CS F364: Design & Analysis of Algorithms

Lecture-01: Introduction

Dr. Kamlesh Tiwari,
Assistant Professor,
Department of Computer Science and Information Systems,
BITS Pilani, Rajasthan-333031 INDIA

Jan 17, 2017 (Campus @ BITS-Pilani Jan-May 2017)

Logistics: (CS F364) Design & Analysis of Algorithms

- T Th S (12-1PM) 6164@BITS-Pilani
- Two instructors **Abhishek Mishra** and **Kamlesh Tiwari**. Jointly to be taught.
- **BOOK:** T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, *Introduction to Algorithms*, 3rd Edition, PHI, 2009.
- Grading
  - Three Quiz (10$\times$3=30%)
  - Mid Semester Exam (30%)
  - Comprehensive Exam (40%)

Learn some basic algorithm design techniques like Divide and Conquer, Greedy, Dynamic Programming, Approximation Algorithms, and Randomized Algorithms. Explore topics like Computational Complexity *etc*.

Introduction

- Computational Problems
- Algorithms
- Pseudo code
- Input size
- Analysis
  - Kind of resources[1]: time, space, number of gates ...
  - Cases: Best, Worst and Average
- Correctness: initialize well, maintain invariance and terminate
- Order of growth
- Insertion and Merge sort

[1] Complexity is a function

Insertion Sort

Incremental algorithm paradigm:

```
INSERTION-SORT(A)
1  for j = 2 to A.length
2      key = A[j]
3      // Insert A[j] into the sorted
           sequence A[1 .. j − 1].
4      i = j − 1
5      while i > 0 and A[i] > key
6          A[i + 1] = A[i]
7          i = i − 1
8      A[i + 1] = key
```

Insertion Sort

INSERTION-SORT(A)	*cost*	*times*
1 **for** $j = 2$ **to** $A.length$	c_1	n
2 $key = A[j]$	c_2	$n-1$
3 // Insert $A[j]$ into the sorted sequence $A[1..j-1]$.	0	$n-1$
4 $i = j - 1$	c_4	$n-1$
5 **while** $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
6 $A[i+1] = A[i]$	c_6	$\sum_{j=2}^{n} (t_j - 1)$
7 $i = i - 1$	c_7	$\sum_{j=2}^{n} (t_j - 1)$
8 $A[i+1] = key$	c_8	$n-1$

- Best case $T(n) = O(n)$
- Worst case $T(n) = O(n^2)$
- Average ?

Merge sort

Divide and conquer paradigm: Divide, Conquer and Combine

```
MERGE-SORT(A, p, r)
1  if p < r
2      q = ⌊(p + r)/2⌋
3      MERGE-SORT(A, p, q)
4      MERGE-SORT(A, q + 1, r)
5      MERGE(A, p, q, r)
```

```
MERGE(A, p, q, r)
 1  n1 = q − p + 1
 2  n2 = r − q
 3  let L[1 . . n1 + 1] and R[1 . . n2 + 1] be new arrays
 4  for i = 1 to n1
 5      L[i] = A[p + i − 1]
 6  for j = 1 to n2
 7      R[j] = A[q + j]
 8  L[n1 + 1] = ∞
 9  R[n2 + 1] = ∞
10  i = 1
11  j = 1
12  for k = p to r
13      if L[i] ≤ R[j]
14          A[k] = L[i]
15          i = i + 1
16      else A[k] = R[j]
17          j = j + 1
```

- T(n) = $\Theta(1)$ if $n \leq c$
- T(n) = $aT(n/b) + D(n) + C(n)$ otherwise
  - Average case $T(n) = O(\log n)$. Best and Worst?

Thank You!

Thank you very much for your attention!

Queries ?