Quiz 1

Started: 9 Apr at 9:40

Quiz instructions

Question 1 — 2 pts

Let f(n) be the running time of Karatsuba Algorithm for multiplying two n-digit integers. Which of the following is correct?

- ○ f(n) is in O(n)
- ◉ f(n) is in O(n^2)
- ○ f(n) is in O(log(n))
- ○ None of the above

Question 2 — 2 pts

Counting sort is faster than quicksort when it comes to sorting integers.

- ○ True
- ◉ False

Question 3 — 2 pts

If a function is in O(n^2+n), then it is also in O(n^2).

- (•) True
- ( ) False

Question 4 — 2 pts

What best describes the complexity of sorting 100 integers using counting sort?

- ( ) O(nlogn)
- ( ) O(n^2)
- ( ) O(n)
- (•) O(1)

Question 5 — 2 pts

Counting sort can only be applied to sorting integers.

- ○ True
- ◉ False

Question 6 — 2 pts

Which selection algorithm has better best-case performance?

- ◉ Randomized Selection
- ○ Median of Medians

Question 7 — 2 pts

Which selection algorithm has better worst-case performance?

- ○ Randomized Selection
- ◉ Median of Medians

Question 8 — 2 pts

You are asked to sort n English words alphabetically. What is the best achievable complexity? Note: we are talking about actual English words that can be found in a Marriam-Webster dictionary.

- (●) O(n)
- ( ) O(nlogn)
- ( ) O(n^2)

Question 9

4 pts

This is the second part of the previous question. Describe the algorithm that achieves the best complexity.

HTML Editor

The algorithm for achieving the best complexity is Radix Sort which is O(n). Also, MSD Radix Sort is the algorithm, I'll write in pseudo-code.

Algorithm: MSDRadixSort(*S*, *length*)

Input: S = {S_1, S_2,....,S_n} of strings and the length *length* of their common prefix.

Output: *S* in ascending order.

> ***if*** |*S*| < *n* ***then return*** StringQuickSort(*S*, *length)*
>
> S_t <--- {*String* ∈ *S* | | *String* | = *length*}; *S* <--- *S* \ S_t
>
> (S_1, S_2....S_{n-1}) <--- CouningSort(*S*, *length*)
>
> ***for*** *i* <--- 0 ***to*** *n* - 1 ***do*** S_i <--- MSDRadixSort(S_i, *length* + 1)
>
> ***return*** S_t *x* S_0 *x* S_1.....S_{n-1}

- Here, CounitngSort(*S*, *length*) returns the partitioning based on symbols at position *length*. Here the time complexity is O(n) for CouningSort.
- The pseudo-code for StringQuickSort is as follows:

if |*S*| ≤ 1 ***then return*** *S*
S_t <--- {*String* ∈ *S* | |*String*| = *length*}; *S* <--- *S* \ S_t
select pivot *p* ∈ *R*
$S_<$ <--- {*String* ∈ *S* | *String*[*length*] < p[*length*]}
$S_=$ <--- {*String* ∈ *S* | *String*[*length*] = p[*length*]}
$S_>$ <--- {*String* ∈ *S* | *String*[*length*] > p[*length*]}
$S_<$ <--- StringQuicksort($S_<$, *length*)
$S_=$ <--- StringQuicksort($S_=$, *length* + 1)
$S_>$ <--- StringQuicksort($S_>$, *length*)
return $S_t \cdot S_< \cdot S_= \cdot S_>$

p

220 words

Question 10

0 pts

What is your preferred format for assignment 2?

- No more reviews/reflection/peer reviews. I just want to be marked based on how many test cases I can pass on websubmission
- Stay the same as assignment 1

Question 11

0 pts

Please let me know whether you have any comments regarding the format of assignment 1.

HTML Editor

12pt Paragraph

The assignment 1 format was almost perfect except for certain things which are stated below:

The groups were assigned to the student of 4. During the discussion time, if the tutor or professor could have just watched the discussion quickly and appreciated the forum, that would help us to know that at least we are being watched by the mentor.

The assignment weightage was not balanced because 40% is a huge amount for a single submission, and it is heavily dependent on the discussion forum. If it will be reduced to 20% then it will benefit the students, because the assignment was tough and to pass the test cases it was challenging back then.

Overall, I'm happy with the format. No additional comments and I encourage if this could continue throughout the semester for all assignments as well.

p 138 words

Saved at 15:27 Submit quiz