Shellsort

Class 18

Shellsort 1

3	5	0	1	7	6	4	9	8	2
0	1	2	3	4	5	6	7	8	9

elements 0 and 5 — are they in relative order with each other?

yes, so move on

Shellsort 2

3	5	0	1	7	6	4	9	8	2
0	1	2	3	4	5	6	7	8	9

look at elements 1 and 6 — are they in order?

no, so order them using insertion sort on this pair

Shellsort 2

3	4	0	1	7	6	5	9	8	2
0	1	2	3	4	5	6	7	8	9

look at elements 1 and 6 — are they in order?

no, so order them using insertion sort on this pair

Shellsort 3

3	4	0	1	7	6	5	9	8	2
0	1	2	3	4	5	6	7	8	9

repeat with elements 2 and 7 — are they in order?

yes, keep going

Shellsort 4

3	4	0	1	7	6	5	9	8	2
0	1	2	3	4	5	6	7	8	9

repeat with elements 3 and 8 — are they in order?

yes, keep going

Shellsort 5

3	4	0	1	7	6	5	9	8	2
0	1	2	3	4	5	6	7	8	9

repeat with elements 4 and 9 — are they in order?

no, so order them (insertion sort)

Shellsort 5

3	4	0	1	2	6	5	9	8	7
0	1	2	3	4	5	6	7	8	9

repeat with elements 4 and 9 — are they in order?

no, so order them (insertion sort)

Shellsort 5

3	4	0	1	2	6	5	9	8	7
0	1	2	3	4	5	6	7	8	9

repeat with elements 4 and 9 — are they in order?

no, so order them (insertion sort)

at this point, the array is 5-sorted

Shellsort 6

3	4	0	1	2	6	5	9	8	7
0	1	2	3	4	5	6	7	8	9

start over with element 0 and every third subsequent element — are they in order?

no, so order them with insertion sort

Shellsort 6

1	4	0	3	2	6	5	9	8	7
0	1	2	3	4	5	6	7	8	9

start over with element 0 and every third subsequent element — are they in order?

no, so order them with insertion sort

Shellsort 7

1	4	0	3	2	6	5	9	8	7
0	1	2	3	4	5	6	7	8	9

now element 1 and every third subsequent element — are they in order?

no, so order them

Shellsort 7

1	2	0	3	4	6	5	9	8	7
0	1	2	3	4	5	6	7	8	9

now element 1 and every third subsequent element — are they in order?

no, so order them

Shellsort 8

1	2	0	3	4	6	5	9	8	7
0	1	2	3	4	5	6	7	8	9

now element 2 and every third one after it — are they in order?

yes, so no change

Shellsort 9

1	2	0	3	4	6	5	9	8	7
0	1	2	3	4	5	6	7	8	9

now element 2 and every third — are they in order?

again no change — the array is now 3-sorted

Shellsort 10

0	1	2	3	4	5	6	7	8	9
1	2	0	3	4	6	5	9	8	7

finally, look at the entire array — is it sorted?

no: the 0, 5, 7, and 8 are out of order

so apply normal insertion sort on the entire array

the array is now sorted

Shellsort 10

0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9

finally, look at the entire array — is it sorted?

no: the 0, 5, 7, and 8 are out of order

so apply normal insertion sort on the entire array

the array is now sorted

```
void shellsort(vector<unsigned>& array)
{
  size_t n = array.size();
  vector<size_t> gaps {5, 3, 1};

  for (auto gap : gaps)
  {
    for (size_t i = gap; i < n; i++)
    {
      unsigned temp = array.at(i);
      size_t j = i;
      while (j >= gap && temp < array.at(j - gap))
      {
        array.at(j) = array.at(j - gap);
        j -= gap;
      }

      array.at(j) = temp;
    }
  }
}
```

Shellsort Analysis

- named for Donald Shell
- B.S. chemical engineering from Michigan Tech in 3 years with the highest GPA in the school's history (record lasted 30 years)
- M.S. and Ph.D. in mathematics
- 1959 published the Shell sort algorithm

- the first known sorting algorithm to break the $O(n^2)$ barrier

- input size is # of array elements
- assume array of gaps is small and finite
- best and worst cases?

Shellsort Analysis

- named for Donald Shell
- B.S. chemical engineering from Michigan Tech in 3 years with the highest GPA in the school's history (record lasted 30 years)
- M.S. and Ph.D. in mathematics
- 1959 published the Shell sort algorithm

- the first known sorting algorithm to break the $O(n^2)$ barrier

- input size is # of array elements
- assume array of gaps is small and finite
- best and worst cases? yes — the while loop

Shellsort Analysis

- named for Donald Shell
- B.S. chemical engineering from Michigan Tech in 3 years with the highest GPA in the school's history (record lasted 30 years)
- M.S. and Ph.D. in mathematics
- 1959 published the Shell sort algorithm

- the first known sorting algorithm to break the $O(n^2)$ barrier

- input size is # of array elements
- assume array of gaps is small and finite
- best and worst cases? yes — the while loop
- best case:
- worst case:

Shellsort Analysis

- named for Donald Shell
- B.S. chemical engineering from Michigan Tech in 3 years with the highest GPA in the school's history (record lasted 30 years)
- M.S. and Ph.D. in mathematics
- 1959 published the Shell sort algorithm

- the first known sorting algorithm to break the $O(n^2)$ barrier

- input size is # of array elements
- assume array of gaps is small and finite
- best and worst cases? yes — the while loop
- best case: linear: while loop never runs
- worst case: quadratic: while loop runs n times

Shellsort

- but this analysis is simplistic
- in fact, analyzing shellsort is fiendishly difficult
- way beyond the scope of this class
- Shell proposed gap sizes of $n/2, n/4, \ldots, n/n$
- these are actually quite bad

- 1963 Hibbard proposed gaps of $2^k - 1, 2^{k-1} - 1, \ldots, 7, 3, 1$
- this gives dramatically improved performance (why?)

- much research into improved gap sequences
- much research into tighter bounds analysis

Visualization

https://www.youtube.com/watch?v=ZZuD6iUe3Pc

Search a List

- search an unsorted list for an element
- analyze two strategies:
  1. search the unsorted list to see if the element exists
  2. sort the list first, then search to see if the element exists

	Strategy 1 (unsorted)	Strategy 2 (sort first)
worst case		
best case		

Search a List

- search an unsorted list for an element
- analyze two strategies:
  1. search the unsorted list to see if the element exists
  2. sort the list first, then search to see if the element exists

	Strategy 1 (unsorted)	Strategy 2 (sort first)
worst case	$T(n) \in O(n)$	$T(n) \in O(n \lg n)$
best case	$T(n) \in \Omega(1)$	$T(n) \in \Omega(n \lg n)$

Search a List

- search an unsorted list for an element
- analyze two strategies:
  1. search the unsorted list to see if the element exists
  2. sort the list first, then search to see if the element exists

	Strategy 1 (unsorted)	Strategy 2 (sort first)
worst case	$T(n) \in O(n)$	$T(n) \in O(n \lg n)$
best case	$T(n) \in \Omega(1)$	$T(n) \in \Omega(n \lg n)$

- clearly, strategy 1 is superior

Multiple Runs

- now assume we are not going to do just one search
- assume we are going to do many (m) searches
- what is the analysis over m searches?

	Strategy 1 (unsorted)	Strategy 2 (sort first)
worst case		
best case		

Multiple Runs

- now assume we are not going to do just one search
- assume we are going to do many (m) searches
- what is the analysis over m searches?

	Strategy 1 (unsorted)	Strategy 2 (sort first)
worst case	$T(m, n) \in O(mn)$	$T(m, n) \in O(m \lg n + n \lg n)$
best case	$T(m, n) \in \Omega(m)$	$T(m, n) \in \Omega(m + n \lg n)$

Multiple Runs

- now assume we are not going to do just one search
- assume we are going to do many (m) searches
- what is the analysis over m searches?

	Strategy 1 (unsorted)	Strategy 2 (sort first)
worst case	$T(m,n) \in O(mn)$	$T(m,n) \in O(m \lg n + n \lg n)$
best case	$T(m,n) \in \Omega(m)$	$T(m,n) \in \Omega(m + n \lg n)$

- what value of m is the break-even point?

Presorting

- sometimes a job is easier if the values are known to be sorted
    - check if a value is unique
    - find the mode of a list of values
    - find the median of a list of values

Transform and Conquer

- a name for this strategy transform and conquer
- modify the arrangement of the data in the data structure to improve either the ease or speed of solution