Shell sort

- Invented by Donald shell in 1959

- This is the first algorithm to break the quadratic time barrier but few years later, a sub quadratic time bound was proven.

- shell sort works by comparing elements that are distinct rather than adjacent elements in an array.

- shell sort uses a sequence $h_1, h_2 \ldots h_t$ called increment sequence.

- Any increment sequence is find as long as $h_1 = 1$ and some other choices are better than others.

- shell short makes multiple passes through a list and sort a number of equally sized sets using the insertion sort.

→ shell sort improves the efficiency of insertion sort by quickly shifting the values to their destination.

- Shell sort is also known as diminishing increment sort
- The distance between comparisons decreases as the sorting algorithm runs until the last phase in which adjacent elements are compared.
- After each phase and some increment $h_1 <$ for every i we have $a[i] \le a[i+h_k]$
- All elements spaced h_k apart are sorted.

Advantage:-

- only efficient for medium sized arrays
- 5 times faster than bubble sort

Example

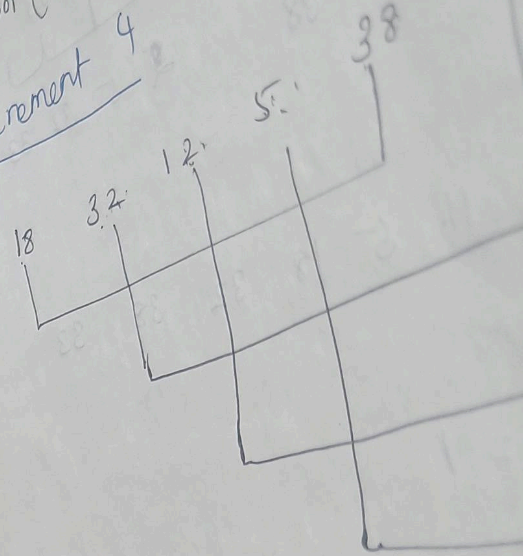
0	1	2	3	4	5	6	7
18	32	12	5	38	33	16	2

Total no of element n=8

Shell increment will

(1) $floor(8/2) = 4$

Increment 4



Compare 18 & 38 (No swa

Compare 32 & 33 (No swap

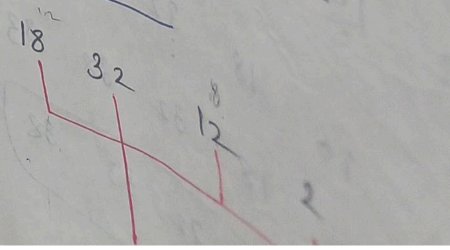
Compare 12 & 16 (No swa

Compare 5 & 2 (5 is max
with

18 32 12 2 38 3

(2) $floor(4/2) = 2$

Increment 2



- Shell sort is also known as diminishing increment sort

- The distance between comparisons decreases as the sorting algorithm runs until the last phase in which adjacent elements are compared.

- After each phase and some increment h_k for every i we have $a[i] \le a[i+h_k]$

- All elements spaced h_k apart are sorted.

Advantage:-

- only efficient for medium sized arrays
- 5 times faster than bubble sort

Example

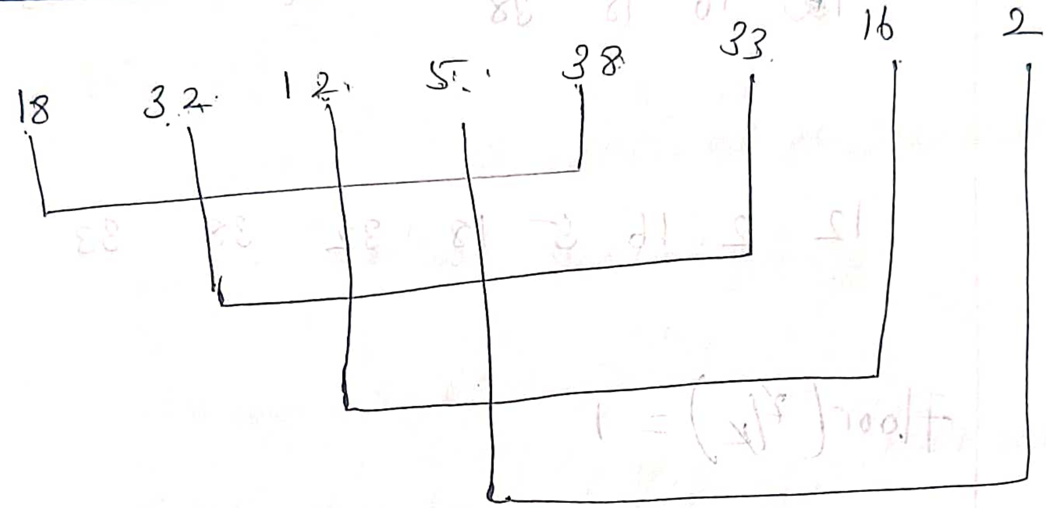
0	1	2	3	4	5	6	7
18	32	12	5	38	33	16	2

Total no of element $n=8$

Shell increment will be $\lfloor N/2 \rfloor$

(1) floor (8/2) = 4

Increment 4

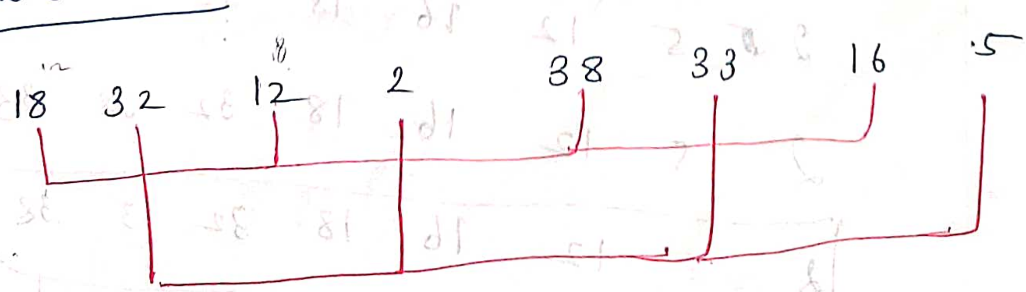


Compare 18 & 38 (No swap)
Compare 32 & 33 (No swap)
Compare 12 & 16 (No swap)
Compare 5 & 2 (5 is maximum so swap with 2)

18	32	12	2	38	33	16	5

(2) floor (4/2) = 2

Increment 2



{18, 12, 38, 16} {32, 2, 33, 5}

18 12 38 16 32 2 33 5

12 16 18 38 2 5 32 33

12 2 16 5 18 32 38 33

Floor(7/k) = 1

Increment 1

0 1 2 3 4 5 6 7

12 2 16 5 18 32 38 33

2 12 16 5 18 32 38 33

2 12 16 5 18 32 38 33

2 5 12 16 18 32 38 33

2 5 12 16 18 32 38 33

2 5 12 16 18 32 38 33

2 5 12 16 18 32 33 38

Shell sort Algorithm

```
Void shellsort (int A[], int N)
{
    int i, j, Increment;
    int temp;
    for(Increment =N/2 ; increment >0; Increment/=2)
      for (i=Increment; i<N; i++)
      {
          Tmp = A[i];
          for (j=i; j>=Increment ; j-=Increment)
          if (Tmp < A[j-Increment])
              A[j]= A[j-Increment];
          else
              break;
          A[j]=Temp;
      }
}
```

0	1	2	3	4	5	6	7
48	32	12	5	38	33	16	2

N=8

Increment = 8/2 = 4 , 4 > 0

i = 4

① Tmp = A[4] = 38

j = 4   4 >= 4 ✓

38 < A[4 − 4]

38 < A[0]

38 < 18

A[4] =

N/2

Increment = ~~7~~/2 = 4/2 = 2

i = 5

(ii) Tmp = A[5]

Tmp = 33

j = 5 ; 5 >= 4 ✓

33 < A[5 − 4]

33 < 32 ×

i = 6

Tmp = 16

6 >= 4 ✓

16 < A[2]

16 < 12 ×

j = 7

Tmp = A[7] = 2

7 >= 4 true

2 < A[7 − 4]

2 < A[3]

2 < 5 ✓

A[7] = A[7 − 4]

A[7] = 5

A[3] = 2

j = 7 − 4

j = 3