Chapter 2 analysis

Analyse Algorithm's performance

Completeness: all case take into considering

Optimality: it's an optimal solution

Time complexity: It takes not long to execute it

space complexity: How much memory is needed to perform such task

Polynomial-time

1. brute force:

   there is a natural brute force to search algorithm ==> to check all the possible solution

   typically takes 2^n time ==> unacceptable

2. desirable scaling property

   algorithm should only **slow down** by some constant factor **C**, we defautly say that it should be bound by cN^d steps

3. definition

   an algorithm is $poly-time$ if the above scaling property hold

worst-case analysis

1. worst-case running time:

   largest possible running time of algorithm on input of a given size N

2. average case running time:

   algorithm tuned for a certain distribution may perform poorly on toher inputs

3. worst-case polynomial-time

   definition: an alforithm is $efficient$ if its running time is polynomail-time

Asymptotic order of growth

1. Asymptotic Order of Growth

   upperbound:

   $$T(n)\ is\ O(f(n))\ if\ there\ exist\ constants\ c>0\ and\ n_0\geq 0 such\ that \\ for\ all\ n\geq n_0\ we\ have\ T(n)\leq c\cdot f(n).$$

   lowerbound:

   $$T(n)\ is\ \Omega(f(n))\ if\ there\ exist\ constants\ c>0\ and\ n_0\geq 0 such\ that \\ for\ all\ n\geq n_0\ we\ have\ T(n)\geq c\cdot f(n).$$

   tight bounds:

   $$T(n)\ is\ \Theta(f(n))\ if\ T(n)\ is\ O(f(n))\ and\ T(n)\ is\ \Omega(f(n))$$

2. Notation

   Slight abuse of notation. $T(n) = O(f(n))$

   Better notation: $T(n) \in O(f(n))$.

3. trasitivity

$$If\ f = O(g)\ and\ g = O(h)\ then\ f = O(h)$$
$$If\ f = \Omega(g)\ and\ g = \Omega(h)\ then\ f = \Omega(h)$$
$$If\ f = \Theta(g)\ and\ g = \Theta(h)\ then\ f = \Theta(h)$$

4. additivity

$$If\ f = O(h)\ and\ g = O(h)\ then\ f + g = O(h)$$
$$If\ f = \Omega(h)\ and\ g = \Omega(h)\ then\ f + g = \Omega(h)$$
$$If\ f = \Theta(h)\ and\ g = \Theta(h)\ then\ f + g = \Theta(h)$$

5. polynomials

$$a_0 + a_1 n + \ldots + a_d n^d\ is\ \Theta(n^d)\ if\ a_d > 0.$$

   logarithms

$$O(log_a n) = O(log_b n),\ \forall a,\ b > 0$$

$$\forall x > 0,\ log\, n = O(n^x)$$

   exponetials

$$\forall r > 1,\ n^d = O(r^n)$$

A Survey of Common Running Times

1. Linear time: $O(n)$

   1. $Computing\ the\ maximum$:

      Computing rhe maximum of n numbers $a_1, \ldots, a_n$

   2. $Merge$:

      combine two sorted list $A = a_1, \ldots, a_n$ with $B = b_1, \ldots, b_n$ into a sorted whole.

2. $O(logn)$ time: arise in divide-and-conquer algorithm

   1. $sorting$:

      Mergesort and heapsort are sorting algorithms that perform $O(logn)$comparisions.

   2. $Largest\ empty\ interval$:

      Given n time-stamps $x_1, \ldots, x_n$ on which copies of a file arrive at a server, what is largest interval of time when no copies of the file arrive?

      Sort the time-stamps. Scan the sorted list in order, identifying the maximum gap between successive time-stamps.

3. Quadratic time: $O(n^2)$

   $Closest\ pair\ of\ points$

Given a list of n points in the plane (x1, y1), ..., (xn, yn), find the pair that is closest

solution: Try all pairs of points.

Remark: $O(n^2)$ seems inevitable, but this is just an illusion.

4. Cubic time: $O(n^3)$

   $Set\ disjointness:$

   Given n sets $S_1, \ldots, S_n$ each of which is a subset of 1, 2, ..., n, is there some pair of these which are disjoint?

   solution: For each pairs of sets, determine if they are disjoint.

5. polymal time: $O(n^k)$

   $Independent\ set\ of\ size\ k:$

   Given a graph, are there k nodes such that no two are joined by an edge

   solution: Enumerate all subsets of k nodes.

   check whether S is an independent set: $O(k^2)$

   number of k element subsets: $O(n^k)$

   $\binom{n}{k} = \frac{n(n-1)...(n-k+1)}{k(k-1)...1} \leq \frac{n^k}{k!}$

   total complexity: $O(n^k)$

6. exponential time

   $maximum\ independent\ set:$

   Given a graph, what is maximum size of an independent set?

   solution: Enumerate all subsets. Then check any subset whether independent

   time complexity: $O(n^2\ 2^n)$