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 07, 2020 (Campus @ BITS-Pilani Jan-May 2017)

Logistics: (CS F364) Design & Analysis of Algorithms

- T Th S (11-12PM) 5101@BITS-Pilani
- Jointly to be taught by
  Dr. Abhishek Mishra (IC) and **Dr. Kamlesh Tiwari**.
- Grading
  - Tutorial Quiz (32%) 4 of 8% each, Open Book
  - Mid Semester Exam (28%) Open Book
  - Comprehensive Exam (40%) Open Book

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

- **Books:**

[1] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, *Introduction to Algorithms*, 3rd Edition, PHI, 2012.

[2] S. Arora, B. Barak, Computational *Complexity: A Modern Approach*, Cambridge University Press, 2009

[3] J.Kleinberg, E. Tardos, *Algorithm Design*, Pearson, 2013.

Introduction

- Computational Problems

[1] Complexity is a function

Introduction

- Computational Problems
- Algorithms: input, output, definiteness, finiteness, effectiveness
- `Pseudo code`

[1] Complexity is a function

Introduction

- Computational Problems
- Algorithms: input, output, definiteness, finiteness, effectiveness
- `Pseudo code`
- Input size

[1] Complexity is a function

Introduction

- Computational Problems
- Algorithms: input, output, definiteness, finiteness, effectiveness
- `Pseudo code`
- Input size
- Analysis
  - **Kind of resources**[1]: time, space, number of gates ...
  - **Cases**: Best, Worst and Average

[1] Complexity is a function

Introduction

- Computational Problems
- Algorithms: input, output, definiteness, finiteness, effectiveness
- `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

[1] Complexity is a function

Introduction

- Computational Problems
- Algorithms: input, output, definiteness, finiteness, effectiveness
- `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: O, o, θ, ω, Ω zoo
- Insertion and Merge sort

[1] Complexity is a function

Insertion Sort

Incremental algorithm paradigm:

Algorithm 1: INSERTION-SORT (A)

```
for j = 2 to A.length do
    key = A[j]
    i = j − 1
    while i > 0 and A[i] > key do
        A[i + 1] = A[i]
        i = i − 1
    A[i + 1] = key
```

Analyse Insertion Sort

INSERTION-SORT (A)	cost	times
for $j = 2$ **to** $A.length$	c_1	n
$key = A[j]$	c_2	$n-1$
$i = j - 1$	c_3	$n-1$
while $i > 0$ **and** $A[i] > key$	c_4	$\sum_{j=2}^{n} t_j$
$A[i+1] = A[i]$	c_5	$\sum_{j=2}^{n}(t_j - 1)$
$i = i - 1$	c_6	$\sum_{j=2}^{n}(t_j - 1)$
$A[i+1] = key$	c_7	$n-1$

Analyse Insertion Sort

INSERTION-SORT (A)	cost	times
for $j = 2$ **to** $A.length$	c_1	n
$key = A[j]$	c_2	$n-1$
$i = j - 1$	c_3	$n-1$
while $i > 0$ **and** $A[i] > key$	c_4	$\sum_{j=2}^{n} t_j$
$A[i+1] = A[i]$	c_5	$\sum_{j=2}^{n}(t_j - 1)$
$i = i - 1$	c_6	$\sum_{j=2}^{n}(t_j - 1)$
$A[i+1] = key$	c_7	$n-1$

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

Merge sort

Divide and conquer paradigm: Divide, Conquer and Combine

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \leq c \\ aT(n/b) + D(n) + C(n) & \text{otherwise} \end{cases}$$

Merge sort

Divide and conquer paradigm: Divide, Conquer and Combine

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \le c \\ aT(n/b) + D(n) + C(n) & \text{otherwise} \end{cases}$$

MERGE-SORT (A,p,r)

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

Merge sort

Divide and conquer paradigm: Divide, Conquer and Combine

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \le c \\ aT(n/b) + D(n) + C(n) & \text{otherwise} \end{cases}$$

MERGE-SORT (A,p,r)

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

MERGE(A, p, q, r)

```
n1 = q − p + 1
n2 = r − q
let L[1 .. n1 + 1] and R[1 .. n2 + 1] be new arrays
for i = 1 to n1
    L[i] = A[p + i − 1]
for j = 1 to n2
    R[j] = A[q + j]
L[n1 + 1] = ∞
R[n2 + 1] = ∞
i = 1
j = 1
for k = p to r
    if L[i] ≤ R[j]
        A[k] = L[i]
        i = i + 1
    else A[k] = R[j]
        j = j + 1
```

Merge sort

Divide and conquer paradigm: Divide, Conquer and Combine

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \leq c \\ aT(n/b) + D(n) + C(n) & \text{otherwise} \end{cases}$$

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

```
MERGE(A, p, q, r)
  n1 = q − p + 1
  n2 = r − q
  let L[1 .. n1 + 1] and R[1 .. n2 + 1]
      be new arrays
  for i = 1 to n1
      L[i] = A[p + i − 1]
  for j = 1 to n2
      R[j] = A[q + j]
  L[n1 + 1] = ∞
  R[n2 + 1] = ∞
  i = 1
  j = 1
  for k = p to r
      if L[i] ≤ R[j]
          A[k] = L[i]
          i = i + 1
      else A[k] = R[j]
          j = j + 1
```

Average case $T(n) = O(n \log n)$. Best and Worst?

Thank You!

Thank you very much for your attention! (Reference[2])

Queries ?

[2] [1] Book - *Introduction to Algorithm*, By THOMAS H. CORMEN, CHARLES E. LEISERSON, RONALD L. RIVEST, CLIFFORD STEIN