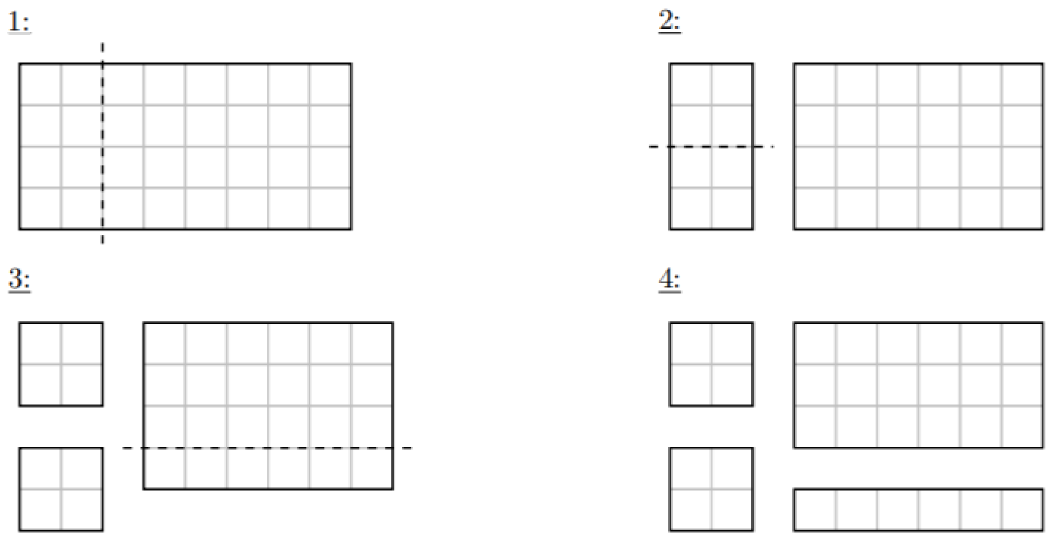
Claire Minahan

Homework 6-10

Homework 6 – Recursive Definitions

Due: Thursday, February 18th

1. (5.2.10) Suppose that a rectangular chocolate bar consists of n individual squares. Further suppose that the bar — or any smaller, rectangular, piece of the bar — can only be broken into two pieces along the horizontal or vertical lines separating its squares.
   For example, one possible rectangular bar of 32 squares is that with dimensions 8×4. One possible way to break this bar 4 times is as follows:

   … producing two bars of 4 squares, one bar of 18, and one bar of 6:
   (a) How many breaks must be made in order to break the bar into its n constituent squares — that is, into *n* bars of 1 square each? **31 → n-1**
   (b) Using a proof by induction, prove the above answer.

Proof:

***Basis Step*:**

Let n = 1. Then 1-1 = 0 and certainly if there is only one square then it will take 0 breaks to get it into 1 square.

***Inductive Hypothesis*:**

Let k be some integer. If there are k squares, then it will take k-1 breaks to get the bars into square each.

***Inductive Step*:**

Let n = k + 1. Let k + 1 = $x*y$ such that x and y are some integers and there are the width and height of the bar. Breaking off part of it will then lead to a $(x - 1) \times y$ bar and a $1 \times y$ bar. Because we proved that k is true, then certainly a $(x\text{-}1) \times y$, and we then need $(x\text{-}1) \times (y\text{-}1)$ breaks to break the $(x\text{-}1) \times y$. Since $y \times 1$ will take y-1 breaks, it will then take $(x\text{-}1)*y - 1$

+ (y-1) + 1 breaks to break the $x \times y$ bar. So that is then $xy - y - 1 + y - 1 + 1 = xy - 1 = k + 1 - 1$.

Therefore, by PMI, for all positive integers, it takes n – 1 breaks in order to break the bar into its n constituent squares.

2. (2.4.9) Find the first five terms of each of the following recurrence relations:
   (a) $a_n = 6a_{n-1}$, where $a_0 = 2$
   2, 12, 72, 864, 10368}
   (b) $a_n = a_{n-1}^2$, where $a_1 = 2$
   {$\sqrt{2}$, 2, 4, 16, 256}
   (c) $a_n = a_{n-1} + 3a_{n-2}$, where $a_0 = 1$ and $a_1 = 2$
   {1, 2, 5, 11, 26}
   (d) $a_n = a_{n-1} + a_{n-3}$, where $a_0 = 1$, $a_1 = 2$, and $a_2 = 0$.
   {1, 2, 0, 1, 3}

3. (2.4.16) Solve each of the following recurrence relations:
   (a) $a_n = -a_{n-1}$, where $a_0 = 5$
   $\boldsymbol{a_n = -a_{n-1}}$
   $\boldsymbol{= -1 * -a_{n-2}}$
   $\boldsymbol{= -1 * -1 * -a_{n-3}}$
   …
   $\boldsymbol{= (-1)^n a_0}$ $\boldsymbol{= (-5)^n}$
   (b) $a_n = a_{n-1} - n$, where $a_0 = 4$
   $\boldsymbol{a_n = a_{n-1} - n}$
   $\boldsymbol{= a_{n-2} - (n - 1) - n}$
   $\boldsymbol{= a_{n-3} - (n - 2) - (n - 1) - n}$
   $\boldsymbol{= a_{n-3} - [(n - 2) + (n - 1) + n]}$
   …
   $\boldsymbol{= a_0 - [\Sigma_{i=1}^{n} i]}$ $\boldsymbol{= 4 - [n(n+1) / 2]}$
   (c) $a_n = 2a_{n-1} - 3$, where $a_0 = -1$
   $\boldsymbol{a_n = 2a_{n-1} - 3}$
   $\boldsymbol{= 2*2a_{n-2} - 3 - 3}$
   $\boldsymbol{= 2*2*2a_{n-2} - 3 - 3 - 3}$
   …
   $\boldsymbol{= 2^n a_0 - 3n}$ $\boldsymbol{= 2^n(-1) - 3n}$
   (d) $a_n = (n + 1)\, a_{n-1}$, where $a_0 = 2$
   $\boldsymbol{a_n = (n + 1)\, a_{n-1}}$
   $\boldsymbol{= (n+1)*(n)*a_{n-2}}$
   $\boldsymbol{= (n+1)*(n)*(n-1)*a_{n-2}}$
   …
   $\boldsymbol{= (n+1)!\, a_0}$ $\boldsymbol{= 2(n + 1)!}$

4. (5.3.7) Give a recursive definition for each of the following sequences:

(a) (a_n), where $a_n = 6n$
***Basis Step*: $a_0 = 0$**
***Recursive Step*: $a_n = a_{n-1} + 6$**
(b) (a_n), where $a_n = 2n + 1$
***Basis Step*: $a_0 = 1$**
***Recursive Step*: $a_n = a_{n-1} + 2$**
(c) (a_n), where $a_n = 10^n$
***Basis Step*: $a_0 = 1$**
***Recursive Step*: $a_n = 10a_{n-1}$**
(d) (a_n), where $a_n = 5$
***Basis Step*: $a_0 = 5$**
***Recursive Step*: $a_n = a_{n-1}$**

5. (5.3.24 and 5.3.25) Give a recursive definition for each of the following sets:
(a) The set of positive integer powers of 3
***Basis Step*: $1 \in A$**
***Recursive Step*: if $x \in A$, then $3x \in A$**
(b) The set of single-variable polynomials with integer coefficients
***Basis Step*: $1 \in A$**
***Recursive Step:* $x + y \in A$ if $x \in A$ and $y \in A$**
$x - y \in A$ if $x \in A$ and $y \in A$
$x * y \in A$ if $x \in A$ and $y \in A$
(c) The set of even integers
***Basis Step*: $2 \in A$**
***Recursive Step*: if $x \in A$ then $(x + 2) \in A$ and $(x - 2) \in A$**
(d) The set of positive integers not divisible by 5
***Basis Step*: $1, 2, 3, 4 \in A$**
***Recursive Step*: if $x \in A$ then $(x + 5) \in A$**

6. (5.3.14) Let f_n denote the n^{th} Fibonacci number. Prove the following statement:

For all integers $n \geq 1$, $f_{n+1} f_{n-1} - f_n^2 = (-1)^n$

[*Hint:* In the inductive step, consider applying the hypothesis to both sides of the equation.]

Proof:

***Basis Step*:**

Let n = 1. Then we have $f_2 f_0 - f_1^2 = (-1)^1 = (1)(0) - 1 = -1$.

***Inductive Hypothesis*:**

Let k be some integer. Then we have $f_{k+1} f_{k-1} - f_k^2 = (-1)^k$

***Inductive Step*:**

Let n = k + 1. Then we have $f_{(k+1)+1} f_{(k+1)-1} - f_{k+1}^2 = f_{k+2} f_k - f_{k+1}^2 =$

Therefore, by PMI, for all integers $n \geq 1$, $f_{n+1} f_{n-1} - f_n^2 = (-1)^n$

7. (5.3.15) Let f_n denote the n^{th} Fibonacci number. Prove the following statement:
   For all integers $n \geq 1$, $f_0 f_1 + f_1 f_2 + \ldots + f_{2n-2} f_{2n-1} + f_{2n-1} f_{2n} = f_{2n}^2$

Proof:

***Basis Step*:**

Let n = 1. Then we have $f_0 f_1 + f_1 f_2 = f_{2(1)}^2 = 0*1 + 1*1 = (1)^2 = 1$.

***Inductive Hypothesis*:**

Let k be some integer. $f_0 f_1 + f_1 f_2 + \ldots + f_{2k-2} f_{2k-1} + f_{2k-1} f_{2k} = f_{2k}^2$

***Inductive Step*:**

Let n = k + 1. Then we have $f_0 f_1 + f_1 f_2 + \ldots + f_{2k-2} f_{2k-1} + f_{2k-1} f_{2k} + f_{2(k+1)-1} f_{2(k+1)}$. By hypothesis, this is equal to $f_{2k}^2 + f_{2k} f_{2k+1} + f_{2k+1} f_{2k+2} = f_{2k}(f_{2k} + f_{2k+1}) + f_{2k+1} f_{2k+2} = f_{2k} f_{2k+2} + f_{2k+1} f_{2k+2} = (f_{2k} + f_{2k+1}) f_{2k+2} = f_{2k+2}^2$

Therefore, by PMI, for all integers $n \geq 1$, $f_0 f_1 + f_1 f_2 + \ldots + f_{2n-2} f_{2n-1} + f_{2n-1} f_{2n} = f_{2n}^2$

8. (5.3.28) Let A be a set of ordered pairs of integers defined by:
   Basis Step: $(0, 0) \in A$
   Recursive Step: If $(a, b) \in A$, then $(a + 2, b + 3) \in A$ and $(a + 3, b + 2) \in A$.
   Prove the following statement:
   If $(a, b) \in A$ after n applications of the recursive step, then $5 \mid a + b$

Proof:

***Basis Step*:**

Let n = 0 and a = 0 and b = 0. (0, 0) ∈ A and certainly, 5 | 0+0.

***Inductive Hypothesis*:**

Suppose, for all integers, 1 ≤ n ≤ k, if (a, b) ∈ A after n applications of the recursive step, then 5 | *a* + *b*.

***Inductive Step*:**

Let n = k + 1. Then (a, b) ∈ A and (a+2, b+3) ∈ A. This then leads to (a+2) + (b+3) = a + b + 5 and since a + b is divisible by 5, as proven in the hypothesis, and 5 is divisible by 5, then we have 5 | (a+2) + (b+3)

We also have (a+3), (b+2) ∈ A. This then leads to (a+3) + (b+2) = a + b +5 which we proved above to be divisible by 5.

Therefore, by PMI, if (*a*, b) ∈ *A* after *n* applications of the recursive step, then 5 | *a* + *b*

9. (5.3.45) The height of a full binary tree is defined recursively as follows:
Basis Step: A single node is a full binary tree with height $h = 0$.
Recursive Step: If T_1 and T_2 are disjoint full binary trees with heights h_1 and h_2, respectively, then the full binary tree consisting of a root r with T_1 as its left subtree and T_2 as its right subtree has height $h = 1 + \max\{h_1, h_2\}$.
Prove the following statement:
Let T be a full binary tree. If T has n total nodes and height h, then $n \geq 2h + 1$

Proof:

***Basis Step*:**

Let n = 1. If n = 1 then the height must be 0, so 2h + 1 = 2(0) + 1 = 1, and certainly 1 ≥ 1.

***Inductive Hypothesis*:**

Let k be some integer such that 1 ≤ n ≤ k. Then n ≥ 2*h* + 1.

***Inductive Step*:**

Let n = k + 1. Because k ≥ 1 and h ≥ 1. This then means that T has a root, r, with two non-empty subtrees, T_1 and T_2 with height h_1 and h_2 and n_1 and n_2 nodes in each.

Then, $h = 1 + \max\{h_1, h_2\}$, thus, $n_1 \geq 2h_1 + 1$ and $n_2 \geq 2h_2 + 1$.

This then means that $n_1 + n_2 \geq (2h_1 + 1) + (2h_2 + 1)$

$$n_1 + n_2 + 1 \geq (2h_1 + 1) + (2h_2 + 1) + 1$$

Let $n = n_1 + n_2 + 1$, which leads to $n \geq 2(1 + \max\{(2h_1 + 1), (2h_2 + 1)\}) + 1$, so then $h = 1 + \max\{(2h_1 + 1), (2h_2 + 1)\}$ and by our hypothesis, this equals $n \geq 2h + 1$.

Therefore, by PMI, if *T* has *n* total nodes and height *h*, then *n* ≥ 2*h* + 1.

Homework 7 – Algorithm Correctness

Due: Thursday, February 25th

1. (3.1.4) Consider the largest difference between consecutive elements in a finite sequence of integers. For example, given the sequence:

   (8, 2, 9, 5, 1)

   … the largest difference between consecutive elements is |2 – 9| = 7.
   Give pseudocode for a recursive algorithm that finds the largest difference between consecutive elements in a sequence.

LARGEST DIFFERENCE (A ← (a_1, a_2, …, a_n))

Input: a finite sequence of integers

Output: the largest difference between consecutive elements

if n = 1:

then return 0

if n = 2:

then return $|a_1 – a_2|$

else, do:

return max{$|a_1 – a_2|$, LARGESTDIFFERENCE((a_2, a_3, …, a_n))}

2. (3.1.6) Consider the number of negative integers in a finite sequence of integers. For example, given the sequence:

   (-1, 0 , -8, 17, 3)

   … the number of negative integers is 2.
   Give pseudocode for a recursive algorithm that finds the number of negative integers in a sequence.

NUM NEGATIVES (A ← (a_1, a_2, …, a_n))

Input: anon-empty, finite sequence of integers

Output: the number of negative integers in that sequence

if n = 1 and n < 0:

return 1

if a < 0, then:

return (1 + NUMNEGATIVES((a_2, …, a_n))

else, do:

return NUMNEGATIVES((a_2, …, a_n))

3. (3.1.8) Consider the index of the largest even integer in a finite sequence of distinct integers. For example, given the sequence:

$$(a_1, a_2, a_3, a_4, a_5) = (5, -19, 2, 8, 1)$$

…the index of the largest even integer is 4.
Give pseudocode for a recursive algorithm that finds the index of the largest even integer in a sequence, returning −1 if there exist no even integers.

MAX INDEX (A ← (a_1, a_2, …, a_n))

Input: an integer, i, that starts at 1, and a finite sequence of integers

Output: the index of the largest integer

if n = 1 and a_1 is even:

return 1

if a is even:

return max{a, MAXINDEX(a_2,

4. (3.1.9) Consider a palindrome: a string that reads the same forwards and backwards, a string that is equal to itself when reversed. For example, given the sequence:

`(a, b, c, b, a)`

…the characters in the above sequence form a palindrome, whereas given the sequence:

`(a, b, c, a, b)`

…these characters do not.
Give pseudocode for a recursive algorithm that determines whether or not the characters in a sequence form a palindrome.

PALINDROME (A ← (a_1, a_2, …, a_n))

Input: characters in a sequence

Output: true or false

if n = 1 or n = 0:

return True

else, if a_1 = a_n:

return PALINDROME(A ← (a_2, a_3, …, a_{n-1}))

else:

return False

5. (5.4.8) Consider the sum of the first n positive integers. For example, given n = 4:

$$\Sigma_{i=1}^{4} i = 1 + 2 + 3 + 4$$

…the sum of the first 4 positive integers is 10.

(a) Give pseudocode for a recursive algorithm that computes the sum of the first n positive integers.

SUM POSITIVE(n)

Input: a finite amount of integers, an integer x that starts at 0, and an integer i = 1.

Output: the sum of the first n positive integers

if n = 1:

return 1

return n + SUMPOSITIVE(n-1)

(b) State a lemma that you would need to prove in order to show the correctness of your algorithm.

Let n be a number of positive integers. If SUMPOSITIVE(n) returns n!, then n is some integer.

(c) Prove that your algorithm is correct.

Proof: Let |A| be denoted n.

***Basis Step*:**

Let n = 1. Then A = (a_1) = 1. Certainly the sum the first positive integer is 1.

***Inductive Hypothesis*:**

Suppose, for all integers 1 ≤ n ≤ k, if SUMPOSITIVE(n) returns n!, then n is some integer.

***Inductive Step*:**

Let n = k+1. Since k ≥ 1, n ≥ 2. SUMPOSITIVE recurses on n, which is the size of ≤ k.

By the hypothesis, SUMPOSITIVE obtains some value such that n is some positive integer.

6. (5.4.11) Consider the minimum element of a finite sequence of integers. For example, given the sequence:

$$(0, -2, 9, 24, 6)$$

…the minimum element is −2.

(a) Give pseudocode for a recursive algorithm that finds the minimum of a finite sequence of integers.

MINELEMENT(A ← (a_1, a_2, …, a_n))

Input: a finite, non-empty, sequence of numbers

Output: the minimum element of A

if n = 1, then:

return a_1

else, do:

return MINELEMENT(($\min\{a_1, a_2\}$, (a_3, …, a_n))

(b) State a lemma that you would need to prove in order to show the correctness of your algorithm.

Let A be a finite, non-empty sequence of integers. If MINELEMENT(A) returns x, then $x \in A$ and $\forall a \in A\ (x \geq a)$.

(c) Prove that your algorithm is correct.

Proof: Let |A| be denoted n.

***Basis Step*:**

Let n = 1. Then A = (a_1). Certainly, $a_1 \in A$ and $a_1 \geq a_1$. Thus, MINELEMENT correctly returns a_1.

***Inductive Hypothesis*:**

Suppose, for all integers $1 \leq n \leq k$, if A is a finite sequence of n integers and MINELEMENT(A) returns x, then $x \in A$ and $\forall a \in A\ (x \geq a)$.

***Inductive Step*:**

Let n = k+1. Since $k \geq 1$, $n \geq 2$. Let (a_2, a_3, …, a_n) be denoted A'. Then MINELEMENT recurses on A', which is of size $\leq$ k.

By the hypothesis, MINELEMENT obtains some value such that $x \in A'$ and $\forall a \in A'\ (x \geq a)$.

If $a_1 \leq x$, then $\forall a \in A'\ (a_1 \leq a)$, and certainly $a_1 \leq a_1$. Thus $\forall a \in A\ (a_1 \leq a)$. Further, $a_1 \in A$. Thus MINELEMENT correctly returns a_1.

If $a_1 > x$, since $\forall a \in A'\ (x < a)$, then $\forall a \in A(x < a)$. Further, since $x \in A'$, $x \in A$. Thus, MINELEMENT correctly returns x.

In all possible cases, MINELEMENT returns x such that $x \in A$ and $\forall a \in A\ (x \geq a)$.

Therefore, by the PMI, MINELEMENT is correct. □

7. (5.4.15) Consider the greatest common denominator of two positive integers, a and b. Note that if $a = b$, then $\gcd(a, b) = a$. Furthermore, if $a < b$, then $\gcd(a, b) = \gcd(a, b - a)$. For example, given $a = 8$ and $b = 12$:

$$\gcd(8, 12) = \gcd(4, 8) = \gcd(4, 4) = 4$$

…the greatest common denominator of 8 and 12 is 4.

(a) Give pseudocode for a recursive algorithm that computes the greatest common denominator of two positive integers.

GCD(a, b)

Input: two positive integers, a and b

Output: the greatest common denominator of a and b

if a = b:

return a

else if a = 0, then:

return b

else if a = b – a, then:

return a

else if a < b – a, then:

return GCD(a, b-a)

else, do:

return GCD(b-a, a)

(b) State a lemma that you would need to prove in order to show the correctness of your algorithm.

Let a and b be positive integers. If GCD(a, b) returns x, then gcd(a, b) = x.

(c) Prove that your algorithm is correct.

Proof:

***Basis Step*:**

Let a = b. Then gcd(a, b) = a. Certainly if a = b then the greatest common denominator is the integer a itself.

***Inductive Hypothesis*:**

Suppose,

8. (5.4.37) Consider the *reversal* of a finite sequence of arbitrary elements. For example, given the sequence:

$$(a_1, a_2, a_3, a_4, a_5)$$

…the reversal is $(a_5, a_4, a_3, a_2, a_1)$.

(a) Give pseudocode for a recursive algorithm that reverses a sequence.

REVERSE(A $\leftarrow$ **(**$\mathbf{a_1, a_2, \ldots, a_n}$**))**

Input: a non-empty, finite sequence of arbitrary elements

Output: the reverse of a sequence

if n ≤ 1, then:

return ($\mathbf{a_1}$**)**

else, do

return (a_n), REVERSE(a_1, a_2, …, a_n)

(b) State a lemma that you would need to prove in order to show the correctness of your algorithm.

Let A be a non-empty, finite sequence of arbitrary elements. If REVERSE(A) returns B, then B ∈ A and, |A| = |B|, and ∀a ∈ A(a_i = n_{n+1-i} ∈ B).

(c) Prove that your algorithm is correct.

Proof: Let |A| be denoted n.

***Basis Step*:**

Let n = 1. Then A = (a_1). Certainly a_1 ∈ A and an + 1 – 1 = a_1. Thus, REVERE correctly returns a_1.

***Inductive Hypothesis*:**

Suppose, for all integers 1 ≤ n ≤ k, if A is a finite, non-empty sequence of arbitrary elements and REVERE(A) returns B, then B ∈ A and, |A| = |B|, and ∀a ∈ A(a_i = b_{n+1-i} ∈ B).

Inductive Step:

Let n = k+1. Since k ≥ 1, n ≥ 2. Let the (a_2, …, a_n) be denoted A'. Then REVERE recurses on A', which is of size ≤ k. By the hypothesis, REVERSE obtains some value such that B ∈ A' and, |A'| = |B|, and ∀a ∈ A'(a_i = b_{n+1-i} ∈ B).

If a_1 is then added on, then REVERE will return a_1, as stated in the basis step, at the end of the sequence to be returned. Thus, REVERE correctly returns the reversal of its input.

In all possible cases, REVERSE returns B such that B is the reverse of A.

Therefore, by the PMI, REVERE is correct.

Homework 8 – Algorithm Complexity

Due: Thursday, March 4th

1. (3.2.1 and 3.2.28) Determine whether each of the following functions is $O(x)$, $\Omega(x)$, or both (i.e., $\Theta(x)$):
   (a) $f(x) = 10$ **$O(x)$**
   (b) $f(x) = 3x + 7$ **$\Theta(x)$**
   (c) $f(x) = x^2 + x + 1$ **$\Omega(\mathrm{x})$**
   (d) $f(x) = 5 \log x$ **$O(x)$**

2. (3.2.12) Prove the following statement:
   Let $f(x) = x \log_2 x$ and $g(x) = x^2$ be real functions. Then $f(x)$ is $O(g(x))$, but $g(x)$ is not $O(f(x))$.

   $f(x)$ → the smallest Big-O estimate is $O(\mathrm{x}^2)$ and $g(x)$ is x^2

   $g(x)$ → the smallest Big-O estimate is $O(\mathrm{x}^2)$ but not $O(\mathrm{x} \log_2 \mathrm{x})$

3. (3.2.25 and 3.2.26) Give the smallest Big-O estimate (i.e., the Big-Θ estimate) for each of the following functions:
   (a) $(n^2 + 8)(n + 1)$
   $O(\mathrm{n}^2)$
   (b) $(n! + 2^n)(n^3 + \log_2(n^2 + 1))$
   $O(\mathrm{n}^3\ \mathrm{n}!)$
   (c) $(n^3 + n^2 \log_2 n)(\log_2 n + 1) + (17 \log_2 n + 19)(n^3 + 2)$
   $O(\mathrm{n}^3 \mathrm{logn})$
   (d) $(n^n + n2^n + 5^n)(n! + 5^n)$
   $O(\mathrm{n}^\mathrm{n}\mathrm{n}!)$

4. Using the tree method, solve each of the following recurrence relations. Where possible, verify the answer using the Master Theorem.
   (a) $T(n) = T(^n/_2) + O(n^2)$
   $O(\mathrm{n}^2)$
   (b) $T(n) = 3T(^n/_3) + O(n)$
   $O(\mathrm{nlogn})$
   (c) $T(n) = 2T(^n/_3) + O(1)$
   $O(\mathrm{n}^{\log}3^2)$
   (d) $T(n) = 2T(n - 2) + O(1)$
   $O(\log \mathrm{n})$

5. (3.2.15) Explain what it would mean for an algorithm to have complexity $O(1)$.

All functions for which there exist real numbers k and C with $|f(x)| \leq C$ for $x > k$. These are the functions $f(x)$ that are bounded for all sufficiently large x.

6. (3.2.36) Explain what it would mean for an algorithm to have complexity $\Omega(1)$.

***f*(*x*) cannot get closer to 0 than some fixed bound, when *x* is sufficiently large.**

7. (3.2.37) Explain what it would mean for an algorithm to have complexity Θ (1).

If *f*(*x*) is *Θ*(1), then |*f*(*x*)| is bounded between positive constants C_1 and C_2. In other words, *f*(*x*) cannot grow larger than a fixed bound or smaller than the negative of this bound and must not get closer to 0 than some fixed bound.

8. (3.3.26) Recall that a binary search exploits the ordered nature of a sorted sequence.
   (a) Consider the following recursive algorithm for searching a sorted sequence:

$\textsc{TernarySearch}(x, A \leftarrow (a_1, a_2, \ldots, a_n))$

Input: An integer, x, and a finite, non-empty, sorted sequence of integers, A
Output: Whether or not x is an element of A

let $i \leftarrow \lceil n/3 \rceil$ and $j \leftarrow \lceil 2n/3 \rceil$
if $a_i = x$ **or** $a_j = x$ **then**
    return T
else if $n \leq 2$ **then**
    return F
else if $x < a_i$ **then**
    return $\textsc{TernarySearch}(x, (a_1, a_2, \ldots, a_i))$
else if $a_i < x < a_j$ **then**
    return $\textsc{TernarySearch}(x, (a_{i+1}, a_{i+2}, \ldots, a_j))$
else
    return $\textsc{TernarySearch}(x, (a_{j+1}, a_{j+2}, \ldots, a_n))$

Set up and a solve a recurrence relation giving a Big-O estimate for the complexity of this algorithm.

T(n) = 1 * T(n/3) + *O*(1)

Using the Master Theorem:

a = 1, b = 3, d= 0

$\log_b a = \log_3 1 = 0 = d$

***O*(log n)**

(b) Compared to a binary search, would any algorithm that attempted to divide the sequence into *b* > 2 subsequences offer any further improvement in time complexity?

No, because no matter how many times b is increased, it will always lead to $\log_b 1 = 0$ which will always lead back to a time complexity of O(log n).

9. (8.3.19) Recall that there is often a tradeoff between *time complexity* and *space complexity*.

(a) Consider the following recursive algorithm for computing x^n:

```
NaïvePower(x, n)
Input: A real number x and a positive integer n
Output: x^n
if n = 1 then
    return x
else if n is even then
    return NaïvePower(x, n/2) · NaïvePower(x, n/2)
else
    return x · NaïvePower(x, ⌊n/2⌋) · NaïvePower(x, ⌊n/2⌋)
```

Set up and a solve a recurrence relation giving a Big-*O* estimate for the complexity of this algorithm.

T(n) = T(n/2) + O(1)

(b) In contrast, consider the following recursive algorithm for computing x^n:

```
SmartPower(x, n)
Input: A real number x and a positive integer n
Output: x^n
if n = 1 then
    return x
else
    let x' ← SmartPower(x, ⌊n/2⌋)
    if n is even then
        return x' · x'
    else
        return x · x' · x'
```

Set up and solve a recurrence relation giving a Big-*O* estimate for the complexity of this algorithm.

T(n) = T() + O(logn)

(c) Compared to NaïvePower, how much additional space does SmartPower require?

Compared to NaïvePower, Smart power requires log n additional space

Homework 9 – Permutations and Combinations

Due: Thursday, March 11th

1. (6.1.1) Suppose there are 18 mathematics majors and 325 computer science majors at a university.
   (a) How many ways can 2 students be picked, a mathematician and a computer scientist?
   18 * 325 = <u>5850</u>
   (b) How many ways can 1 student be picked, either a mathematician or a computer scientist?
   18 + 325 = <u>343</u>

2. (6.1.3) Suppose a multiple-choice test contains 10 questions, each with 4 possible answers.
   (a) How many ways can a student answer every question on the test?
   4^{10} = <u>1048576</u>
   (b) How many ways can a student answer every question if they may leave answers blank?
   5^{10} = <u>9765625</u>

3. (6.1.11) How many bit strings of length 10 both begin and end with "1"?
   $(1) * (2^8) * (1)$ = <u>256</u>

4. (6.1.12) How many bit strings are there of length 6 or less, not counting the empty string?
   <u>½</u> <u>½</u> <u>½</u> <u>½</u> <u>½</u> <u>½</u> $= 2^6 +$
   <u>½</u> <u>½</u> <u>½</u> <u>½</u> <u>½</u> $= 2^5 +$
   <u>½</u> <u>½</u> <u>½</u> <u>½</u> $= 2^4 +$
   <u>½</u> <u>½</u> <u>½</u> $= 2^3 +$
   <u>½</u> <u>½</u> $= 2^2 +$
   <u>½</u> $= 2^1$
   $= 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2 =$ <u>126</u>

5. (6.1.41) How many bit strings of length n are palindromes?
   If n is even → $2^{n/2}$ **If n is odd → $2^{(n+1)/2}$**

6. (6.1.16) The English alphabet consists of 26 letters.
   (a) How many strings are there of 4 lowercase letters?
   26^4
   (b) How many strings of 4 lowercase letters *do not* include the letter 'x'?

25^4

(c) How many strings of 4 lowercase letters *do* include the letter ‘x’?

$26^4 – 25^4$ = <u>66351</u>

7. (6.1.42) A *DNA sequence* consists of the *bases* ‘A’, ‘C’, ‘G’, and ‘T’.
   (a) How many DNA sequences of length 4 do not include the base ‘T’?

   3^4 = <u>81</u>

   (b) How many DNA sequences of length 4 include the subsequence ‘ACG’?

   4 + 4 = <u>8</u>

   (c) How many DNA sequences of length 4 include all 4 bases?

   4*3*2*1 = <u>24</u>

   (d) How many DNA sequences of length 4 include exactly 3 of the 4 bases?

   3*3*2*1 * 4 = <u>72</u>

8. (6.1.49) Suppose that there are 6 people in a wedding party, including the bride and groom.
   (a) How many ways can they be arranged for a picture such that the bride is next to the groom?

   (5 * 4!) * 2 = <u>240</u>

   (b) How many ways can they be arranged for a picture such that the bride is not next to the groom?

   6! – 240 = 780 – 240 = <u>480</u>

   (c) How many ways can they be arranged for a picture such that the bride is to the left of the groom?

   6! / 2 = 720/2 = <u>360</u>

9. (6.2.3) Suppose there are 24 socks, of which 12 are brown and 12 are black. How many socks must be chosen at random to guarantee 2 socks of the same color?

   3

10. (6.2.5) Suppose that each student at a university has one of 4 expected graduation years and one of 21 majors. How many students must be enrolled to guarantee 2 graduations in the same year and major?

    85

11. (6.2.8) Suppose that there are *n* + 1 integers, not necessarily consecutive, where *n* is a positive integer. How many integers have the same remainder when divided by *n*?

    $\lceil n + 1 / n \rceil > 1$ but $< 2 \rightarrow$ <u>2</u>

12. (6.2.21) Suppose that there are 25 students in a class, each either a freshman, a sophomore, or a junior. How many students must be in the same cohort?

    $\lceil n / k \rceil$, n = 25 and k = 3 $\rightarrow \lceil 25 / 3 \rceil$ = <u>8</u>

13. (6.3.3) How many permutations of the set {*a*, *b*, *c*, *d*, *e*, *f*, *g*} end with the element *a*?

6! = <u>720</u>

14. (6.3.12) How many bit strings of length 12 contain an equal number of ‘0’s and ‘1’s?

P(12, 6) = <u>924</u>

15. (6.3.22) How many permutations of the string “ABCDEFGH” contain the strings “CAB” and “BED”?

CABED, F, G, H → 4 possible options

P(4, 4) = <u>24</u>

16. (6.3.33) The English alphabet consists of 21 consonants and 5 vowels.

(a) How many strings of 6 lowercase letters contain exactly 2 vowels?

6nCr2 * 5^2 * 21^4 = 15 * 5^2 * 21^4 = <u>72930375</u>

(b) How many strings of 6 lowercase letters contain at least 2 vowels?

(6nCr2)(5^2*21^4) + (6nCr3)(5^3*21^3) + (6nCr4)(5^4*21^2) + (6nCr5)(5^5*21) + 5^6

= <u>100626625</u>

17. (6.3.40) How many ways can 12 countries be selected to serve on a United Nations council if 3 are selected from a block of 45, 4 are from a block of 57, and the others from the remaining 91 countries?

45nCr3 = 14190

57nCr4 = 395010

12 – 4 – 3 = 5 → 91nCr5 = 46504458

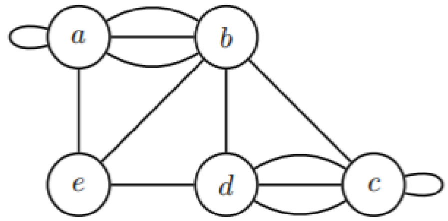
Using the product rule, 14190 * 395010 * 46504458

= <u>2.6067*10^{17}</u>

Homework10 – Graphs

Due: Tuesday, March 16th

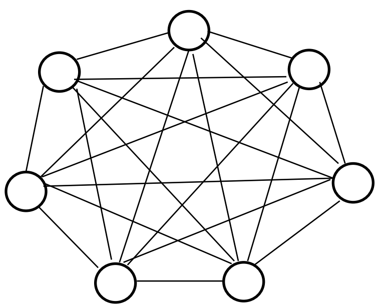
1. (10.2.2) Find the degree of each vertex in the graph below. Find the sum of the degrees and verify that it equals twice the number of edges.



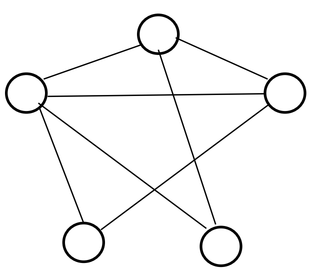
$V_a = 4$, $V_b = 6$, $V_c = 4$, $V_d = 5$, $V_e = 3$ → 22

Edges = 11

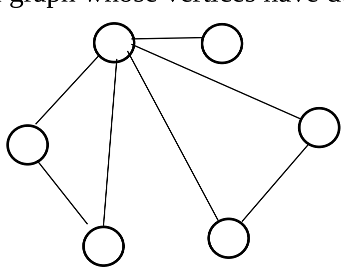
2. (10.2.20) Draw K_7, the complete graph on 7 vertices.



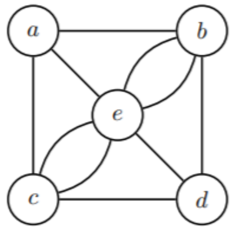
3. (10.2.40) Draw a graph whose vertices have degrees 4, 3, 3, 2, and 2.



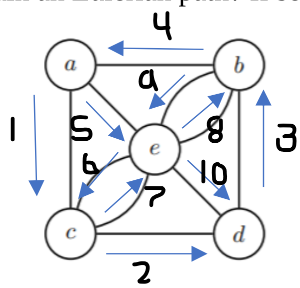
4. (10.2.41) Draw a graph whose vertices have degrees 5, 2, 2, 2, 2, and 1.



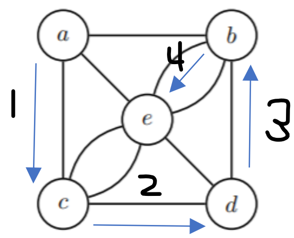
5. (10.5.3) Consider the following graph:



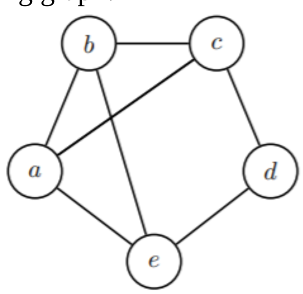
(a) Does this graph contain an Eulerian path? If so, construct one.



(b) Does this graph contain a Hamiltonian path? If so, construct one.



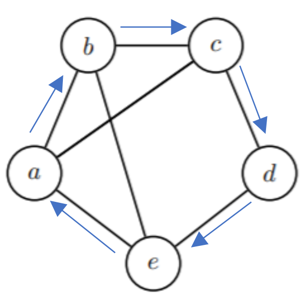
6. (10.5.1) Consider the following graph:



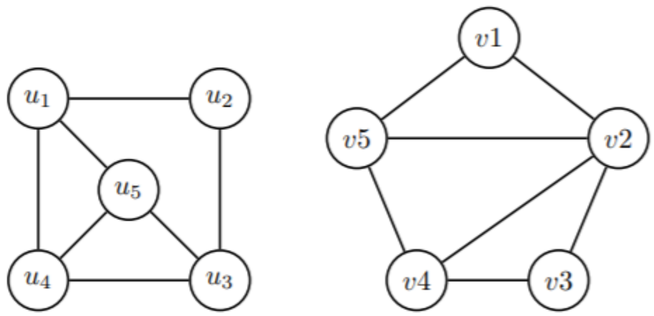
(a) Does this graph contain an Eulerian path? If so, construct one.

No

(b) Does this graph contain a Hamiltonian path? If so, construct one.

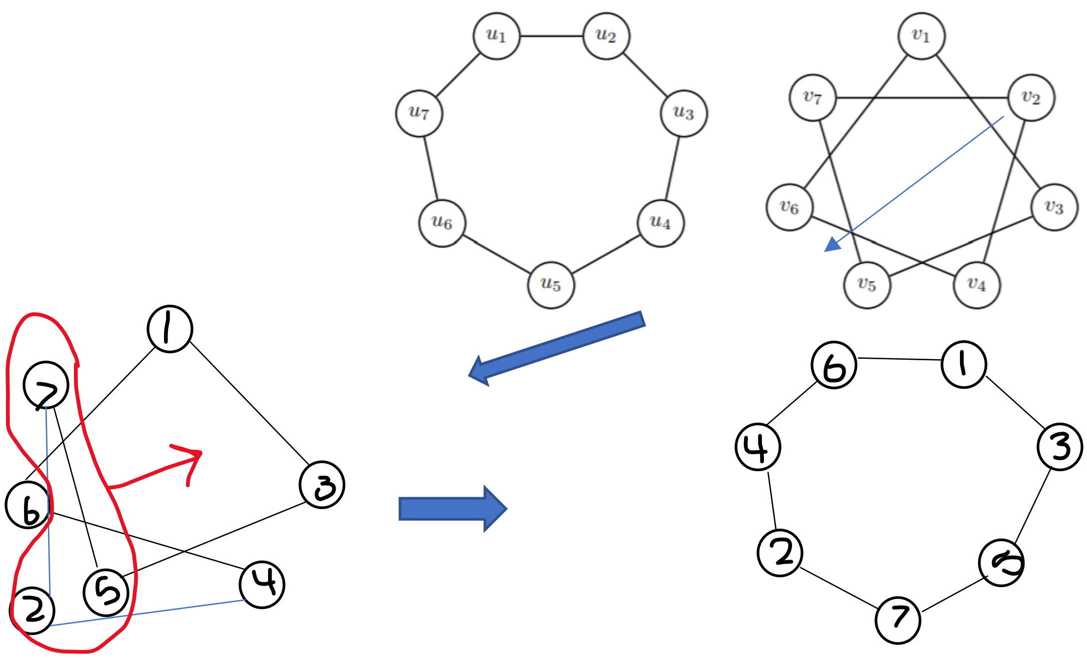


7. (10.3.36) Determine whether or not the graphs below are isomorphic. If so, provide an isomorphism; if not, briefly explain why.



They are not isomorphic because v_2 has degree 4 but not of the vertices in graph U have a degree 4.

8. (10.3.37) Determine whether or not the graphs below are isomorphic. If so, provide an isomorphism; if not, briefly explain why.



f **: $V_u \rightarrow V_v$ = {(u_1, v_6), (u_2, v_1), (u_3, v_3), (u_4, v_5), (u_5, v_7), (u_6, v_2), (u_7, v_4)}**

9. (10.3.48) Determine whether or not the graphs below are isomorphic. If so, provide an isomorphism; if not, briefly explain why. **Not isomorphic because they already have the same shape and their edges don't match.**