Final Reference Sheet — Math 61 — Winter 2015

Ky-Cuong L. Huynh

20 March 2015

0.1 Basics

Triangle numbers: $(T_n = T_{n-1} + n) = (\sum_{k=1}^{n} k) = \frac{n(n-1)}{2}$

Geometric sum: $\sum_{k=0}^{n} b^k = \frac{b^{k+1}-1}{x-1}$

Fibonacci numbers (the number of ways to tile an $1 \times n$ strip with squares and dominoes): $F_{n+2} = F_{n+1} + F_n$ for $F_0 = 0, F_1 = 1$

The power set $\mathcal{P}(A)$ of a set A is the set of all subsets of A. $|\mathcal{P}| = 2^n$. The Cartesian product of two sets A and B is the set of all ordered pairs of elements from both sets: $A \times B = \{ (a, b) : a \in A, b \in B \}$ Rule: $|A \times B| = |A| \cdot |B|$. Two sets A, B are equal $(A = B)$ if $A \subseteq B$ and $B \subseteq A$.

Base case: Show $P(n)$ holds for some small integer $b < n$, i.e., $P(b)$ is true. Inductive step: Assume for some fixed $n \geq b$ that $P(n)$ holds. Show that expression in $(n+1)$th case equals $P(n+1)$. Invoke inductive hypothesis of $P(n)$ to do so. Then, $P(n) \Rightarrow P(n+1)$.

Tricks to rewrite $P(n+1)$: split off last term in sum; factor out terms; highest or lowest term doesn't exist/is 0, so rewrite indices to match; point out inequalities; etc.

For strong induction, we prove $P(n)\ \forall\, n \geq b$. Base case: $P(b)$. Then, fix a value $n > b$, and prove that $P(n)$ is true assuming that $P(m)$ is true for all m satisfying $b \leq m < n$.

0.1.1 Functions

A function $f : X \to Y$ assigns exactly one element of Y to every element of X. Each element in the domain (input) has only one associated element in the codomain (output), e.g., at most one arrow from each dot in domain. Formal definition: A function $f : X \to Y$ is a subset $f \subseteq X \times Y$ such that for each $x \in X$, there is exactly one $y \in Y$ such that $(x, y) \in f$.

A function $f : X \to Y$ is surjective if for every $y \in Y$, there exists $x \in X$ such that $f(x) = y$, i.e., its range (actual output) is equal to its codomain (set in which output constrained to fall); every dot in the codomain has at least one arrow pointing to it.

A function $f : X \to Y$ is injective if for every $x_1, x_2 \in X$, $f(x_1) = f(x_2) \Rightarrow x_1 = x_2$, i.e., every element of the domain maps to at most one element of the codomain; every dot in the codomain has at most one arrow pointing to it. A function is bijective if it is both surjective (onto) and injective (one-to-one), which means that it is invertible and f^{-1} exists (and can be constructed), with $(f \circ f^{-1}) : Y \to Y$ and $(f^{-1} \circ f) : X \to X$.

If there is an injection $X \to Y$, then $|X| \leq |Y|$ (more inputs than outputs, double-map). If there is a surjection $X \to Y$, then $|X| \geq |Y|$ (need enough enough inputs to map to all the outputs). If there is a bijection, then $|X| = |Y|$.

0.2 Relations

A relation on a set S is a subset $R \subseteq S \times S$. We write $x \,\mathrm{R}\, y$ if $(x, y) \in S$. Congruence modulo m is $a \equiv_m b$ for $m \in \mathbb{Z}_{\geq 0}$ if $m|(a - b)$ ('—' == "divides without remainder").

Reflexivity: For all $x \in S$, $x \,\mathrm{R}\, x$. Transitivity: For all $x, y, z \in S$, $(x \,\mathrm{R}\, y) \wedge (y \,\mathrm{R}\, z) \Rightarrow x \,\mathrm{R}\, z$; Symmetry: For all $x, y \in S$, $x \,\mathrm{R}\, y \Rightarrow y \,\mathrm{R}\, x$. Anti-symmetry: For all $x, y \in S$, if $x \,\mathrm{R}\, y$, then *not* $y \,\mathrm{R}\, x$; the relationship is one-way only. Comparability: For all $x, y \in S$, then $x \,\mathrm{R}\, y$ or $y \,\mathrm{R}\, x$.

A partial order is a relation that is reflexive, transitive, and anti-symmetric. A total order is a relation that is reflexive, transitive, anti-symmetric, and comparable. Every total order is a partial order, but not vice versa.

An equivalence relation is a relation that is reflexive, symmetric, and transitive. An equivalence class $[x]$ (for an element x in a set S and an equivalence relation $\sim$ is the set of all $y \in S$ such that $y \sim x$. Every element of S belongs to exactly one equivalence class, and two different equivalence classes are automatically disjoint.

If $|A| = n$ and $|B| = m$, then there are m^n functions $A \to B$, $\frac{m!}{(m-n)!}$ that are injective. If $|S| = n$, then there are 2^{n^2} possible relations on it (the subsets of $S \times S$), with $2^{n(n-1)}$ reflexive, and $2^{(n(n+1))/2)}$ symmetric.

0.2.1 Hasse Diagrams

To build a Hasse diagram, list out the elements and put one at the next level if the partial ordering links to it. For example, 1, 2, 3, 4—stop, next level, as 2 divides 4, but 5 is at one's level.

0.3 Combinatorics

A k-combination of a set S with $|S| = n$ is an unordered subset of k (distinct) elements taken from S: $C(n, k) = \frac{P(n,k)}{k!} = \frac{n!}{k!(n-k)!}$ A k-permutation of a set S with $|S| = n$ is an ordered sequence of k (distinct) elements taken from S (assume $k = n$ if k not specified): $P(n, k) = \frac{n!}{(n-k)!}$.

A partition of A is a set $S = \{A_1, \ldots, A_n\}$ of subsets of A such that $A_1 \cup \cdots \cup A_n = A$ and whose elements are pairwise disjoint: $A_i \cap A_j = \varnothing$ for $i \neq j$.

The Binomial Theorem: $(x + y)^n = \sum_{k=0}^{n} \binom{n}{k} x^k y^{n-k}$.

0.3.1 Counting Principles

Multiplication Principle: If a decision is made in t steps, and at each step there are n_t choices, then the number of possible decisions is $n_1 \cdots n_t$.

Addition Principle: If A is a finite set and $\{A_1, \ldots, A_n\}$ is a partion of A, then $|A| = |A_1| + \cdots + |A_n|$.

Inclusion-Exclusion Principle: For any two (possibly overlapping) sets A and B, $|A \cup B| = |A| + |B| - |A \cap B|$. For three sets, $|X \cup Y \cup Z| = |X| + |Y| + |Z| - |X \cap Y| - |Y \cap Z| - |Z \cap X| + |X \cap Y \cap Z|$

0.4 Pigeonhole Principle

If there are p objects to be put into q boxes, and $p > q$, then at least one box will contain > 1 objects. If we place p objects into q

boxes, and $p > kq$, then at least one box will contains $> k$ objects. Key phrase: "Show that two objects share the property..."

0.5 Linear Recurrences

If we have a recurrence $a_n = c_1 a_{n-1} + c_2 a_{n-2}$, then find the roots of the characteristic equation $r^2 - c_1 r - c_2 = 0$. If we have two distinct roots, then the general solution is $a_n = Ar_1^n + Br_2^n$. If we have one repeated root, then the general solution is $a_n = Ar_1^n + nBr_2^n$. Plug in initial conditions to find A, B.

0.6 Graphs

Handshaking lemma: In any undirected graph, the number of vertices with odd degree is even. This is a consequence of the degree sum formula: $\sum_{v \in V} \deg(v) = 2|E|$.

A graph $G = (V, E)$ is bipartite iff it has no cycles of odd length. Its vertices can be partioned into two (nonempty) sets $A \subsetneq V$ and $B \subsetneq V$ such that vertex in A is adjacent only to vertices in B, and vice versa. In a complete graph, every pair of vertices is adjacent and there are $\binom{n}{2}$ edges.

The trivial path is the one not leaving a vertex. A simple path is one that avoids repeating vertice. A closed path begins and ends at the same vertex. A cycle is a closed path that avoids repeating edges.

An Euler cycle is a path that begins and ends at the same vertex, visiting every edge exactly once. A graph is Eulerian (contains an Euler cycle) iff every vertex has even degree. An Euler trail is a path that visits every edge exactly once, but that may not begin/end at the same vertex. A graph is Semi-Eulerian (contains an Euler trail) iff there are exactly two vertices of odd degree, and the rest even.

A Hamilton cycle is a cycle that visits every vertex exactly once. A graph is Hamiltonian if it contains a Hamilton cycle; this an NP-complete problem. A Hamilton trail is a path that visits every vertex exactly once, but that may not begin/end at the same vertex. A graph is Semi-Hamiltonian if it contains such a path, and it is an NP-complete problem to determine this.

Euler's formula: $|F| - |E| + |V| = 2$ (for planar graph only) For simple, planar graphs: $|E| \leq 3|V| - 6$. For simple, bipartite, planar graphs: $|E| \leq 2|V| - 4$.

A planar graph is one that that is isomorphic to one that *can be* drawn *in the plane* without edges that cross/overlap. Kuratowski's Theorem states that a graph G is planar if and only if it does not have *any* subgraphs G' that (series)-reduce to (i.e., are homeomorphic to) K_5 or $K_{3,3}$.

Adjacency matrices: The ijth entry in A^n counts the number of n-step paths from v_i to v_j.

0.6.1 Graph Isomorphisms

Two graphs $G_1 = (V_1, E1)$ and $G_2 = (V_2, E_2)$ are said to be isomorphic to each other, $(G_1) \simeq (G_2)$, if: 1) There exists two bijective functions $f_V : V_1 \to V_2$ and $f_E : E_1 \to E_2$. 2) Let v, w be vertices in G_1. Then, v, w are adjacent if and only $f_V(v)$ and $f_V(w)$ are adjacent in G_2. That is, we can map vertices from one graph to another, while preserving their edges. Two vertices adjacent in one graph are also adjacent in the other.

To see if two graphs are ismorphic, first check to see if they are not, by verifying that their graph invariants agree. These properties must be true for both graphs XOR false for both: number of vertices, edges, k-cycles, vertices with degree k; bipartiteness; Eulerian; Hamiltonian; connectedness; etc. If these all agree, attempt to map vertices edges from one to the other, and check that their adjacency relationships sitll hold.

A series reduction removes a vertex intermediate between two other vertices, and and links the two on either side of it. Two graphs are called homeomorphic if their reduced forms are isomorphic to each other: $(G_1)^{\text{red.}} \simeq (G_2)^{\text{red.}}$

0.7 Trees

A tree is a connected, bipartite graph with $|E| = |V| - 1$, no cycles, and a unique shortest path between any pair of vertices. It is any graph with a unique simple path between any two vertices. The trivial graph (one node) is a tree. A binary tree is a rooted tree where every vertex has at most 2 children that are specified as left or right children (angle needed). A spanning tree of a graph G is a subgraph G' such that G' contains all the vertices of G and G' is a tree.

Theorems: Every graph has at least one spanning tree. If T is a full binary tree with i internal vertices, then T has $i + 1$ terminal vertices and $2i + 1$ vertices total. If a binary tree has height h and t terminal vertices, then $\lg(t) \leq h$.

To build a Huffman tree: Given a frequency table for some symbols, add the lowest scores together, and continue doing so until only two remain. (If three symbols of same frequency, choose two arbitrarily). Then, these are the children of the root node for a tree with value equal to their sum. Expand until no longer possible. Replace the numbers with the original symbols, and label left children with 1 and right children with 0.

0.8 Sorting

Lower bound on sorting: $\frac{1}{4} n \lg(n)$ comparisons in the worst case for a list of $n \geq 4$. Upper bound on mergesort: $\leq 2n \lg(n) + 1$. Mergesort merging: compare least element of two cells at current level, like combining sorted stacks of paper.

0.9 Graph Computations

Dijkstra's Algorithm: Setup best guesses list; circle vertex on list with shortest distance; update its neighbors' best guesses on list if they're less than the current best guess; repeat until shortest path lengths found for all vertices. When we circle a vertex, we lock in that choice as the best possible. We never connect to a previously-circled vertex, as that would introduce a cycle.

Breadth-first search builds a spanning tree: Start at vertex with smallest label; add all of its neighbors to our tree; go through the added neighbors' in alphanum order and add them if not already added and they don't introduce a cycle. Iterate until no vertices remain.

Depth-first search: Start at vertex with smallest label; choose its smallest (by alphanum order), undiscovered (i.e., does not introduce a cycle) neighbor; iterate until no vertices remain or we hit a deadend, in which case, backtrack as little as possible to add remaining vertices.

Prim's Algorithm builds the minimal spanning tree: Circle a starting vertex; add the edge with the least weight and circle that vertex; examine the edges adjacent to currently-circled vertices and (without introducing a cycle), add the one with the least weight (break ties with alphanum order of indices); iterate until no vertices remain.