ((N*6) < (M*7) && S>6) || M>N)
    {
        cout << "NO";
    }
    else {
        int days = (M*S)/N;
        if ((M*S) % N) != 0)
            days++;
        cout << "Yes" << days << endl;
    }
}
```

TC - O(1)

SC - O(1)

Date: / / Page no:

Ques- Maximum product subset of an array

Given array 'a', we've to find max. product possible with the subset of elements present in the array. Max. prod. can be single element also.

(*) Example :-

a[] = {-1, -1, -2, 4, 3}

output = 24.

max product will be => (-2 * -1 * 4 * 3) = 24

(*) Intution :-

(1) If there is even no. of -ve no. & no zero, result is simply prod. of all.

(2) If odd no. of -ve no. & no zero, result is product of all except -ve int. with least abs. value.

(3) If there are zeros, then result is product of all except these zeros with 1 exceptional case.

(#) The exceptional case is when, 1 -ve no. and all are 0 then result is 0.

Date 9/4/22 Page no:

Ques→ Maximize sum after k-negation

Given an array of int, size N and no. K, your must modifying array arr[] exactly K no. of time. Here modify array means in each operation you can replace any array element either arr[i] by -arr[i] or +arr[i] by arr[i] you need to perform this operation in such a way that after K operations, sum of array must be max.

⊛ Example :-

N=5, K=1 arr[] = {1, 2, -3, 4, 5-3}

output => 15

We've K=1, so that we can change -3 to 3 and sum all element to produce 15 as output.

⊛ Intution :-

We just have to replace the min. element arr[i] in array by -arr[i] for current operation.

In this way we can make sum of array maximize after K operation. One intersting case is, once min. element becomes 0, we don't need to make any more changes.

Date: __/__/____ Page no: ____

⊛ Implementation

```
int MaxSum ( int arr[], int n, int k )
{
    int min = INT_MAX;
    int max = INT_MIN;
    int index = -1;
for ( int j=0; j<n; j++)
{   if (arr[j] < min) {
        min = arr[j];
        index = j;
    }
}
if (min == 0)    break;
arr[index] = -arr[index];
    int sum = 0;
for ( int i = 0; i<n; i++)
    sum += arr[i];
return sum;
}
```

Date: / / Page no:

Ques → Maximize sum(arr[i]*i) of array

Given an array A of N size, task is write a prog. to find max. value of Σarr[i]*i, where i = 0, 1, 2, ---, n-1.

⊛ Example :-

Arr[] = {5, 3, 2, 4, 1}

O/P = 40

If we arrange array as, 1, 2, 3, 4, 5 then we can see that min. index will multiply with min. no. & max. index will multiply with max. no.

⊛ Intuition :-

The fact that the largest value should be scaled max. and the smallest value should be scaled min.

So we multiply the min. value of i, with the min. value of arr[i]. So, sort the given array in ↑sing order and compute of the sum of array[i]*i, where i = 0 to n-1.

Date: __/__/____ Page no: ____

Snippet

```
int maxSum(int arr[], int n)
{
    sort(arr, arr+n);
int sum=0;
    for(int i=0; i<n; i++)
    sum += (arr[i] * i);

return sum;
}
```

Date: / / Page no:

Ques → Maximize sum of absolute difference of any permutation

Given array, we need to find the max. sum of abs. diff. of any permutation of given array

(*) Example :-

Input : {1, 2, 4, 8} O/P = 18

for given array there are several seq like : {2, 1, 4, 8}, {9, 2, 1, 8} & some more

Now abs. diff. of any seq. & will be like for this array sequence {1, 2, 4, 8} the abs. diff sum is

$= |1-2| + |2-4| + |4-8| + |8-1| = 14$

for the given array,

$|1-8| + |8-2| + |2-4| + |4-1| = 18$

(*) Intution :- (Greedy)

(1) We'll sort the array

(2) Calc. final seq. by taking 1 smallest element and largest element from the sorted array & make one vector array of this final size

(3) Finally, calc. sum of abs. diff. b/w elements of array

Date: / / Page No.

* Snippet :-

```
int MaxSumDiff(int a[], int n)
{
    vector<int> final;
    sort(a, a+n);
    for(int i=0; i<n/2; i++)
    {
        final.push_back(a[i]);
        final.push_back(a[n-i-1]);
    }
    if(n%2 != 0)
        final.push_back(a[n/2]);
    int maxi = 0;
    for(int i=0; i<n-1; i++)
    {
        maxi = maxi + abs(final[i] - final[i+1]);
    }
    maxi = maxi + abs(final[n-1] - final[0]);

    return maxi;
}
```