1. a. True

$\frac{n(n+1)}{2}$ grows no faster than n^3

b. False

$\frac{n(n+1)}{2}$ doesn't grow the same as n^3

c. True

$\frac{n(n+1)}{2}$ grows at least as fast as n

2.

a. $g(n) = 5n^7 + 14n^4 \log n + 3$

$\Theta(g(n)) = \Theta(n^7)$

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

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

3.

a. it counts the number that each element is less than other elements

b. This space is in space.

Because the input array doesn't require extra space.

c. n

d. The basic operation is

If $A[i] < A[j]$

$C[j] = C[j] + 1$

e

the worst case of $C(n)$ is n^2

f $\Theta(g(n)) = \Theta(n^2)$

4.

a.

this algorithm is arithmetic progression

b

$Q(1) = 1$

$Q(2) = Q(1) + 2 \times 2$

$Q(3) = Q(2) + 2 \times 3 = Q(1) + 2 \times 2 + 2 \times 3 = Q(1) =$

$Q(4) = Q(3) + 2 \times 4 = Q(1) + 2 \times 2 + 2 \times 3 + 2 \times 4 =$

$Q(n) = Q(n-1) + 2 \times n = Q(1) + 2(2 + 3 \cdots n)$

$n \geq 2$ $= Q(1) + 2(\sum_{i=1}^{n} n - 1)$

c.

the efficiency class of this algorithm is $\Theta(n^2)$

d. Use memoization to avoid redundant caculation

So it can improve the efficiency to $\Theta(n)$

5.

a. the algorithm will find the minimum element in the array.

b. Step 1

if array has only one element, return element

Step 2

Find the minimum element of subarray.

Step 3

Compare the minimum of array with the last element [n-1]. If mini is less than [n-1] return mini. otherwise, return [n-1

6. visited : h, c, d, b, g, f, a, e

stack : ~~h~~, ~~c~~, ~~d~~, ~~b~~, ~~g~~, ~~f~~, ~~a~~, ~~e~~

pop : a, f, g, b, d, e, c, h

Top sort: h, c, e, d, b, g, f, a

7.

n	m
~~86~~	~~35~~
43	70
21	140
~~10~~	~~280~~
5	560
~~2~~	~~1120~~
1	2240

$70 + 140 + 560 + 2240 = 3010$

$n \times m = 86 \times 35 = 3010$ ※

8.

oppen oppen

48 45 40 35 30 25 20 15 10 5

a. I should take 3 coins and I should keep coins as above number

b. No. I can't win If the opponent play the first player.

9. Suppose there are n cities in this state

Step 1. Divide these cities into two subsets by $X = C$

So half of these cities lie to the left and the right.

Step 2. Find the closest pairs for the lef and the right subsets. d_1 for the left

However, $d = \min(d_1, d_2)$ might not be the shortest. d_2 for the right.

the closer of points can lie on the opposite sides separting the line.

Step 3. Check the boundary points

$$T(n) = 2T\left(\frac{n}{2}\right) + f(n)$$

$$f(n) \in O(n)$$