BASIC SORTING, PART 2

cs2420 | Introduction to Algorithms and Data Structures | Spring 2015

administrivia…

- assignment 3 is due tonight at midnight (11:59pm)

- assignment 4 is out later today
  - requires pair programming
  - due next Thursday

- midterm 1 in two weeks
  - Monday's lab will cover exam review questions
    - *review questions posted by Sunday night*
  - no lab the week of the midterm

last time…

sorting

- sorting is a fundamental application in computing
  - one of the most intensively studied and important operations

- most data is useless unless it is in some kind of order

- for any given problem, or specific goal isn't necessarily sorting… but we often need to sort to efficiently solve problems
  - computer graphics
  - look-up tables
  - games

selection sort

the simplest sorting algorithm

insertion sort

good for small **N**

selection sort

1) find the minimum item in the unsorted part of the array

2) swap it with the first item in the unsorted part of the array

3) repeat steps 1 and 2 to sort the remainder of the array

WHAT DOES THIS LOOK LIKE?

```
void selectionSort(int[] arr)
{
  for(int i=0; i < arr.length-1; i++)
  {
    min = i;
    for(int j=i+1; j < arr.length; j++)
      if (arr[j] < arr[min])
        min = j;

    temp = arr[i];
    arr[i] = arr[min]
    arr[min] = temp;
  }
}
```

A) **c**
B) **log N**
C) **N**
D) **N log N**
E) **N^2**
F) **N^3**

WHAT IS THE COMPLEXITY OF SELECTION SORT?

insertion sort

1) the first array item is the sorted portion of the array

2) take the second item and insert it in the sorted portion

3) repeat steps 1 and 2 to sort the remainder of the array

WHAT DOES THIS LOOK LIKE?

```
void insertionSort(int[] arr)
{
  for(int i=1; i < arr.length; i++)
  {
    index = arr[i];
    j = i;
    while(j>0 && arr[j-1]>index)
    {
      arr[j] = arr[j-1];
      j--;
    }
    arr[j] = index;
  }
}
```

WHAT IS THE COMPLEXITY OF INSERTION SORT?

unsortedness

-requires a measure of ***unsortedness*** for array

-**inversion**: a pair of array items that are out of order

45	-3	9	76	11	-8	0

HOW MANY INVERSIONS ARE THERE?

-sorting efficiency depends on how many inversions are removed per step

insertion sort complexity

each swap to the left removes one inversion…

…we must visit each item at least once (**N**)…

…and we must undo **I** inversions

45	-3	9	76	11	-8	0

SWAP REMOVES ONE INVERSION

insertion sort is O(N+I)

HOW DO WE FIGURE OUT WHAT **I** IS?

today…

-measuring the complexity of insertion sort

-shellsort

insertion sort is O(N+I)

HOW DO WE FIGURE OUT WHAT **I** IS?

worst case scenario…

-what are the number of inversions in the worst case?

-what ***IS*** the worst case?

76	45	11	9	0	-3	-8

INVERTED

HOW MANY INVERSIONS ARE THERE?

-when every **unique pair** is inverted…

-how many unique pairs are there?

-(hint: remember Gauss's trick!)

$$N * (N-1)/2 = (N^2 - N)/2$$

insertion sort is O(N+I)

WHAT IS THE WORST-CASE COMPLEXITY OF INSERTION SORT?

A) c
B) log N
C) N
D) N log N
E) N^2
F) N^3

insertion sort is O(N+I)

WHAT IS THE **BEST**-CASE COMPLEXITY OF INSERTION SORT?

A) c
B) log N
C) N
D) N log N
E) N^2
F) N^3

average case scenario…

-assume that there is a 50% chance that any given pair is inverted

-average number of inversions = (number of pairs) / 2

$$((N^2 - N) / 2) / 2 = (N^2 - N) / 4$$

NUMBER OF PAIRS

insertion sort is O(N+I)

WHAT IS THE AVERAGE-CASE COMPLEXITY OF INSERTION SORT?

A) c
B) log N
C) N
D) N log N
E) N^2
F) N^3

recap…

selection vs insertion

	selection	insertion
WORST:	$O(N^2)$	$O(N^2)$
AVERAGE:	$O(N^2)$	$O(N^2)$
BEST:	$O(N^2)$	$O(N)$

WHICH ONE PERFORMS BETTER IN PRACTICE?

A) **selection**
B) **insertion**

summary

- an ***inversion*** is a pair of items that are out of order
    - a sorted array has 0 inversions
    - an average (and worst) array has ~N^2 inversions

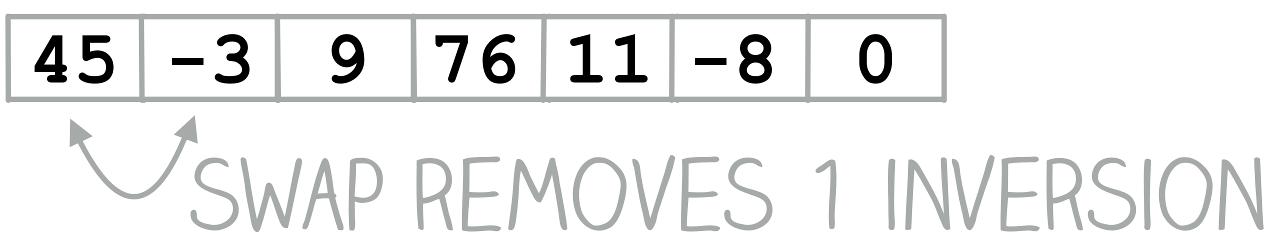
- thus, we must undo N^2 inversions

- to do better than **$O(N^2)$** we must remove more than 1 inversion per step
    - (insertion sort only removes 1 inversion per step!)

what we want…

-a sorting algorithm that has **subquadratic** complexity

-swapping adjacent items removes exactly 1 inversion



-what if we consider swapping nonadjacent pairs?

-removes inversions not involved with the swap

shellsort

the simplest subquadratic sorting algorithm

shellsort

insertion sort, with a twist

1) set the **gap size** to **N/2**
2) consider the subarrays with elements at **gap size** from each other
3) do insertion sort on each of the subarrays
4) divide the **gap size** by 2
5) repeat steps 2 — 4 until the is **gap size** is <1

WHAT DOES THIS LOOK LIKE?

HOW DO WE DESCRIBE INSERTION SORT WITH RESPECT TO SHELLSORT?

-each *x*-sort (for a gap *x*) is performing an insertion sort on *x* independent subarrays

-is also known as the *diminishing gap sort*

-Shell originally suggested gaps **N/2, N/4, N/8, ..., 1**

- -gap sequences in which consecutive gaps share no common factors have been shown to perform better

DIMINISHING GAP SEQUENCE

```
void shellSort(int[] arr)
{
  for(gap = arr.length/2; gap > 0; gap /= 2)
  {
    for(i = gap; i < arr.length; i++)
    {
      val = arr[i];
      for(j = i-gap; j >= 0 && arr[j] > val; j -= gap)
        arr[j+gap] = arr[j];
      arr[j+gap] = val;
    }
  }
}
```

ITEM TO BE INSERTED

INSERT ITEM

UNTIL THE INSERTION POSITION IS FOUND, SHIFT SORTED ITEMS

WHAT IS THE COMPLEXITY OF SHELLSORT?

shellsort complexity

-**worst case:** **$O(N^2)$** with Shell's gaps, **$O(N^{3/2})$** with better gaps

-**average case:** **$O(N^{3/2})$** with Shell's gaps, **$O(N^{5/4})$** with better gaps

-proofs of these bounds are complicated
    -the **$O(N^{5/4})$** bound is based on simulations only!

-insertion sort performs better the more sorted the array
    -remember, approaches **$O(N)$** for a sorted array!

shellsort complexity

- still, **$O(N^{5/4})$** is an encouraging bound for the average case
- for moderate **N**, this is better than **O(N log N)** algorithms
- around **N=100K**, **O(N log N)** wins
- best sorting algorithms are **O(N log N)**
    - **log N** suggests repeated dividing by 2
    - "divide and conquer"

WHAT ALGORITHM DO WE KNOW OF THAT IS **log N**?

WHAT DOES THIS IMPLY ABOUT THE "CONQUER" STEP?

next time…

- reading
  - chapters 7 & 8.5 - 8.8
- homework
  - assignment 3 due today
  - assignment 4 out today