Shuni Li
Comp221 HW2
Professor Shilad Sen
Feb 17, 2016

1. (10 pts) Solve each of the summations below, showing your work. For those who are not very familiar with this, the result should be a formula with no summations, no ellipses (...), just a typical polynomial-type formula. Make sure to use the known summations and sum manipulation rules that are given in Appendix A (uploaded to Moodle).
    a. $f_1(n) = \sum_{i=0}^{n}(5i^2 + i + 5)$

$$f_1(n) = \sum_{i=0}^{n}(5i^2 + i + 5) = \sum_{i=0}^{n} 5i^2 + \sum_{i=0}^{n} i + \sum_{i=0}^{n} 5 = 5\sum_{i=0}^{n} i^2 + \sum_{i=0}^{n} i + \sum_{i=0}^{n} 5$$

$$= 5(\tfrac{n^3}{3}) + \tfrac{n^2}{2} + 5(n+1) = \tfrac{5}{3}n^3 + \tfrac{1}{2}n^2 + 5n + 5$$

    b. $f_2(n,\ m) = \sum_{i=1}^{n} 3(\sum_{j=1}^{m}(4ij + 1))$

$$f_2(n,\ m) = \sum_{i=1}^{n}\sum_{j=1}^{m}(12ij + 3) = \sum_{i=1}^{n}\sum_{j=1}^{m} 12ij + \sum_{i=1}^{n}\sum_{j=1}^{m} 3 = \sum_{i=1}^{n} 12i\sum_{j=1}^{m} j + \sum_{i=1}^{n} 3m$$

$$= \sum_{i=1}^{n} 12i * (\tfrac{m^2}{2}) + \sum_{i=1}^{n} 3m = \sum_{i=1}^{n} 6m^2 * i + 3mn$$

$$= 6m^2\sum_{i=1}^{n} i + 3mn = 6m^2 * \tfrac{n^2}{2} + 3mn = 3m^2n^2 + 3mn$$

2. (10 pts) Complete question 6 in section 2.3, just parts a through d. This asks you to analyze this algorithm, determining the algorithm's purpose, a basic operation, the summation formula for the algorithm's efficiency, and its efficiency class, given in big-O or big-Theta notation.

a. What does this Algorithm compute?
   This algorithm determines if the input matrix is symmetric across the upper-left to lower-right diagonal. If the matrix is diagonal symmetric, return true; otherwise, return false.
b. What is its basic operation?
   The basic operation is **if** A[i, j] ≠ A[j, i] .
c. How many times is the basic operation executed?
   The basic operation is executed $\sum_{i=o}^{n-2}\sum_{j=i+1}^{n-1} 1$ times.
d. What is the efficiency class of this algorithm?

$$\sum_{i=o}^{n-2}\sum_{j=i+1}^{n-1} 1 = \sum_{i=o}^{n-2}(n-1)-(i+1)+1 = \sum_{i=o}^{n-2} n-i-1 = \sum_{i=o}^{n-2} n - \sum_{i=o}^{n-2} i - \sum_{i=o}^{n-2} 1 = n(n-1) + \tfrac{n^2-n}{2} + (n-1)$$

$$= n^2 - n + n^2/2 - n/2 + n - 1 = \frac{3n^2}{2} - \frac{n}{2} - 1 = \Theta(n^2)$$

This algorithm has time complexity of $\Theta(n^2)$.

3. (10 pts) Perform efficiency analysis on the recursive algorithm below. State your basic operation(s), set up a recurrence relation that captures the efficiency, and solve it, showing your work.

```
Algorithm MysteriousRecursion2(num)
1.      if num = 0 then
2.              return 1
3.      else if num is a multiple of 3 then
4.              return num * MysteriousRecursion2(num - 3)
5.      else
6.              return MysteriousRecursion2(num - 1)
```

The basic operation is **if** num = 0 (line 1).

$T(0) = 1$

$$T(n) = \begin{cases} 1 + T(n-3) & n \text{ is a multiple of } 3 \\ 1 + T(n-1) & \text{otherwise} \end{cases}$$

To solve this recurrence relation, we use back substitution method:
When n is a multiple of 3,

$$T(n) = 1 + T(n-3) \text{ (1st expansion)}$$
$$= 1 + 1 + T(n - 3 * 2) \text{ (2nd expansion)}$$
$$= i + T(n - 3 * i) \text{ (ith expansion)}$$
$$= \frac{n}{3} + T(n - 3 * \frac{n}{3}) \text{ (n/3)th expansion).}$$

So $T(n) = \frac{n}{3} + T(0) = \frac{n}{3} + 1 = \Theta(n)$.
When n is not a multiple of 3, we first need n mod 3 operations to convert n to a multiple of 3. Then it follows the recurrence showed above.

$$T(n) = n \bmod 3 + T(3 * \lfloor n/3 \rfloor) \text{ (0th expansion after conversion)}$$
$$= n \bmod 3 + 1 + T(3 * \lfloor n/3 \rfloor - 3) \text{ (1st expansion after conversion)}$$
$$= n \bmod 3 + i + T(3 * \lfloor n/3 \rfloor - 3i) \text{ (ith expansion after conversion)}$$
$$= n \bmod 3 + \lfloor n/3 \rfloor + T(0) \text{ (} \lfloor \text{n/3} \rfloor \text{ th expansion after conversion).}$$

So $T(n) = n \bmod 3 + \lfloor n/3 \rfloor + 1 = \Theta(n)$.
Therefore, the time complexity for this algorithm is $\Theta(n)$.

4. (10 pts) Complete questions 9 and 10 in section 3.3. Give the algorithm in nice pseudocode similar to the book's style, or mine.

Algorithm extreme(S[p1,p2…,pn])

```
pMaxX = S[p1]
PMinX = S[p1]

for i = 1 to n
    if S[pi].getXCoordinate >  pMaxX.getXCoordinate
    then pMaxX = S[pi]
    if S[pi].getXCoordinate < PMinX.getXCoordinate
    then pMinX = S[pi]

return pMaxX, PMinX
```

What modification needs to be made in the brute-force algorithm for the convex-hull problem to handle more than two points on the same straight line?
(Source: Zhenghan Zhang)

Algorithm convex(A[p1, p2, p3,...pn])

```
    A.sort by x coordinates;
    ArrayList uc = null; //upperconvex
    ArrayList lc = null;  //lowerconvex

    for i from 1 to n
        while ( lc.size() >= 2 &&
            (cross product of lc[lc.size() - 2] lc[lc.size() - 1] A[i] < 0 ||
            lc[lc.size() - 1] is midpoint of lc[lc.size() - 2] A[i] ))

                remove  lc[lc.size() - 1]
        add A[i] to lc

    for i from n to 1
        while ( uc.size() >= 2 &&
            (cross product of uc[uc.size() - 2] uc[uc.size() - 1] A[i] < 0 ||
            uc[uc.size()-1] is midpoint of uc[uc.size() - 2]A[i] ))

                remove  uc[uc.size() - 1]
        add A[i] to uc
    return lc and uc
```

Extra credit: Completing at most one of the following will earn up to 5 points of extra credit on this assignment.

- Look at question 10 in section 2.3. You can take Levitin's challenge of figuring out how to compute the sum in your head. But for this question, I want you to generalize the problem for any $n \times n$ table. Derive a formula in terms of n and analogous to the formula in question 9, that computes the sum of the numbers in the table, without needing any loop. Explain your work.

Basic idea: flip the n*n table across the upper-right and lower-left diagonal. Each entry becomes 2*n except diagonal entries, which are equal to n.

For diagonal entries, we have n entries and each is equal to n. Thus, in total we have n^2.

For other entries, we have $1 + 2 + ... + (n-1)$ entries and each is equal to $2n$. Thus, in total we have $[1 + 2 + ... + (n-1)] * 2n = \frac{n(n-1)}{2} 2n = n^3 - n^2$.

Therefore, the total sum is $n^3 - n^2 + n^2 = n^3$.