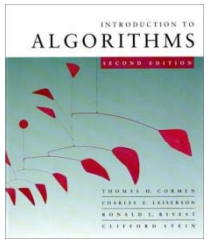

CMPS 2200 – Fall 2012

Randomized Algorithms & Quicksort

Carola Wenk

Slides courtesy of Charles Leiserson with additions by Carola Wenk

Deterministic Algorithms

Runtime for deterministic algorithms with input size *n*:

- Best-case runtime

  ➔Attained by one input of size *n*

- Worst-case runtime

  ➔ Attained by one input of size *n*

- Average runtime

  ➔ Averaged **over all possible inputs** of size *n*

Deterministic Algorithms: Insertion Sort

Best-case runtime: $O(n)$, input $[1,2,3,\ldots,n]$

➔ Attained by one input of size n

- Worst-case runtime: $O(n^2)$, input $[n, n\text{-}1, \ldots,2,1]$

➔ Attained by one input of size n

- Average runtime : $O(n^2)$

➔ Averaged **over all possible inputs** of size n

- What kind of inputs are there?
- How many inputs are there?

Average Runtime

- What kind of inputs are there?
  - Do [1,2,…,*n*] and [5,6,…,*n*+5] cause different behavior of Insertion Sort?
  - No. Therefore it suffices to only consider all permutations of [1,2,…,*n*] .
- How many inputs are there?
  - There are *n*! different permutations of [1,2,…,*n*]

Average Runtime Insertion Sort: *n*=4

```
for j=2 to n {
   key = A[j]
   // insert A[j] into sorted sequen
   i=j-1
   while(i>0 && A[i]>key){
      A[i+1]=A[i]
      i--
   }
   A[i+1]=key
}
```

- Inputs: 4!=24

[1,2,3,4] **0**	[4,1,2,3] **3**	[4,1,3,2] **4**	[4,3,2,1] **6**
[2,1,3,4] **1**	[1,4,2,3] **2**	[1,4,3,2] **3**	[3,4,2,1] **5**
[1,3,2,4] **1**	[1,2,4,3] **1**	[1,3,4,2] **2**	[3,2,4,1] **4**
[3,1,2,4] **2**	[4,2,1,3] **4**	[4,3,1,2] **5**	[4,2,3,1] **5**
[3,2,1,4] **3**	[2,1,4,3] **2**	[3,4,1,2] **4**	[2,4,3,1] **4**
[2,3,1,4] **2**	[2,4,1,3] **3**	[3,1,4,2] **3**	[2,3,4,1] **3**

- Runtime is proportional to: 3 + **#times in while loop**
- Best: 3+**0**, Worst: 3+**6**=9, Average: 3+**72**/24 = 6

Average Runtime: Insertion Sort

- The average runtime averages runtimes over all $n!$ different input permutations
- Disadvantage of considering average runtime:
  - There are still worst-case inputs that will have the worst-case runtime
  - Are all inputs really equally likely? That depends on the application

$\Rightarrow$ **Better:** Use a randomized algorithm

Randomized Algorithm: Insertion Sort

- **Randomize the order of the input array:**
  - Either prior to calling insertion sort,
  - or during insertion sort (insert random element)
- This makes the runtime depend on a probabilistic experiment (sequence of numbers obtained from random number generator)

  $\Rightarrow$Runtime is a random variable (maps sequence of random numbers to runtimes)
- **Expected runtime** = expected value of runtime random variable

Randomized Algorithm: Insertion Sort

- Runtime is independent of input order ([1,2,3,4] may have good or bad runtime, depending on sequence of random numbers)
- No assumptions need to be made about input distribution
- No one specific input elicits worst-case behavior
- The worst case is determined only by the output of a random-number generator.

$\Rightarrow$ When possible use expected runtimes of randomized algorithms instead of average case analysis of deterministic algorithms