HW 1 - Skyler Benjamin

Tuesday, September 7, 2021 2:22 PM

1. Algorithm Design 1.1, 1.2
    a. False
        i. In an instance of the Stable Matching Problem wherein there are two men and two women, w1 and m1 are engaged, and m2 and w2 are engaged. Both men are engaged to the woman they most prefer, while both women would actually prefer the other man. In this case, neither pair is with their highest ranking preference and no engagements will be broken because both men are happy enough not to propose to another woman.
    b. True
        i. This must be true based on the definition of stable matching, if two people are both the highest ranking on each others preference list, when they meet they will always end up together.

2. findMedian
    a. What quantity best represents the "input size" for this problem?
        - n, the number of terms in the list (length of list)
    b. Suppose the only primitive operations we are interested in counting are comparisons -- i.e., the number of times we compare two elements *in the list*.
    Given this, what is the **worst-case** running time T(n) of this algorithm? Justify your answer.
        - Each item in the list is compared to all items in the list (including itself) at least once, and at most twice if compared item is greater than or equal to current item.
        - Half of the list will always be smaller than the median, and thus have a greater overall number of "lessThan" counts (resulting in 1 comparison). While the other half of the list will always be greater than the median and thus have a greater overall number of "grtThan" counts (resulting in 2 comparisons). Because both halves are balanced and each item compare itself to half the elements once and the other half twice, resulting in 1.5n + 2 (for itself) comparisons at worst.

grtThan	0	4	5	2	6	1	3
	7	3	2	5	1	6	4
lessThan	6	2	1	4	0	5	3

        - grtThan ops = 21*2 = 42
        - lessThan ops = 21
        - Self ops = 7*2 = 14
        - Total ops = 42+21+14 = 77

        - T(n) = 1.5n^2 + 2
    a. What is the **best-case** running time T(n) of this algorithm? Justify your answer.
        - In the best case, instead of all elements running through the initial loop, the first index in the list is the median, then the loop will exit through the return in the if statement. In this case only 1 number is compared to an n length list with a perfect split of lessThan and grtThan counts, averaging to 1.5n operations + 2 for the self-comparison.
        - T(n) = 1.5n + 2

3. Text Reader
    a. What quantity, or quantities, best represent the "input size" for this problem? Think carefully, there are multiple ways of defining the input size for this problem.
        - n, the number of lines in argument file
    b. What is the **worst-case** running time T(n) of this algorithm? Justify your answer.
        - Each line iterates through the while loop once, each line is then split into t tokens (t denotes the max number of tokens possible in a given line), and each token is attempted to be inserted into the unique hashSet, then all unique tokens, q, are printed. In the worst case, all tokens are unique and must be printed
        - $T(n) = 3 + n(2 + t) + q$
    c. What is the **best-case** running time T(n) of this algorithm? Justify your answer.
        i. There is no difference in the best case running time because all operations are necessary and will be performed the same number of times for a given input n.
        ii. $T(n) = 3 + n(2 + t) + q$

4. Function Definition

A function T(n) is O(f(n)) if there exists constants c, n0 > 0 s.t. t(n) <= cf(n) for all n >= n0

5.
- $10^{10}*60*60$ = 3.6E13 = 36,000,000,000,000 op/hour
    a. n^2
        i. N = Sqrt(36,000,000,000,000) = 6,000,000
    b. n^3
        i. N = Cubrt(36,000,000,000,000) = 33,019
    c. $100n^2$
        i. N = sqrt(36,000,000,000,000)/100 = 60,000
    d. N log n
        i. ~~N = 6000000 * log.2 (6000000) = 135,099,186~~
        ii. ~~N = 60000000 * log.2 (60000000) = 1,550,307,549~~
        iii. ~~N = 600000000 * log.2 (600000000) = 17,496,232,355~~
        iv. ~~N = 6000000000 * log.2 (6000000000) = 194,893,892,128~~
        v. ~~N = 60000000000 * log.2 (60000000000) = 2,148,254,606,975~~
        vi. ~~N = 600000000000 * log.2 (600000000000) = 2.347E13~~
        vii. ~~N = 750000000000 * log.2 (750000000000) = 2.958E13~~
        viii. N = 900000000000 * log.2 (900,000,000,000) = 3.574E13
        ix. ~~N = 910000000000 * log.2 (910000000000) = 3.615E13~~

e. 2^n
    i. N = log.2 (36,000,000,000,000) = 45
f. 2^(2^n)
    i. N = "" = log.2 (45) = 5

Running Time (in ascending order)	Largest input size
N log n	9E11 < n < 9.1E11
N^2	6,000,0000
100n^2	60,000
N^3	33,019
2^n	45
2^(2^n)	5

Collaborated with Jack, Clark, and Dominic