Homework 1

1.1 6a. gcd (31415, 14142) $\quad m = 31415 \quad n = 14142$

$$
\begin{aligned}
\gcd(31415, 14142) &= \gcd(14142, 3131) && r = 3131 \\
&= \gcd(3131, 1618) && r = 1618 \\
&= \gcd(1618, 1513) && r = 1513 \\
&= \gcd(1513, 105) && r = 105 \\
&= \gcd(105, 43) && r = 43 \\
&= \gcd(43, 19) && r = 19 \\
&= \gcd(19, 5) && r = 5 \\
&= \gcd(5, 4) && r = 4 \\
&= \gcd(4, 1) && r = 1 \\
&= \gcd(1, 0) && r = 0 \\
&= \boxed{1}
\end{aligned}
$$

10

b. $\min\{31415, 14142\}$ $\quad t = 14142$

$14142 \le$ # of iterations $\le 2 * 14142$ since gcd of 31415 and 14142 is 1 and t has to divide possible m and n to see if it is the gcd for the numbers

Thus Euclid's algorithm is at least $14142/10 \approx 1414$ times faster and at most $28282/10 \approx 2828$ times faster than consecutive integer checking.

12. n lockers 1...n initially closed n passes

$n = 10$

	1	2	3	4	5	6	7	8	9	10
1	C	C	C	C	C	C	C	C	C	C
2	O	O	O	O	O	O	O	O	O	O
3	O	C	O	C	O	C	O	C	O	C
4	O	C	C	O	O	O	O	C	C	C
5	O	C	C	O	C	O	O	C	C	O
6	O	C	C	O	C	C	O	C	C	O
7	O	C	C	O	C	C	C	C	C	O
8	O	C	C	O	C	C	C	C	C	O
9	O	C	C	O	C	C	C	C	O	O
10	O	C	C	O	C	C	C	C	O	C

– 3 open (1, 4, 9) perfect squares

After the last pass, $\lfloor \sqrt{n} \rfloor$ locker doors are open and $n - \lfloor \sqrt{n} \rfloor$ doors are closed. All ith doors that are perfect squares are open and rest are closed.

$2+1+5+1$ $\qquad$ $10+1+5+1$

1.2 2. person 1: 1 min, person 2: 2 min, person 3: 5 mins, person 4: 10 mins
17 minutes for all to cross

1, 2, 3, 4	∅	
3, 4	1, 2	2
1, 3, 4	2	3
1	2, 3, 4	13
1, 2	3, 4	15
	1, 2, 3, 4	17

4. $ax^2 + bx + c = 0$ find real roots pseudocode

```
Algorithm roots of quadratic equation (a,b,c)
    // Input: arbitrary real coefficients a, b, c
    // Output: real roots for equation

    if a ≠ 0
        temp = b*b - 4*a*c
        if temp > 0,
            x1 = (-b + sqrt(temp))/(2*a)
            x2 = (-b - sqrt(temp))/(2*a)
            return x1, x2
        else if temp = 0
            return -b/(2*a)

        else
            return "no real roots"
    else
        if b ≠ 0
            return -c/b
        else if b = 0 & c = 0
            return "all real numbers"
        else if b = 0
            return "no real roots"
```

4 3 2 1 0
1 0 1 1

19 9 1
9 4 1

4 2 0
2 1 0
1 0 1

5a.
1. Set the decimal number to variable N
2. Create a string str to hold binary representation of decimal number.
3. Repeat next following steps until N becomes 0.
4. Divide N by 2. Set remainder to R and quotient to Q
5. Add R to str (right to left), making it the next digit in binary number
6. Assign Q to N, making the quotient the new decimal number.

5b. Algorithm Binary(N)
// Input: positive decimal integer N
// Output: binary representation of N
$i \rightarrow 0$
while $N \neq 0$
$str_i \leftarrow N \bmod 2$
$i \leftarrow i + 1$
$N \leftarrow \lfloor \frac{N}{2} \rfloor$
return str

1.3 1a. A: 60, 35, 81, 98, 14, 47

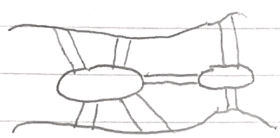
count: 0, 0, 0, 0, 0, 0
i=0, count: 3, 0, 1, 1, 0, 0
i=1, count: 1, 2, 2, 0, 1
i=2, count: 4, 3, 0, 1
i=3, count: 5, 0, 1
i=4, count: 0, 2
final count: 3, 1, 4, 5, 0, 2

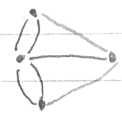
S: 14, 35, 47, 60, 81, 98

1b. This algorithm is not stable, because the algorithm will not preserve the relative order of two equal elements. When it comes down between comparing the two equal elements, the first one in the unsorted array will be incremented by 1 and thus be later in the array when sorted.

1c. Not in-place, because it uses two arrays of size n and S.

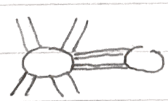
4a.

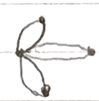




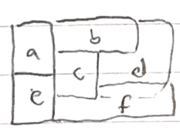
Here is a multigraph. The problem is that we have to find a path were all edges are traversed exactly once and reaches back to the starting vertex.

b. This problem does not have a solution because some vertices have an odd number of edges connected to them. There would be a solution if all vertices had an even number of edges connected to vertices. It would take a total of one more extra bridge to make such a stroll possible.



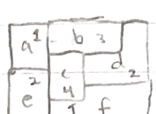


8a.



We can use graph coloring ~~by~~ breaking up the object ~~into~~ into different vertices and color those vertices making sure that no two adjacent vertices have the same color.

b.



The smallest number of colors is 4.

2.1 1a. Computing the sum of n numbers
natural size for its inputs: n
basic operation: addition of the two number
basic operation count: cannot be different for inputs of same size

d. Euclid's algorithm
natural size for its inputs: Size of larger of two input numbers
or size of smaller of two input numbers
or sum of sizes of two input numbers
basic operation: mod division.
basic operation count: can be different for inputs of same size.

7a. $T(n) \approx C_{op} C(n)$ Gaussian elimination algorithm: $\frac{1}{3}n^3$ multiplications

$C(n) = \frac{1}{3} n^3$

$T(500) \approx C_{op} C(500) \approx C_{op} \frac{1}{3}(500)^3$
$T(1000) \approx C_{op} C(1000) \approx C_{op} \frac{1}{3}(1000)^3$

$\frac{T(1000)}{T(500)} \approx \frac{C_{op} \frac{1}{3}(1000)^3}{C_{op} \frac{1}{3}(500)^3} = \frac{T(2n)}{T(n)} \approx \frac{C_{op} \frac{1}{3}(2n)^3}{C_{op} \frac{1}{3}(n)^3}$

$= 2^3 = \boxed{8}$

Gaussian elimination will run 8 times longer on a system of 1000 equations compared to a system of 500 equations.

b. $T(n) = 1000\, C_{op} \frac{1}{3} n^3$
$T(N) = C_{op} \frac{1}{3} N^3$

$T(n) = T(N)$
$1000 C_{op} \frac{1}{3} n^3 = C_{op} \frac{1}{3} N^3$
$1000 = \frac{N^3}{n^3}$
$1000 = \left(\frac{N}{n}\right)^3$
$\sqrt[3]{1000} = \frac{N}{n}$
$\frac{N}{n} = 10$

By a factor of 10, the faster computer increases the sizes of systems solvable in same amount of time as the old one.

9a. $n(n+1)$ and $2000n^2$

$f(n) = n(n+1) \approx n^2$, thus the pair of functions have the same order of growth within a constant multiple

c. $\log_2 n$ and $\ln n$

$\log_a n = \log_a b \cdot \log_b n$

All logarithmic function have same order of growth to within a constant multiple.

Thus the pair of log functions have the same order of growth to within a constant multiple,

e. 2^{n-1} and 2^n

$f_1(n) = 2^{n-1} = 2^n 2^{-1} = \frac{1}{2} 2^n$

$f_2(n) = 2^n$

Thus the pair of functions has the same order of growth to within a constant multiple.

2.2 3b. $\sqrt{10n^2+7n+3}$

$\sqrt{10n^2+7n+3} \approx \sqrt{10n^2} \approx \sqrt{10}\, n$ $\quad \sqrt{10}\, n \in \Theta(n)$

$$\lim_{n\to\infty} \frac{\sqrt{10n^2+7n+3}}{n} = \lim_{n\to\infty} \sqrt{\frac{10n^2+7n+3}{n^2}} = \lim_{n\to\infty} \sqrt{10 + \frac{7}{n} + \frac{3}{n^2}} = \sqrt{10}$$

Thus $\sqrt{10n^2+7n+3} \in \Theta(n)$

c. $2n \lg (n+2)^2 + (n+2)^2 \lg \frac{n}{2}$

$= 2n\, 2 \lg (n+2) + (n+2)^2 (\lg n - \log_2 2)$

$= 4n \lg (n+2) + (n+2)^2 (\lg n - 2\log_2 2)$

$4n \lg (n+2) + (n+2)^2 (\lg n - 1) \in \Theta(n \lg n) + \Theta(n^2 \lg n) \in \Theta(n^2 \lg n)$

Therefore $2n \lg (n+2)^2 + (n+2)^2 \lg \frac{n}{2} \in \Theta(n^2 \lg n)$

6a. $p(n) = a_k n^k + a_{k-1} n^{k-1} + \dots + a_0$ with $a_k > 0$

$$\lim_{n\to\infty} \frac{a_k n^k + a_{k-1} n^{k-1} + \dots + a_0}{n^k} = \lim_{n\to\infty} \frac{a_k n^k}{n^k} + \frac{a_{k-1} n^{k-1}}{n^k} + \dots + \frac{a_0}{n^k}$$

$$= \lim_{n\to\infty} a_k + \frac{a_{k-1}}{n} + \dots + \frac{a_1}{n^{k-1}} + \frac{a_0}{n^k}$$

$$= a_k + 0 + \dots + 0$$

$$= a_k$$

Thus any polynomial belongs to $\Theta(n^k)$

b. a_i, a_j

$\lim_{n\to\infty} \frac{a_i^n}{a_j^n}$ using properties of limits

$a_i^n < a_j^n$: $\lim_{n\to\infty} \frac{a_i^n}{a_j^n} = \lim_{n\to\infty} \left(\frac{a_i}{a_j}\right)^n = 0$ thus $a_i^n \in o(a_j^n)$

$a_i^n = a_j^n$: $\lim_{n\to\infty} \frac{a_i^n}{a_j^n} = \lim_{n\to\infty} \left(\frac{a_i}{a_j}\right)^n = 1$ thus $a_i^n \in \Theta(a_j^n)$

$a_i^n > a_j^n$: $\lim_{n\to\infty} \frac{a_i^n}{a_j^n} = \lim_{n\to\infty} \left(\frac{a_i}{a_j}\right)^n = \infty$ thus $a_j^n \in o(a_i^n)$

Therefore the order of growth of an exponential function a^n vary base a on the values of base $a > 0$.

2.3 1a. $1 + 3 + 5 + 7 + \dots + 999$

$$\sum_{i=1}^{500} 2i - 1 = \sum_{i=1}^{500} 2i - \sum_{i=1}^{500} 1$$

$$2\,\frac{500 \times 501}{2} - 500$$

$$500 \times 501 - 500$$

$$250500 - 500$$

$$= \boxed{250000}$$

d. $$\sum_{i=3}^{n+1} i = \sum_{i=0}^{n+1} i - \sum_{i=0}^{2} i$$

$$\frac{(n+1)(n+2)}{2} - 3$$

$$\boxed{\frac{n^2+3n-4}{2}}$$

e. $\sum_{i=0}^{n-1} i(i+1) = \sum_{i=0}^{n-1} i^2 + i = \sum_{i=0}^{n-1} i^2 + \sum_{i=0}^{n-1} = \frac{(n-1)\cdot n\cdot(2n-1)}{6} + \frac{(n-1)\cdot n}{2} = \boxed{\frac{(n^2-1)\cdot n}{3}}$

g. $\sum_{i=1}^{n}\sum_{j=1}^{n} ij = \sum_{i=1}^{n} i \sum_{j=1}^{n} j = \frac{n(n+1)}{2}\frac{n(n+1)}{2} = \boxed{\frac{n^2(n+1)^2}{4}}$

2a. $\sum_{i=0}^{n-1} (i^2+1)^2 = \sum_{i=0}^{n-1} i^4 + 2i^2 + 1 = \sum_{i=0}^{n-1} i^4 + \sum_{i=0}^{n-1} 2i^2 + \sum_{i=0}^{n-1} 1$

$= \frac{n(n-1)(2n-1)(3n^2-3n-1)}{30} + 2\frac{n(n+1)(2n+1)}{6} + (n-1-0+1)$

$= \frac{n(n-1)(2n-1)(3n^2-3n-1)}{30} + \frac{1}{3}n(n+1)(2n+1) + n$

$\approx \frac{1}{30}n^{4+1} + \frac{1}{3}n^3 + n$

$= \Theta(n^5) + \Theta(n^3) + \Theta(n)$

$\approx \boxed{\Theta(n^5)}$

Therefore $\sum_{i=0}^{n-1} (i^2+1)^2 \in \Theta(n^5)$

b. $\sum_{i=2}^{n-1} \lg i^2 = \sum_{i=2}^{n-1} 2\lg i = 2\sum_{i=1}^{n} \lg i - 2\log_2 n$ because $\sum_{i=1}^{n} \lg i \approx n\log n$

$\in \Theta(n\log_2 n) - \Theta(\log_2 n)$

$\in \Theta(n\log_2 n)$

$\boxed{\text{Thus } \sum_{i=2}^{n-1} \lg i^2 \in \Theta(n\log_2 n)}$

4a. This algorithm computes the sum of squares for n numbers, $S(n) = \sum_{i=1}^{n} i^2$

b. The basic operation of the algorithm is multiplication

c. Since one multiplication is done each loop, therefore the basic operation ~~u~~number of times executed is $C(n) = \sum_{i=1}^{n} 1$, thus the basic operation is executed n times

d. The efficiency class of the algorithm is $C(n) = \sum_{i=1}^{n} 1 = n \in \Theta(n)$

e. To improve efficiency class, we can use $S(n) = \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$ to compute the sum in $\Theta(1)$ time