

Contents

1	DIS 0B	2
1.1	Intro	2
1.1.1	Some CS70 advice	2
1.2	Propositional Logic	2
1.3	Proofs	2
1.3.1	Direct proof	3
1.3.2	Contraposition	3
1.3.3	Contradiction	3
1.3.4	Cases	3
2	DIS 1A	4
2.1	Induction	4
2.1.1	(Weak) Induction	4
2.1.2	Strengthening the Hypothesis	4
2.1.3	Strong Induction	4
2.1.4	Weak vs Strong	4
3	DIS 1B	5
3.1	Stable Matching	5
3.1.1	The Propose and Reject Algorithm	5
3.1.2	Stability	5
3.1.3	Optimality	5
3.1.4	Potpourri	6
4	DIS 2A	7
4.1	Graphs	7
4.1.1	Notation	7
4.1.2	Vocabulary	7
4.1.3	The holy grail for graph proofs	8
4.1.4	Relevant Potpourri	8
5	DIS 2B	10
5.1	Trees	10
5.2	Planarity	10
5.3	Coloring	10
5.4	Hypercubes	11

1 DIS 0B

1.1 Intro

- My OH is Monday 1-2 and Tuesday 3-4 in Cory 212.
- Email is first.last@
- 3rd year cs + math major
- hobbies?

1.1.1 Some CS70 advice

- Goal: enhance problem solving techniques/approach
- Don't fall behind on content, catching up will not be fun
- problems, problems, more problems
- Ask lots of questions (imperative for strong foundation)
- Don't stress, we're in this ride together

1.2 Propositional Logic

Relevant notation:

- \wedge = and
- \vee = or
- \neg = not
- \implies = implies
- \exists = there exists
- \forall = forall
- \mathbb{N} = natural numbers $\{0, 1, \dots\}$
- $a|b$ = a divides b

$P \implies Q$ is an example of an implication. We can read this as "If P , then Q ." An implication is false only when P is true and Q is false. If P is false, the implication is vacuously true.

Definition 1.1 (Contrapositive)

If $P \implies Q$ is an implication, then the implication $\neg Q \implies \neg P$ is known as the **contrapositive**.

An important identity is that $P \implies Q \equiv \neg Q \implies \neg P$.

1.3 Proofs

Induction will be in its own section.

Different methods.

1.3.1 Direct proof

Want to show $P \implies Q$ by assuming P and logically concluding Q .

1.3.2 Contraposition

Want to show $P \implies Q$ by equivalently proving $\neg Q \implies \neg P$.

1.3.3 Contradiction

Want to show P . We do this by assuming $\neg P$ and concluding $R \wedge \neg R$.

Why? Idea is that if we can show the implication $\neg P \implies (R \wedge \neg R)$ is True, this is the same as showing $\neg P \implies F$ is True. The contraposition gives $T \implies P$.

1.3.4 Cases

Break up a problem into multiple cases i.e. odd vs even.

2 DIS 1A

2.1 Induction

Goal of induction is to show $\forall n P(n)$.

2.1.1 (Weak) Induction

- Prove $P(0)$ is true (or relevant base cases), then $\forall n \in \mathbb{N} (P(n) \implies P(n+1))$.
- Induction dominoes analogy!
- Sometimes you might have multiple base cases (Problem about $4x + 5y$ in Notes 3)

2.1.2 Strengthening the Hypothesis

Sometimes proving $P(n) \implies P(n+1)$ is not straightforward with induction. In such a scenario, we can try to introduce a (stronger) statement $Q(n)$. We want to construct Q such that $Q(n) \implies P(n)$. Inducting on Q proves P .

2.1.3 Strong Induction

- Prove $P(0)$ is true (or relevant base cases), then $\forall n ((P(0) \wedge P(1) \wedge \dots \wedge P(n)) \implies P(n+1))$.
- Dominoes analogy, but emphasis on the difference between weak and strong induction (assuming middle domino works vs everything from start to middle).

2.1.4 Weak vs Strong

A common point of confusion is when one should use strong induction in lieu of weak induction. Strong induction **always** works whenever weak induction works. However, there may be scenarios in which the induction hypothesis to prove $n = k + 1$ requires more information than just $n = k$. A scenario like this requires strong induction.

3 DIS 1B

3.1 Stable Matching

Cool application of induction.

3.1.1 The Propose and Reject Algorithm

Suppose jobs proposes to candidates.

- both jobs and candidates have a list of preferences
- every day a job that doesn't have a deal with a candidate will propose to the next best candidate on its preference list
- every candidate will tentatively "waitlist" the offer from the job (put it on a string)
- if a candidate has multiple offers, they will choose the one they prefer the most
- the algorithm ends when every candidate has a job on their "waitlist" (all these WLs becomes acceptances)

(walk through q1 of dis as a class to visualize this)

3.1.2 Stability

Definition 3.1 (Rogue Couple)

A job-candidate pair (J, C) is denoted as a **rogue couple** if they prefer each other over their final assignment in a stable matching instance.

Definition 3.2 (Unstable)

A matching that has at least one rogue couple is considered **unstable**.

Conversely, a **stable** matching is one that has no rogue couples.

Some tricky vocab stuff like stable matching instance.

Lemma 1 (Improvement) *If a candidate has a job offer, then they will always have an offer from a job at least as good as the one they have right now.* \square

Matchings produced by the algorithm are always **stable**.

3.1.3 Optimality

The propose and reject algorithm is proposer *optimal* and receiver *pessimal*.

Definition 3.3 (optimal)

A pairing is optimal for a group if each entity is paired with who it most prefers while maintaining stability.

Can be thought of a (well that's the best I could do) analogy.

Definition 3.4 (pessimal)

A pairing is pessimal for a group if each entity is paired with who it least prefers while maintaining stability.

Can be thought of a (well it can't get worse than this) analogy.

3.1.4 Potpourri

It is possible that there exists a stable matching instance that is neither job optimal nor candidate pessimal.

Consider the following preferences

Jobs	Preferences	Candidates	Preferences
<i>A</i>	1 > 2 > 3	1	<i>B</i> > <i>C</i> > <i>A</i>
<i>B</i>	2 > 3 > 1	2	<i>C</i> > <i>A</i> > <i>B</i>
<i>C</i>	3 > 1 > 2	3	<i>A</i> > <i>B</i> > <i>C</i>

The matching above can generate (at least) 3 stable matching instances

$$S = \{(A,1), (B,2), (C,3)\}$$

$$T = \{(A,3), (B,1), (C,2)\}$$

$$U = \{(A,2), (B,3), (C,1)\}.$$

We see

- *S* is job-optimal/candidate-pessimal (result of running propose and reject with jobs proposing to candidates)
- *T* is candidate-optimal/job-pessimal (result of running propose and reject with candidates proposing to jobs)
- *U* is neither optimal nor pessimal for both candidates and jobs (*S* and *T*) corroborate that.

Also some other important facts that can be seen (from discussion worksheet questions):

- There is at least one candidate that will receive only one proposal (that too on the last day)
- We can upper bound the number of days needed by P&R algorithm to $(n-1)^2 + 1 = n^2 - 2n + 2$ (think about why)
- As a consequence of above, we can upper bound the number of rejections needed by P&R algorithm to $(n-1)^2 = n^2 - 2n + 1$ rejections.

4 DIS 2A

4.1 Graphs

4.1.1 Notation

- V denotes set of vertices (points)
- E denotes set of edges (lines)
- $|V|$ denotes size of set of vertices i.e number of vertices; $|E|$ similarly
- Graph G with vertices V and edges E is denoted $G = (V, E)$.

4.1.2 Vocabulary

Definition 4.1 (Path)

A **path** is a sequence of edges. In CS70, we assume a path is *simple* which means no repeated vertices.

Definition 4.2 (Cycle)

A **cycle** is a simple path that starts and ends at the same vertex.

Definition 4.3 (Walk)

A **walk** is any arbitrary connected sequence of edges.

Definition 4.4 (Tour)

A **tour** is a walk that starts and end at the same vertex.

Definition 4.5 (Connected)

A graph is **connected** if there exists a path between any two distinct vertices.

Definition 4.6 (Eulerian Walk)

An **Eulerian walk** is a walk covering all edges without repeating any.

Definition 4.7 (Eulerian Tour)

An **Eulerian tour** is an Eulerian walk that starts and ends at the same vertex.

To summarize,

	no repeated vertices	no repeated edges	start = end	all edges	all vertices
Walk					
Path	✓	✓			
Tour			✓		
Cycle	✓*	✓	✓		
Eulerian Walk		✓		✓	
Eulerian Tour		✓	✓	✓	
Hamiltonian Tour	✓	✓	✓		✓

(*except for start and end vertices)

Theorem 4.1 (Euler's Theorem)

An undirected graph G has an Eulerian tour iff G is connected and all its vertices have even degree.

The requires condition for an Eulerian walk is that we have exactly 2 vertices of odd degree. (Of course, the case of 0 odd vertices trivially works since we claim from Euler's Theorem that we can find an Eulerian tour which is a stronger statement than an Eulerian walk)

Definition 4.8 (Bipartite)

A graph is considered bipartite if V can be partitioned into two sets L and R where $V = L \cup R$ such that there are no edges between vertices in L and no edges between vertices in R .

4.1.3 The holy grail for graph proofs

Induct, induct, induct, and induct.

- Think about what you want to induct on (edges or vertices???)
- Base case (read the problem carefully!)
- Prove for n by going from $n \rightarrow n - 1 \rightarrow I.H. \rightarrow n$.
 - **DO NOT** go from $n - 1 \rightarrow n$ directly.
 - Why? Build-up error!
 - Good example of build-up error when trying to prove “if every vertex of a graph has degree at least 2, then there exists a cycle of length 3.” Any attempt at induction will give us a false proof but we cannot make square from triangle!
 - It's also a logistical nightmare lol (in the times it might accidentally work). Try generating all 5-vertex trees from all 4-vertex trees yikes.

4.1.4 Relevant Potpourri

Some other relevant information.

Definition 4.9 (Degree)

The **degree** of a vertex v denoted $\deg(v)$ is defined to be the number of incident edges to v .

Lemma 2 (Handshake)

$$\sum_{v \in V} \deg(v) = 2|E|.$$

□

The idea of a degree (with no adjective) is only well-defined for undirected graphs. We see for directed graphs it's a little funky; we need to introduce the concept of indegree and outdegree.

In a directed graph, the number of outgoing edges equals the number of ingoing edges.

We will discuss trees, planarity, coloring, and hypercubes in the next discussion.

5 DIS 2B

5.1 Trees

A graph $G = (V, E)$ is a Tree if any of the statements below is true. TFAE (The following are equivalent):

- G is connected and has no cycles
- G is connected and $|E| = |V| - 1$
- G is connected and removing a single edge disconnects G
- G has no cycles and adding a single edge creates a cycle

Definition 5.1

A leaf is a node of degree 1.

A consequence of above is that every tree has at least 2 leaves.

5.2 Planarity

Definition 5.2 (planar)

A graph is **planar** if it can be drawn without any edge crossings.

Theorem 5.1 (Euler)

For every connected planar graph, $f + v = e + 2$.

Corollary 1 *If a graph is planar, then $e \leq 3v - 6$.* □

Theorem 5.2 (Kuratowski)

A graph is non-planar iff it contains K_5 or $K_{3,3}$.

(draw the two above graphs on the board)

The notation K_x denotes a complete graph with x vertices.

Definition 5.3 (complete graph)

A **complete graph** is a graph where all possible edges exist. Formally, in graph $G = (V, E)$, for any distinct $u, v \in V$, then $\{u, v\} \in E$.

5.3 Coloring

Two types: edge and vertex

- edge: color edges so that no two adjacent edges have the same color
- vertices: color vertices so that no two adjacent vertices have the same color

Theorem 5.3 (4 color theorem)

If a graph is planar, then it can be colored with 4 (or less) colors.

5.4 Hypercubes

A hypercube of dimension n is a graph whose vertices are bitstrings of length n . An edge between two vertices exists iff the two vertices differ at exactly 1 bit.

(draw $n = 1, 2, 3$ on the board)

We can see that $|V| = 2^n$ and $|E| = n2^{n-1}$.

Give some motivation on induction on hypercubes.