Introduction to Analysis of Algorithms Final
CSE4081

Eric Pereira

December 15, 2017

Contents

1 Problem 1

$$\sum_{i=0}^{n}\sum_{j=0}^{i}\sum_{k=0}^{j}1$$

$$\sum_{i=0}^{n}\sum_{j=0}^{i}j$$

$$\sum_{i=0}^{n}\frac{i(i+1)}{2}$$

$$\frac{1}{2}\sum_{i=0}^{n}i^2+i$$

$$\frac{1}{2}(\sum_{i=0}^{n}i^2+\sum_{i=0}i)$$

$$\frac{1}{2}(\frac{1}{6}n(n+1)(2n+1)+\frac{n(n+1)}{2})$$

$$\frac{n^3+3n^2+2n}{6}$$

2 Problem 2

What is being stated here is that $O(g(n)) = f(n)$ so that there exists constants, lets say c and n_0 so that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0$ and vice versa with $f(n) = O(g(n))$. $f(n) = O(g(n))$ means $f(n) \leq O(g(n))$.

3 Problem 3

As stated in problem 2, we will test out what was proven but square all the numbers. $O(g(n)^2) = f(n)^2$ so that there exists constants, lets say c and n_0 so that $0 \leq f(n)^2 \leq cg(n)^2$ for all $n \geq n_0$ and vice versa with $f(n)^2 = O(g(n)^2)$. $f(n)^2 = O(g(n)^2)$ means $f(n)^2 \leq O(g(n)^2)$.

4 Problem 4

For both problems an example with n being double will show how difficult it is to keep track when both numbers are doubled. The factor that 2^n increases when doubled is 2^n, compared to 2^3 when seen in the n^3 example. This shows that it is not tractable.

4.1 4a

Doubling input increases output by a factor of 2^3

$$f(n) = n^3$$
$$f(2n) = (2n)^3$$
$$f(2n) = 2^3 f(n)$$

lets say n = 10

$$f(10) = 10^3 = 1,000$$
$$f(20) = 20^3 = 8,000$$

4.2 4b

Doubling input increases output by a factor of 2^n

$$f(n) = 2^n$$
$$f(2n) = 2^{2n}$$
$$f(2n) = 2^n f(n)$$

lets say n = 10

$$f(10) = 2^{10} = 1,024$$
$$f(20) = 2^{20} = 1,048,576$$

5 Problem 5

$$T(n) = 3T(n/3) + n$$
$$T(n) \leq 3(c\frac{n}{3}\log_3\frac{n}{3} + n$$
$$T(n) = cn\log_3\frac{n}{3} + n$$
$$T(n) = cn\log_3 n - cn\log 33 + n$$
$$T(n) = cn\log_3 n - cn + n$$

6 Problem 6

$$T(n) = 3T(n/3) + n$$
$$T(n) = 3(3T(n/9) + n/3)) + n$$
$$T(n) = 9(3T(n/27) + 3n$$
$$T(n) = 27T(n/27) + 3n$$
$$T(n) = 3^iT(n/3^i) + i * n$$

$$\frac{n}{3^i} = 1 = i = \log_3 n$$

$$T(n) = 3^{log_3 n}T(1) + \log_3 n$$
$$T(n) = n + \log_3 n * n$$
$$T(n) = O(n)$$

7 Problem 7

$$G(z) = \sum_{n=0}^{\infty} t_n z^n$$

$$G(z) = \sum_{n=1}^{\infty} [t_{n-1} + 2^{n-1}]z^n$$

$$G(z) = \sum_{n=1}^{\infty} t_{n-1}z^n + \sum_{n=1}^{\infty} 2^{n-1}z^n$$

$$G(z) = z\sum_{n=0}^{\infty} t_n z^n + z\sum_{n=0}^{\infty} 2^z z^n$$

$$G(z) = zG(z) + \frac{z}{1-2z}$$

$$G(z) - zG(z) = \frac{z}{1-2z}$$

$$G(z)[1-z] = \frac{z}{1-2z}$$

$$G(z) = \frac{z}{(1-z)(1-2z)}$$

$$G(z) = \frac{1}{1-2z} - \frac{1}{1-z}$$

$$G(z) = \sum_{n=0}^{\infty} 2^n z^n - \sum_{n=0}^{\infty} z^n$$

$$G(z) = \sum_{n=0}^{\infty} (2^n - 1)z^n$$

8 Problem 8

8.1 8a

$$O(n + M)$$

8.2 8b

$$O(nM)$$

8.3 8c

I think there could be two greedy algorithms used for this problem. The first greedy algorithm could be to do the quickest jobs first, maximizing amount of chores done. The next greedy algorithm is to try and do chores that pay the best. The time complexity for each would be O(n), and these are terrible was to do the problem, they will not always produce an optimal solution.

9 Problem 9

9.1 9a

$$\binom{n}{k}$$

9.2 9b

$$\binom{k}{2}$$

9.3 9c

$$O(\binom{n}{k}\binom{k}{2})$$

9.4 9d

$$n * n!$$

9.5 9e

It is possible to use a nested for loop in order to find the independent amount of vertices. You could have one loop go through each k element subset and another loop go through sets within graph G. This would allow for a polynomial time algorithm, and still get the correct answer.