You can use this as a study guide. You will also be able to use it on the Final Exam on Tuesday. If there’s anything else you feel should be on this, please send me email before Monday night.

(1) Sequences and Summations

(a) Fibonacci sequence (1, 1, 2, 3, 5, 8, 13, ⋯):

Recursive Definition

(i) **Base Cases:** \( f_1 = f_2 = 1 \)

(ii) **Recursive definition:** \( f_n = f_{n-1} + f_{n-2} \)

Closed Form Definition

\[
f_n = \frac{(1 + \sqrt{5})^n - (1 - \sqrt{5})^n}{2^n \sqrt{5}}
\]

(b) Some summations:

(i) \( 1 + 2 + 3 + 4 + \cdots + n = \frac{n^2 + n}{2} \)

(ii) \( 1^2 + 2^2 + 3^2 + 4^2 + \cdots + n^2 = \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} \)

(iii) \( 9 + 9^2 + 9^3 + \cdots + 9^n = \frac{9^{n+1} - 9}{8} \)

(2) Proof by Induction

(a) Proof that \( 1 + 2 + 3 + \cdots + n = \frac{n^2 + n}{2} \)

(i) **Base case:** \( 1 = \frac{1^2 + 1}{2} = 1 \)
(ii) **Inductive step:** Assume that $1 + 2 + 3 + \cdots + i = \frac{i^2 + i}{2}$ for $i$ from 1 to $n - 1$.

$$
\begin{align*}
1 + 2 + 3 + \cdots + n &= (1 + 2 + 3 + \cdots + n - 1) + n \\
&= \frac{(n - 1)^2 + (n - 1)}{2} + n \\
&= \frac{n^2 - n}{2} + n \\
&= \frac{n^2 - n + 2n}{2} \\
&= \frac{n^2 + n}{2}
\end{align*}
$$

(3) **Recursion**

(a) Kevin Bacon Actors’ Guild recursive definition (where $KB$ is Kevin Bacon and $A(x, y)$ means that $x$ and $y$ acted in a movie together):

$$
KBAG = \{ x | x = KB \lor \exists y (y \in KBAG \land A(x, y)) \}
$$

(b) The set $MA$ of all Marc’s ancestors (excluding Marc) is given by the following definition (where $M$ is Marc and $p(x, y)$ means $x$ is $y$’s parent):

$$
MA = \{ x | p(x, M) \lor \exists y (y \in MA \land p(x, y)) \}
$$

(c) $N! = 1 \cdot 2 \cdot 3 \cdots N$

(i) **Base Case:** $0! = 1$

(ii) **Recursive definition:** $n! = n \cdot (n - 1)!$

(4) **Counting, Permutations, and Combinations**

(a) Number of TLA’s (Three Letter Acronyms) $= 26^3$

(b) Number of 10 digit phone numbers, excluding those that start with 0 or 1 or have 555 as their middle 3 numbers: $= 10^{10} - (2 \cdot 10^9 + 10^7 - 2 \cdot 10^6)$

(c) Pigeonhole principle: If there are 27 people in the class, and 4 Starburst flavors, then at least one flavor must be the favorite of at least $\left\lceil \frac{27}{4} \right\rceil = 7$ people.

(d) $\text{n Pick a} = \frac{n!}{(n-a)!}$

(e) $\text{n Choose a} = \binom{n}{a} = \frac{n!}{(n-a)!a!}$
(f) Number of ways to put 9 baseball players on a field (with assigned positions) out of a team of 20 = $\binom{20}{9}$

(g) Number of ways to put 6 men and 4 women on a dodgeball team (from 10 women and 12 men) where the positions don’t matter = $\binom{10}{6} \binom{12}{4}$

(h) Number of ways to match exactly 3 of 6 lotto numbers (out of 49) = $\binom{6}{3} \binom{43}{3}$

(i) Number of full houses in 5 card poker = $13 \binom{4}{3} 12 \binom{4}{2}$

(j) Number of ways to get a pair of kings in 5 card poker = $\binom{4}{2} \binom{48}{3}$

(k) Binomial expansion: $(A + B)^n = \sum_{i=0}^{n} \binom{n}{i} A^i B^{n-i}$

(l) Pascal’s identity: $\binom{n+1}{k} = \binom{n}{k-1} + \binom{n}{k}$

(m) Pascal’s triangle:

```
      1
     1 1
    1 2 1
   1 3 3 1
  1 4 6 4 1
 1 5 10 10 5 1
   (etc.)
```

(n) Ways to arrange letters in DELETE = $\frac{6!}{3!}$, SERENDIPITY = $\frac{11!}{2!2!}$

(o) Ways to make a pack of Starburst (4 flavors, 12 Starbursts in a pack) = $\binom{12+(4-1)}{4-1}$

(5) Probability

(a) Prob of 2 of 27 people having same birthday = 1 - Prob of all B-days being different = $1 - \frac{366}{366} \cdot \frac{365}{366} \cdot \frac{364}{366} \cdots \frac{360-26}{366} = 1 - \frac{366!}{(366-27)!366^{27}} \approx 0.6258$

(b) Discrete Probability principles (where $P(E)$ means the probability of event $E$):

   (i) $0 \leq P(E) \leq 1$

   (ii) If $O$ is the set of all possible outcomes $1 = \sum_{o \in O} P(o)$

   (iii) $1 - P(E) = P(\sim E)$

   (iv) $P(A \lor B) = P(A) + P(B) - P(A \land B)$
(v) Events $A$ and $B$ are independent iff $P(A \land B) = P(A) \cdot P(B)$

(c) Conditional Probability:

(i) Bayes’s law: $P(B|A) = \frac{P(A|B)P(B)}{P(A)}$

(ii) $P(A \land B) = P(A|B)P(B)$

(iii) $P(A) = P(A \land B) + P(A \land \sim B) = P(A|B)P(B) + P(A|\sim B)P(\sim B)$

(d) Bernoulli Distribution: $P(rolling\ 1)\ exactly\ 3\ times\ out\ of\ 10 = P(rolling\ 1)^3(1 - P(rolling\ 1))^7\left(\binom{10}{3}\right)$. More generally, the probability of rolling exactly $a$ 1’s when rolling $b$ $n$ sided dice is $\binom{1}{n}^a \left(\frac{n-1}{n}\right)^{b-a} \binom{k}{a}$.

(e) Some statistics

(i) Expected value $= E[X] = \sum_{x \in X} P(x)x$

(ii) Variance $= V[X] = E[(x - E[X])^2] = \sum_{x \in X} P(x) (x - E[X])^2$

(iii) Standard Deviation $= \sigma[X] = \sqrt{V[X]}$

(6) Recurrence Relations:

If the $n$th term of a recurrence relation is defined as $a_n = c_1a_{n-1} + c_2a_{n-2} + \cdots + c_ka_{n-k}$ (and the base cases are defined to be $a_1 = m_1, a_2 = m_2, \cdots, a_k = m_k$ (where $m_i$ is a constant)), then there will be a solution of the form $a_n = r^n$. The characteristic equation for $a$ is then

$$r^k - c_1r^{k-1} - c_2r^{k-2} - c_3r^{k-3} - \cdots - c_k = 0$$

If this equation has distinct roots $r_1, \cdots, r_k$, then $a_n = \alpha_1r_1^n + \cdots + \alpha_kr_k^n$, where $\alpha_i$ is a constant. We can solve for the $\alpha$s by solving the system of linear equations given for the first $k$ $a$’s.

(7) Relations

(a) Relation $<$ on Natural numbers as a matrix

$$
\begin{bmatrix}
1 & 2 & 3 & 4 & 5 & \ldots \\
1 & 0 & 1 & 1 & 1 & \\
2 & 0 & 0 & 1 & 1 & \text{etc}.
\end{bmatrix}
$$

$$
\begin{bmatrix}
3 & 0 & 0 & 0 & 1 & 1 & \ldots
\end{bmatrix}
$$
4 0 0 0 0 1
5 0 0 0 0 0 . . .
   etc...

(b) Number of possible relations on set $S = 2^{|S|^2}$

(c) **reflexive** $\forall a \in S, (a, a) \in R$: Examples: $\leftrightarrow, \leq, =$. Non-examples: “Parent”, $\land$

(d) **symmetric** $\forall a, b \in S, (a, b) \in R \iff (b, a) \in R$: Examples: “Acted with”, “married to”, XOR. Non-examples: “loves”, $\implies$

(e) **antisymmetric** $\forall a, b \in S, (a, b) \in R \land (b, a) \in R \implies a = b$: Example: $\implies$, $\leq$. Non-example: XOR.

(f) **asymmetric** $\forall a, b \in S, (a, b) \in R \land (b, a) \in R$: Examples: “Parent”, $<$

(g) **transitive** $\forall a, b, c \in S, (a, b) \in R \land (b, c) \in R \implies (a, c) \in R$: Examples: “Ancestor”, $<$, “Taller than”. Non-examples: “beats at chess”, “Acted with”.

(h) Since Relations are sets, we can combine them with set operators ($\cup$, $-$, $\cap$). E.g. if $R1$ is the relation between a set of students in 203 and classes offered at UMBC that contains $(s, c)$ iff student $s$ has taken class $c$, and $(s, c) \in R2$ iff $c$ is a required course for $s$’s major, then:

   (i) $(s, c) \in R1 \cup R2$ means that $s$ is guaranteed to have taken $c$ by the time he/she graduates.

   (ii) $(s, c) \in R2 - R1$ means that $s$ needs to take $c$ yet to graduate.

   (iii) $(s, c) \in R1 \cap R2$ means that $s$ has taken $c$ as a required class (as opposed to an elective).

(i) Composition of Relations: $(a, b) \in R1 \land (b, c) \in R2 \implies (a, c) \in R1 \circ R2$: E.g Grandparent = Parent $\circ$ Parent

(8) **Graphs**

   (a) UMBC students and UMBC classes (and the edges representing that a student is taking a class) is a **bipartite graph** because no class is directly connected to any other class (and no student is directly connected to another student).
(b) If there’s a path from every UMBC student to every class and every other student, then the UMBC student/class graph is not disjoint.

(c) A graph can be represented as a set of edges, and for this reason, all the set operations can apply to graphs too. For example, the complement of a simple undirected graph that has $n$ nodes and $e$ edges has $e - \frac{n^2 - n}{2}$ edges: an edge everywhere there wasn’t an edge before.

(d) A pair of graphs $G_1$ and $G_2$ are isomorphic if there’s a mapping from the nodes of $G_1$ to the nodes of $G_2$ such that if you rename $G_1$’s nodes according to this mapping, you’ll have a graph identical to $G_2$.

(e) A graph is planar if the graph can be laid out on a surface without edges intersecting (edges need not be straight though).

(f) There are $2^{\frac{n^2 - n}{2}}$ possible simple undirected graphs over $n$ nodes (see the solutions for Exam 2 for a derivation of this expression).

(9) Trees

(a) Huffman’s algorithm for making an encoding tree:

(i) Compute the letter frequencies

(ii) Make 26 “trees” (assuming you have 26 characters), each with a count of the character they represent and a single node representing a character.

(iii) “Merge” the 2 trees with the lowest count. This means to create a new tree whose count is the sum of the 2 other counts, and whose left branch is marked with a 0, and whose right branch is marked with a 1 (or vice versa).

(iv) Keep merging until you have only 1 tree.

(v) The encoding for a character will be markings along the path from the root to the leaf that represents that character.

(b) Some discrete multi-player turn-based games (e.g., Chess, Tic-Tac-Toe) can be represented as a game tree. The root is the starting state, and each level
represents different players turns. The leaves represent final outcomes of the
game. Given a complete tree for a deterministic game, one can determine an
optimal strategy for each player.

(10) Boolean Algebra

(a) For any Boolean expression (e.g. \( \sim (A \land B) \implies C \)) You should be able to
construct a circuit using logic gates. How to represent these in \LaTeX\ goes beyond
my expertise, but if needed for the final, I’ll give the logic circuit symbols for
\textbf{and}, \textbf{or}, \textbf{not}, \textbf{xor}, \textbf{nand}, and \textbf{nor}.

(b) A logical operator is \textbf{universal} iff it alone can be used to construct any other
logical operator. \textbf{nand} and \textbf{nor} are universal. \textbf{and} and \textbf{not} together make a
\textbf{universal set} because they can be used together to make any logical expression.
\( \implies \) combined with a \textbf{True} signal makes a universal set.

(11) Languages and Grammars

(a) \(<S> ::= A <S> A | D <S> D | A | D | \lambda \)

Is a grammar that generates all palindromes over the characters \( A \) and \( D \). \(<S> \) is
a non-terminal, and \( A \) and \( D \) are terminals \( \lambda \) (or \( \lambda \)) is the empty string.

(b) The string \textbf{ADDA} is in the \textbf{language} of the above grammar. This means that it
can be \textbf{parsed} by the grammar. One (and the only in this case) parsing is

\[
<\mathrm{S}>
/ \mid \backslash
\]

\[
A <\mathrm{S}> A
/ \mid \backslash
\]

\[
D \mid D
\mid
(\lambda)
\]

(12) Finite-State Automata
(a) A finite state automaton (FSA) has states and transition rules. Each transition rule says if we read character $c$ while in state $x$, we transition to state $y$. One of the states is marked as the starting state and one or more states is marked as an accepting state. If, when reading a string and starting in the start state, we end in an accepting state, we say that the string is in the language of the FSA.

(b) For every FSA there’s (at least one) regular expression and vice versa. That is, the regular expression and the FSA have the same language.

(c) A regular expression may contain the following:

(i) $A|B$ This is or. So $((MARC) | (B*))$ would be either the string MARC or a string of 0 or more Bs (but not both.)

(ii) $AB$ I.e., one regular expression can follow another.

(iii) $A*$ Also called a Kleene star. This means you may repeat the preceding regular expression 0 or more times. For example the $(AA|B)^*$ means any string composed only of Bs and pairs of As.

(13) Turing Machines

(a) A Turing machine has a reader head on an infinite tape and a transition table where each entry says: when in state $x$ and reading a $y$ on the tape, write a $z$, move the reader head (left or right), and transition to state $w$.

(b) A Universal Turing machine is one that can read a description of any other Turing machine on the tape (as well as that machine’s input), then simulates running that machine on that input. A Universal Turing machine has as much computing power (though not in terms of runtime) as any computer ever built, and it’s believed that no computer can ever have more computing power.

(c) Halting problem: It’s impossible to have a function boolean $\text{halt}(P)$ which takes a program $P$ and returns true iff $P$ halts eventually (when started with no input). Proof by contradiction: Assume such a function existed, then we could write the program $M$: 
M:

```java
int i = 0
if halt(M)
    while(true)
        i = i + 1
else
    exit
```

Thus $M$ halts iff $\text{halt}(M)$ is false which is a contradiction.