WMx

Warwick Mathematics Exchange

CS137

Discrete Mathematics & Applications II

2022, June 13th

Desync, aka The Big Ree

Contents

Introduction

This module covers basic concepts in discrete maths, and on applications of discrete maths in algorithms and data structures. Specifically, we cover combinatorics, recurrence relations, basic graph theory and random processes. Despite the name, this module is almost purely theoretic, with *Applications* referring to "applications in proving theorems".

This document is intended to broadly cover all the topics within the DM2 module. Knowledge of basic algorithms and methods on solving linear recurrence relations is assumed and will not be covered. If you need a refresher on the latter, please read my guide on MA133 *Differential Equations*.

This document is not designed to be a replacement for lecture notes, although you can certainly use it as one if you already have a solid understanding of the content from outside of the course - much of the content is covered in a different order than is taught in the course (for example, graph theory is taught in two completely separate parts, but are presented here all together to keep the sections cohesive), so it is not recommended to learn the module from these notes unless you are familiar with most of the content already.

This notice is present in all of my guides, but applies particularly strongly for this guide, which effectively amounts to a list of computational skills you'll need.

Many of the techniques developed here are used extensively in further (and many core!) modules for *Discrete Mathematics*, *Data Science* and *Computer Science*, so take care not to just forget everything you've learned once the exam has passed by.

Disclaimer: This document was made by a first year student who did not go to any lectures for the second half of this module (and missed two-thirds of the first half due to timetable collisions). I make *absolutely no guarantee* that this document is complete nor without error. In particular, any content covered exclusively in lectures (if any) will not be recorded here. Additionally, this document was written at the end of the 2022 academic year, so any changes in the course since then may not be accurately reflected.

Notes on formatting

New terminology will be introduced in *italics* when used for the first time. Named theorems will also be introduced in *italics*. Important points will be **bold**. Common mistakes will be underlined. The latter two classifications are under my interpretation. YMMV.

> Content not taught in the course will be outlined in the margins like this (if any). Anything outlined like this is not examinable, but has been included as it may be helpful to know alternative methods to solve problems.

The table of contents above, and any inline references are all hyperlinked for your convenience.

History

First Edition: 2022-06-12*
Current Edition: 2022-06-13

Authors

This document was written by R.J. Kit L., a maths student. I am not otherwise affiliated with the university, and cannot help you with related matters.

Please send me a PM on Discord @Desync#6290, a message in the WMX server, or an email to Warwick.Mathematics.Exchange@gmail.com for any corrections. (If this document somehow manages to persist for more than a few years, these contact details might be out of date, depending on the maintainers. Please check the most recently updated version you can find.)

If you found this guide helpful and want to support me, you can buy me a coffee!

(Direct link for if hyperlinks are not supported on your device/reader: ko-fi.com/desync.)

*Storing dates in big-endian format is clearly the superior option, as sorting dates lexicographically will also sort dates chronologically, which is a property that little and middle-endian date formats do not share. See ISO-8601 for more details. This footnote was made by the computer science gang.

1 Complexity Analysis

Consider two functions, $f(n)$ and $g(n)$.

Suppose

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} \to a$$

If,

- $a = 0$, then $f(n) \in O(g(n))$
- $a = \infty$, then $f(n) \in \Omega(g(n))$
- $a \in (0,\infty)$, then $f(n) \in \Theta(g(n))$

We sometimes use $=$ instead of $\in$, but this equality is not symmetric. $O(f(n)) = O(g(n))$ is not the same as $O(g(n)) = O(f(n))$. For example, $O(n) = O(n^2)$, but $O(n^2) \neq O(n)$. For this reason, $\in$ will be preferred in this document.

As we only care about the shape of growth as n becomes very large, when analysing runtime complexity, we discard all coefficients, and keep only the term with the highest growth rate, as it will eventually dominate everything else.

Applying a concave function, such as log, to both $f(n)$ and $g(n)$ does not change the asymptotic relationship between them.

Example. Is $2^{\log(\log(n))^3} \in \Omega\left(\sqrt{n}\right)$?

$$\begin{aligned}
f(n) &= 2^{\log(\log(n))^3} \\
g(n) &= \sqrt{n} \\
f^*(n) &= \log(\log(f(n))) \\
&= \log\left(\log\left(2^{\log(\log(n))^3}\right)\right) \\
&= \log\left(\log(\log(n))^3 \log(2)\right) \\
&= \log\left(\log(\log(n))^3\right) + \log(\log(2)) \\
&= 3\log(\log(\log(n))) + \log(\log(2)) \\
g^*(n) &= \log\log(g(n)) \\
&= \log\left(\log\left(\sqrt{n}\right)\right) \\
&= \log\left(\frac{1}{2}\log(n)\right) \\
&= \log(\log(n)) + \log\left(\frac{1}{2}\right) \\
\lim_{n\to\infty} \frac{f^*(n)}{g^*(n)} &= \lim_{n\to\infty} \frac{3\log(\log(\log(n))) + \log(\log(2))}{\log(\log(n)) + \log\left(\frac{1}{2}\right)}
\end{aligned}$$

Let $N = \log(\log(n))$. As $n \to \infty$, $N \to \infty$.

$$\begin{aligned}
&= \lim_{N\to\infty} \frac{3\log(N) + \log(\log(2))}{N + \log\left(\frac{1}{2}\right)} \\
&= \lim_{N\to\infty} \frac{3\log(N)}{N + \log\left(\frac{1}{2}\right)} + \lim_{N\to\infty} \frac{\log(\log(2))}{N + \log\left(\frac{1}{2}\right)} \\
&= \lim_{N\to\infty} \frac{3\log(N)}{N + \log\left(\frac{1}{2}\right)}
\end{aligned}$$

As $N \to \infty$, $3\log(N) \to \infty$ and $N + \log\left(\frac{1}{2}\right) \to \infty$, so apply l'hopital's rule.

$$\begin{aligned} &= \lim_{N\to\infty} \frac{3}{N} \\ &= 0 \end{aligned}$$

So $2^{\log(\log(n))^3} \in O\left(\sqrt{n}\right)$, and $2^{\log(\log(n))^3} \notin \Omega\left(\sqrt{n}\right)$.

Example. Bubble sort.

To sort a list using *bubble sort*, we check the first two elements of the list, and swap them, if they are out of order. Then, we move along one, and check two elements again, then repeat until we reach the end of the list, where we start another pass. Once we pass through the list without performing any swaps, we know that the list is sorted.

Algorithm 1 Bubble Sort

```
1:  procedure BUBBLESORT(A)                                  ▷ Input array
2:      n ←LEN(A)
3:      repeat
4:          swapped = false
5:          for i = 1 to n − 1 do
6:              if A[i − 1] > A[i] then          ▷ Check if the elements are out of order
7:                  (A[i − 1],A[i]) ← (A[i],A[i − 1])          ▷ Swap elements
8:                  swapped = True
9:              end if
10:         end for
11:     until swapped = False
12:     return A                                          ▷ Return the sorted list
13: end procedure
```

The comparison and swapping takes $\Theta(1)$ time, but runs $(n-1)$ times due to the for loop. The repeat statement will also run the for loop $(n-1)$ times, so overall, the algorithm takes $\Theta((n-1)^2) = \Theta(n^2)$.

One way to remember this result is to think about what happens if the smallest element is in the last place. Every time it is swapped, it only moves one place back, so $(n-1)$ passes are required, each one taking $\Theta(n)$ time, giving $\Theta(n^2)$.

1.1 Master Theorem

Master Theorem: For an algorithm that has complexity that obeys the equation,

$$T(n) = aT\left(\frac{n}{b}\right) + \Theta(n^d), T(c) = \Theta(1)$$

we have,

$$T(n) \in \begin{cases} \Theta(n^d) & a < b^d \\ \Theta(n^d \cdot \log n) & a = b^d \\ \Theta(n^{\log_b a}) & a > b^d \end{cases}$$

Example. Merge sort.

In *merge sort*, we divide the list into halves, then run the algorithm again on each half, returning the list once the list is length 1. Then, we merge the sorted sublists together until only one list remains.

Algorithm 2 Merge Sort

```
1:  procedure MERGESORT(A)                                   ▷ Input array
2:      n ←LEN(A)
3:      if n ≤ 1 then
4:          return A          ▷ If the list only contains one element, it is already sorted
5:      end if
6:      left← [ ]
7:      right← [ ]
8:      for i = 1 to n do
9:          if i < n/2 then
10:             APPEND(left)          ▷ Split the list into two sublists, left and right
11:         else
12:             APPEND(right)
13:         end if
14:     end for
15:     left←MERGESORT(left)                                 ▷ Sort the two sublists
16:     right←MERGESORT(right)
17:     return MERGE(left,right)                  ▷ Merge the two sorted sublists together
18: end procedure
```

where the merge subroutine combines two sorted lists into one sorted list in linear time.

The algorithm takes

$$T(n) = \underbrace{T\left(\left\lfloor \frac{n}{2} \right\rfloor\right)}_{\text{Sort left}} + \underbrace{T\left(n - \left\lfloor \frac{n}{2} \right\rfloor\right)}_{\text{Sort right}} + \underbrace{\Theta(n)}_{\text{Merge}}, n > 1$$

and we know $T(1) = \Theta(1)$, as the algorithm just returns the list for an array of length 1, taking constant time.

$$\sim 2T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + \Theta(n)$$

So, using the Master theorem, we have $a = 2$, $b = 2$, $d = 1$, so,

$$= \Theta(n \log n)$$

2 Graph Theory

A *graph*, G, is represented by V, a set of *vertices* or *nodes*, and E, a set of pairs of vertices, called *edges* or *arcs*. We write $G = (V,E)$. If the edge pairs are ordered, the graph is *directed*, and can also be referred to as a *digraph*.

A vertex and an edge are *incident* if the vertex is at either end of the edge. The *degree*, *valency* or *order* of a vertex is the number of edges incident to it. The *indegree* and *outdegree* of a vertex of a digraph is the number of edges pointing into and out from the vertex.

The *degree sequence* of a graph is a sorted list of its vertex degrees. If a sequence of number is the degree sequence of some graph, it is *graphical*. For example, 3, is not a graphical sequence, as there is no graph with a single node of degree 3, while 2,2,2 *is* a graphical sequence, because the triangle graph has 2,2,2 as a degree sequence.

An edge that starts and ends at the same vertex is called a *loop*. If multiple copies of the same edge pair exists in the edge set, then the edges are called *parallel edges*.

A graph that does not contain loops or parallel edges is called a *simple* graph. A graph that can contain parallel edges and loops is a *multigraph*.

If each edge also has a number associated with it (the *weight* of the edge), the graph is a *weighted graph* or a *network*.

A *walk* is a route through a graph. A walk is *closed* if the first and last vertices are the same, and *open* otherwise. A *path* is a walk in which no vertex is visited more than once. A *trail* is a walk in which no edge is visited more than once. A *cycle* is a path in which the ending and starting vertex are the same.

A *Hamiltonian cycle* is a cycle that visits every vertex. An *Eulerian walk* is a trail which traverses every edge. An *Eulerian circuit* is both a trail and cycle which traverses every edge.

Euler's Theorem: An Eulerian circuit exists if and only if every vertex is of even degree.

Corollary: An Eulerian walk exists if and only if there are at most two vertices of odd degree.

A graph that admits an Eulerian walk is *traversable* or *semi-Eulerian*. A graph that admits an Eulerian circuit is *Eulerian*.

Two vertices are *connected* if there is a path between them. Two vertices, u and v, are *adjacent* if they are connected by an edge, so $(u,v) \in E$. u and v are also called *neighbours*. In a directed graph, the *in-neighbours* of a vertex v, are all vertices u such that $(u,v) \in E$, and the *out-neighbours* are all vertices u such that $(v,u) \in E$.

$N(v)$ represents the set of neighbours of v, but does not include v itself. This notation can also be used on sets of vertices to represent the set of neighbours of that set of vertices.

Euler's Handshaking Lemma: In any undirected graph, the sum of the degrees of the vertices is equal to twice the number of edges. It follows that the number of odd degree vertices is even.

A *path* graph, P_n, is a graph consisting of a sole path, without cycles. It's basically a line of nodes, with a single path/trail running through it. Symbolically, it's a graph on n nodes, $v_1, \cdots, v_n$ with $E = \{(v_i,v_{i+1}) | i \in [1,n-1]\}$.

A *cycle* graph, C_n, is a graph consisting of a sole cycle.

A *complete* graph, K_n, is a graph on n nodes with every possible edge included once.

A *tournament* is a directed complete graph. If an edge points from a vertex a to a vertex b, then a *dominates* b. If $D = (V,E)$ is a tournament, and $S \subset V$ with $|S| = k$, then S is a *k-strong set* if for every $v \in V \setminus S$, there exists a $u \in S$ such that $(u,v) \in E$. In other words, every vertex not in S is dominated by at least one vertex in S.

A *bipartite* graph is a graph that has a vertex set that can be partitioned into two subsets, commonly denoted L and R, such that for, every edge, $(u,v) \in E$ either $u \in L$ and $v \in R$ *or* $u \in R$ and $v \in L$.

For all $k \in \mathbb{N}$, there exists a tournament on $n \in \mathbb{N}$ vertices without a k-strong set.

A graph is *connected* if every pair of vertices is connected.

A graph is a *tree* if it does not contain a cycle. An unconnected tree graph may also be called a *forest*. Every *tree* is bipartite.

A *subgraph* of a graph, G, is a graph whose vertices and edges all belong to G. A subgraph is *induced* if every edge that can be included (depending on the vertex subset), is included. A *l-clique* is a subgraph that is a *complete* graph on l vertices. A *spanning tree* is a subgraph that contains every vertex of the original graph, and is also a tree.

Every connected graph has a spanning tree. Every connected graph over n has at least $(n-1)$ edges with exactly $(n-1)$ if and only if the graph is a tree.

The *complement* of a graph, $G = (V,E)$, denoted $\bar{G}$ or G^c is the graph (V,E'), where E' is the set of edges over V that are not in E.

Two graphs, $G = (V,E)$ and $H = (W,F)$ are *isomorphic* if there exists a bijective function, $\phi : V \to W$ such that if $(v_1,v_2) \in E$, then $\phi(v_1),\phi(v_2) \in F$, and vice versa. If such a function exists, we write $G \cong H$. The best known algorithm to determine whether two given graphs are isomorphic is $O(n^{\log n})$.

A *matching* over a graph $G = (V,E)$ is a set of edges $M \subseteq E$ such that no vertex is incident to more than one edge. A matching in which every vertex is incident to an edge is a *perfect matching*. A perfect matching is only possible on graphs of with an even number of vertices.

An *alternating chain* with respect to a matching, M, is a path whose edges alternate between matched and unmatched edges. M admits an alternating chain if and only if M is not maximal.

A *vertex cover* is a subset, $S \subseteq V$ such that every edge in E is incident to at least one vertex in S.

For a graph, $G = (V,E)$, if $M \subseteq E$ is a matching and $S \subseteq V$ is a vertex cover, then $|M| \leq |S|$. This also implies that the size of a maximum matching is at most the size of a minimum vertex cover.

Kőnig's Theorem: In any bipartite graph, $|M| = |S|$ for a maximum matching M and minimum vertex cover S.

A subset of vertices $S \subseteq V$ is an *independent set* of the graph if there are no edges between any pair of vertices in S.

Consider $G = (V,E)$ and let $S \subseteq V$. S is a vertex cover of G if and only if $V \setminus S$ is an independent set.

The distance between two vertices, u and v, written as $d(u,v)$, is the length of the shortest path from u to v. On an undirected graph,

- $d(u,v) = 0 \Leftrightarrow u = v$ (Identity of Indiscernibles);
- $d(u,v) = d(v,u)$ (Symmetry);
- $d(u,v) + d(v,w) \geq d(u,w)$ (Triangle Inequality).

thus satisfying the requirements for a metric. A graph, along with this definition of a distance function, is a metric space.

Characterisation of Bipartite Graphs: G is bipartite if and only if every closed walk in G is of even length.

A *cut* is a partition of the vertex set of a graph into two disjoint sets, L and R. An edge is in the cut (L,R) if it connects a vertex in L with a vertex in R. The value of the cut, (L,R) is the number of edges in the cut.

If $G = (V,E)$ is a graph, then there exists a cut in G with value at least $\frac{|E|}{2}$.

3 Combinatorics

3.1 Hall's Condition

Consider a family of sets, S, with $A_1, A_2, \cdots A_n \subseteq S$. A *system of distinct representatives* (an *SDR*) is a set of distinct elements, $\{x_1,x_2, \cdots ,x_n\} \subseteq S$, such that for all $i \in [1,n]$, $x_i \in A_i$.

A family of sets, S, satisfies *Hall's condition* if, for each subfamily $W \subset S$, we have,

$$|W| \leq \left| \bigcup_{A \in W} A \right|$$

A family of sets admits an SDR if and only if Hall's condition is satisfied.

In words, there exists an SDR for a family of sets, $A_1, A_2, \cdots A_n$ if the union of any k of these sets contains at least k elements for all k from 1 to n.

In terms of a graph, a bipartite graph $G = (L \cup R,E)$ has a perfect matching $M \subseteq E$ of size $|L|$ if and only if, for every subset, A, of L, $|N(A)| \geq |A|$. Considering the elements of L as sets and the elements of R as elements, this graph theoretic version is isomorphic to the set family statement.

3.2 Pigeonhole Principle

Pigeonhole principle: If n elements are partitioned into m non-empty sets, with $n > m$, then at least set must contain more than one element.

More formally, a function whose codomain is smaller than domain cannot be injective.

Example. Let nine points be placed inside a square of side length 1, with no three points lying on the same line. Prove that it is always possible to select 3 points that form a triangle with an area of at most $\frac{1}{8}$.

Divide the unit square into 4 subregions of area $\frac{1}{4}$; for simplicity, and without loss of generality, let these regions be squares of side length $\frac{1}{2}$.

As there are 9 points, and 4 squares, there will always be at least one square containing at least 3 points by the pigeonhole principle (note: a point that lies on the edge of the square can be considered to be contained within that square). Selecting these three points within the square to be the vertices of a triangle, the entire triangle must be fully contained within that square.

The largest area it can be is half the area of the square. As the square has area $\frac{1}{4}$, it follows that the area of the triangle is at most $\frac{1}{8}$, as required.

Example. Consider the complete graph on 6 vertices. Colour each edge either red, or blue. Prove that, no matter how the edges are coloured, the graph will always contain a triangle with all three sides the same colour.

Consider a particular vertex of the K_6 graph. There are 5 vertices adjacent to the selected vertex, and so, by the pigeonhole principle, at least three of the incident edges are of the same colour. Without loss of generality, assume that this colour is red. If any of the edges connecting those three vertices are red, a red monochromatic triangle including the original vertex is formed. If none are red, then all three must necessarily be blue, forming a blue monochromatic triangle.

Exercise. Can you extend this proof to show that a monochromatic triangle must always exist when colouring a complete graph on 17 vertices with 3 colours?

This last example can also be stated in terms of cliques, where, instead of colouring edges red or blue, we include or exclude edges from a graph on 6 vertices, and want to prove that at least one 3-clique or 3-independent set exists. (The exercise, however, cannot. Cliques and independent sets are only equivalent for two-colour cases.)

3.3 Ramsey Numbers

For every $k,l \in \mathbb{N}$, $R(k,l)$ denotes the smallest positive integer such that every graph on $R(k,l)$ vertices contains a k-clique or an independent set of size l.

As shown in the last example of the previous section, we know that we always have a 3-clique or a 3-independent set on a graph with 6 vertices, so we have proved that $R(3,3) \leq 6$ (we haven't proved that

smaller graphs don't have this property, so this is just an upper bound.). However, we can easily prove $R(3,3) = 6$ by producing counterexamples for smaller graphs.

Clearly, $R(1,n) = R(n,1) = 1$ for all $n \in \mathbb{N}$, as the single vertex in the trivial graph simultaneously satisfies the condition for a 1-clique and a 1-independent set.

By symmetry, $R(k,l) = R(l,k)$ for any $k,l \in \mathbb{N}$. If every graph on n vertices satisfies $R(k,l)$, then every such graph contains an k-clique or a l-independent set. It follows that the complement of every graph then contains a l-clique or an k-independent set.

$R(k,l) \leq \binom{k+l-2}{k-1} \leq 2^{k+l}$ for all $k,l \in \mathbb{N}$.

$R(k,k) \leq \binom{2k-2}{k-1} \leq 4^k$ for all $k \in \mathbb{N}$.

If n,k are natural numbers satisfying $\binom{n}{m}2^{1-\binom{k}{2}} < 1$, then $R(k,k) > n$.

4 Discrete Probability

A *probability space* consists of three elements:

- A *sample space*, Ω, which is the set of all possible outcomes;
- An *event space*, a family of sets $\mathcal{F} \subseteq \mathcal{P}(\Omega)$, with each set representing an *event*;
- A *probability function*, $\mathbb{P} : \mathcal{F} \to [0,1]$, such that
  - $\mathbb{P}(\Omega) = 1$
  - $\mathbb{P}(\emptyset) = 0$
  - If $\{A_i\}_{i=1}^{\infty} \subseteq \mathcal{F}$ are countably many disjoint events, then $\mathbb{P}\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} \mathbb{P}(A_i)$.

and a probability space is *discrete* if Ω is at most countably infinite. An event is *elementary* if it is a set of size 1.

4.1 Boole's Inequality

If $\{A_i\}_{i=1}^{\infty} \subseteq \mathcal{F}$ are countably many events, then

$$\mathbb{P}\left(\bigcup_{i=1}^{\infty} A_i\right) \leq \sum_{i=1}^{\infty} \mathbb{P}(A_i)$$

Two events, A and B, are *independent* if $\mathbb{P}(A \cap B) = \mathbb{P}(A)\mathbb{P}(B)$. A finite set of events is *pairwise independent* if every pair of events in the set is independent. A finite set of events is *mutually independent* if every event is independent from every other set and every intersection of every other event.

4.2 Bayes' Theorem

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(B|A)\mathbb{P}(A)}{\mathbb{P}(B)}$$

for any events, A and B. If A and B are independent, then this reduces to $\mathbb{P}(A|B) = \mathbb{P}(A)$.

4.3 Law of Total Probability

If A is an event that can be written as a countable partition, $A = \{B_i\}_{i=1}^{\infty}$, then

$$\mathbb{P}(A) = \sum_{i=1}^{\infty} \mathbb{P}(A \cap B_i)$$

or equivalently,

$$\mathbb{P}(A) = \sum_{i=1}^{\infty} \mathbb{P}(A|B_i)\mathbb{P}(B_n)$$

4.4 Expected Value

The *expected value* of a random variable, X, is the weighted average of all possible values of X.

$$\mathbb{E}(X) = \sum_{i=1}^{\infty} x_i p_i$$

where x_i are the possible values of X, and p_i are their corresponding probabilities of occurrence.

Expectation is linear, so,

$$\mathbb{E}\left(\sum_{i=1}^{n} c_i X_i\right) = \sum_{i=1} c_i \mathbb{E}(X_i)$$

Markov's Inequality: If X is a random variable, and $a > 0$, then,

$$\mathbb{P}(x \geq a) \leq \frac{\mathbb{E}(X)}{a}$$

4.5 The Probabilistic Method

4.5.1 First Moment Method

If X is a non-negative integer-valued random variable, we can find a lower bound for $\mathbb{P}(X = 0)$ using Markov's inequality. Since X takes integer values, $\mathbb{P}(X > 0) = \mathbb{P}(X \geq 1)$, so $\mathbb{P}(X \geq 1) \leq \mathbb{E}(X)$.

4.5.2 Second Moment Method

Similarly,

$$\mathbb{P}(X > 0) \geq \frac{(\mathbb{E}(X))^2}{\mathbb{E}(X^2)}$$

4.5.3 Lovász Local Lemma

We can (non-constructively) prove the existence of a structure with some desired property by proving that the probability of that property occuring in a random structure is greater than zero, or, equivalently, by proving that the probability of that property not occuring in a random structure is less than one.

Example. Let n,m,$d \in \mathbb{N}$. Suppose a town with n people contains m clubs, each of which contains exactly d members. Any person can be a member of multiple clubs, and there may also be people who are not members of any club.

Prove that, if $m < 2^{d-1}$, then there is always a way to partition the town into two sets in such a way that no club has all its members completely contained in either set.

Let Ω be the set of clubs, and suppose that $m < 2^{d-1}$.

Randomly assign each person to one of the two sets of the partition with equal probability $\frac{1}{2}$ of each.

For each club, $C \in \Omega$, let X_C be the event that C is contained entirely within one set of the partition. $\mathbb{P}(X_C)$ is the probability that every person $c \in C$ is assigned to the same set, multiplied by two, as there are two possible sets to be contained within. So,

$$\begin{aligned}\mathbb{P}(X_C) &= \frac{1}{2^d} \times 2 \\ &= \frac{1}{2^{d-1}}\end{aligned}$$

The probability that at least one club is a subset of one of the partition sets is therefore given by, $\mathbb{P}\left(\bigcup_{C\in\Omega} X_c\right)$ which can be bounded above by Boole's inequality.

$$\begin{aligned}\mathbb{P}\left(\bigcup_{C\in\Omega} X_c\right) &\leq \sum_{C\in\Omega} \mathbb{P}(X_c) \\ &\leq \frac{m}{2^{d-1}}\end{aligned}$$

As m is less than 2^{d-1}, $\frac{m}{2^{d-1}} < 1 \implies \mathbb{P}\left(\bigcup_{C\in\Omega} X_c\right) < 1$, so the probability that a club is entirely contained within one set of the partition is less than 1 when $m < 2^{d-1}$. It follows that, if $m < 2^{d-1}$, there exists at least one partition such that no club has all its members completely contained within one set of the partition.