Introduction To Algorithms
HomeWork 4 Solutions

Junzhe Zheng

February 23, 2014

1.Give an example where quicksort requires $O(n^2)$ steps.

Consider a list:

$$10, 9, 8, 7, 6, 5, 4, 3, 2, 1$$

We choose the last digit in the last as the pivot.Thus time complexity is given as:

$$T(n) \ = \ T(n-1) + T(0) + \Theta(n)$$

By using substitution method, we could get:

$$T(n) \ = \ \Theta(n^2)$$

If it is $\Theta(n^2)$, it is also a $O(n^2)$.

2.Problem 4-6 (Page 110) CLRS(3rd Edition).

a. Need to prove "if and only if", thus the proof will have to separate parts

Proof of 'Only if':

If A is a Monge array, by definition, we have:

$$A[i,j] \ + A[k,l] \ < \ A[i,l] \ + A[k,j] \ \ \forall i \ , \ j \ , \ k \ , \ l$$

$$where \ 1 < i < k < n \ , \ 1 < j < l < m$$

Let $k = i + 1, \ l = j + 1$, we will have:

$$A[i,j] \ + A[i+1,j+1] \ < \ A[i,j+1] \ + A[i+1,j] \ \ \forall i \ , \ j$$

$$where \ 1 < i < i+1 < n \ , \ 1 < j < j+1 < m$$

$$where \ 1 < i < n-1 \ , \ 1 < j < m-1$$

‘Only if’ has been proved.

Proof of ‘if’:
Induction method will be used separately on rows and columns.
For rows:

3.Using the version of heap sort as defined in CLRS(chapter 6-4),show an example where heapsort requires Ω(nlogn) steps.

For an array that each elements are already sorted in an increasing order, the performance of heapsort is $\Omega(nlg(n))$. Because that each of the n-1 calls of Max_HEAPIFY (for i= A.length downto 2)takes $\Omega(lg(n))$.

4.Consider radix sort with numbers (using base 10) that are variable length. Show that you can output any number as soon as you have considered all its digits. Design a method to sort in O(n + k) time where k is the total number of digits in all the numbers.

For radix sort we start at checking the least digits. At the i^{th} digits sorting process, the number are ordered by the i^{th} least digits.
For a number with length j, once we finished checking j^{th} digits, we could output it in the right position in an sorted array.
To output a number, we need to check every number remain in to-be-sort list after l^{th} iteration. For a number with length i, if i=l, we output this number to sorted list. If i>l, then put this number to l+1 bucket for $l + 1^{th}$ iteration.
There are n number and k is the total length over all n digits. Thus the time complexity is $O(n + k)$.