CLRS Solutions

aeslami

September 2024

Chapter 1

The Role of Algorithms in Computing

1.1 Section 1.1

1.1.1 Exercise 1.1-1

1. An example for sorting is sorting a list of Names in alphabetical order.

2. An Example for comptuting a convex hull is finding the diameter of a set of points

1.1.2 Exercise 1.1-2

1. Memory is another important resource. Another example is thermal energy produced by computers.

1.1.3 Exercise 1.1-3

1. Linked list is a example of a data structure. It's advantage is that an item can be added or removed in O(1) time. A disadvantage is that it doens't support random access.

1.1.4 Exercise 1.1-4

- The shortest path problem is mainly concerned about reaching a point from another point with the minimum cost, but the traveling salesman problem is about finding the shortest path that visits all of the points and returns to the origin point with the minimum cost.

1.1.5 Exercise 1.1-5

1. Finding the root of a polynomila is a example of a probelm that only best solution will do.

2. Finding a move in a game of Chess is a example of a problem that an "approximate solution" is good enough.

1.1.6 Exercise 1.2-1

In a navigation application, an algorithm is used to find the shortest path between two points.
$8 \times n^2 \leq 64 \times \lg n$
$n^2 \leq 8lgn$
$2 \leq n \leq 43$

1.1.7 Exercise 1.2-2

$n \leq 15$

1.1.8 Problem 1-1

	1 minute	1 hour	1 day	1 month	1 year	1 century
$\lg n$	$2^{60000000}$	$2^{3600000000}$	$2^{86400000000}$	$2^{2592000000000}$	$2^{31536000000000}$	$2^{3153600000000000}$
$\sqrt{n}$	3.6×10^{12}	1.296×10^{16}	7.46×10^{18}	6.72×10^{21}	9.95×10^{23}	9.95×10^{27}
n	6×10^{7}	3.6×10^{9}	8.64×10^{10}	2.592×10^{12}	3.1536×10^{13}	3.1536×10^{15}
$n \lg n$	2.8×10^{6}	1.3×10^{8}	2.0×10^{9}	4.9×10^{10}	5.4×10^{11}	3.9×10^{13}
n^2	7745966	60000000	293938769	1609968129	5615692821	56156922861
n^3	391420	1532278	4420825	13736056	31593173	146645033
2^n	25	31	36	41	44	51
$n!$	12	13	14	15	16	17

Table 1.1: Maximum size of n for different time complexities and time limits (assuming 1 operation per microsecond)

Chapter 2

Getting Started

2.1 Insertion Sort

The following is a Python implementation of the insertion sort algorithm. This code demonstrates the basic structure and functionality of insertion sort:

```
def insertion_sort(arr):
    """Perform insertion sort on the given array.
    Input: arr - A list of comparable elements
    Output: The same list, sorted in ascending order
    Time complexity: O(n^2), Space complexity: O(1)"""
    for j in range(1, len(arr)):
        key, i = arr[j], j - 1
        while i >= 0 and arr[i] > key:
            arr[i+1] = arr[i]
            i -= 1
        arr[i+1] = key
    return arr

def main():
    test_array = [5, 2, 4, 6, 1, 3]
    print("Original array:", test_array)
    sorted_array = insertion_sort(test_array)
    print("Sorted array:", sorted_array)

if __name__ == "__main__":
    main()
```