Asymptotic complexity[1]

R. Inkulu
http://www.iitg.ac.in/rinkulu/

[1] only very essential notions are covered

RAM model of computation

- The machine independent abstract model is useful in measuring the run time of an algorithm by counting up the number of steps it takes.
- The *Random Access Machine (RAM)* model of computation is the commonly used abstract model:
  * each primitive operation takes constant time (ex. $+, -, *, /, ||, <, ==, =$ etc.,)
  * memory access takes constant time

Input, time and space complexities

- The time and space complexities are expressed in terms of the size of the input (*input complexity*); *asymptotic analysis* assumes that the input is large in size
- The *asymptotic time complexity* is the asymptotic number of units of time taken by a program on RAM model
- The *asymptotic space complexity* is the asymptotic number of bytes used by the program

  the *asymptotic auxiliary space (a.k.a. work space) complexity* is the asymptotic number of bytes used by the program excluding the input space complexity

$O()$ notation

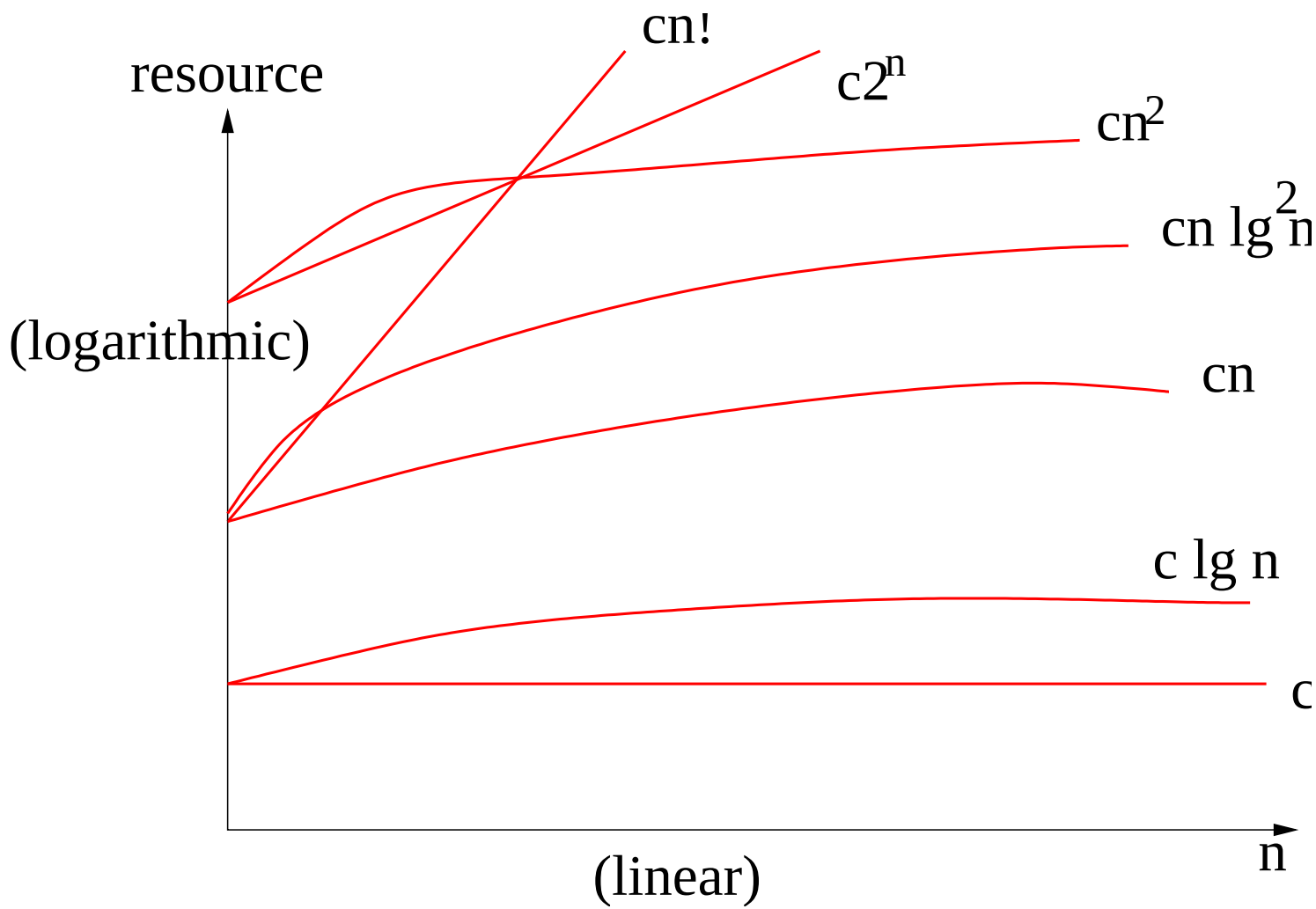
For any given input size n, how long an algorithm takes asymptotically "at most" is denoted with $O()$[2]: ignores constants and non-dominating terms from an expression. Examples -

- summing all the n numbers takes $O(n)$ time and $O(1)$ space
- given a pointer to an array containing n integers, summing the first and last numbers takes $O(1)$ time and $O(1)$ space
- ```
  for (int i=0; i<n; i++)
      for (int j=0; j<m; j++)
          sum += A[i][j];
  ```

  execution of the above nested loop is said to take $O(nm)$ time and $O(1)$ space

[2] precise definition of $O()$ notation is avoided
  ```

Comparing algorithms based on the asymptotic complexity



Time and space resources

- space reource is reusable whereas time is not
- space used by an algorithm is less than or equal to the time spent (excluding the input complexity)
- time-space tradeoffs

Worst-case asymptotic complexity

For every fixed asymptotically large n, if an algorithm A computes a solution to problem P for any input of size n in (resp. using) $O(f(n))$ time (resp. space), then $O(f(n))$ is said to be the *tight*[3] *worst-case asymptotic upper bound* on the time (resp. space) complexity[4] of A.

- Given a pointer to an array containing n integers in sorted order, binary search takes in worst-case $O(\lg n)$ time and $O(1)$ space.
- Bubble sort: $O(n^2)$ worst-case time and $O(1)$ worst-case space; takes $O(n)$ time in the best-case
- homework: analyze the worst-case and best-case asymptotic time and space complexities of selection sort and insertion sort algorithms for the sorting problem

[3] $O(n)$ time algorithm takes $O(n^3)$ as well but the latter is not a tight bound

[4] analogously, best-case and average-case resource complexities are defined