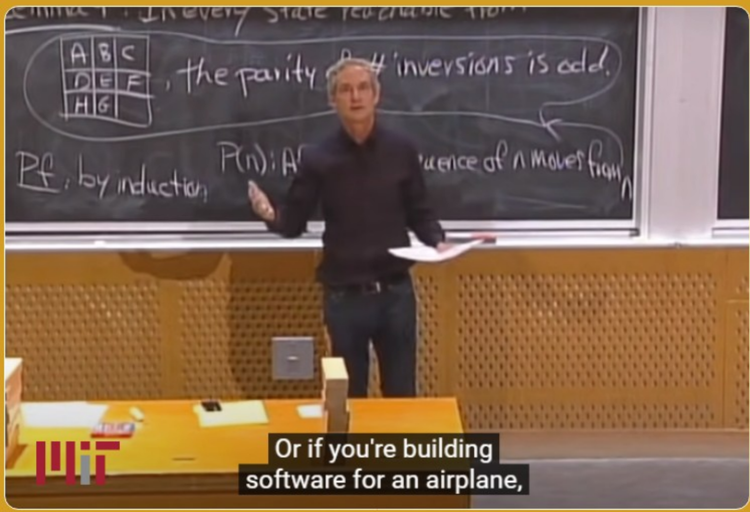


Lec 3 | MIT 6.042J Mathematics for Computer Science
Instructor: Tom Leighton
Subject: Invariant & Strong Induction

ملخص المحاضرة الثالثة من كورس Mathematics for computer science لمعهد الـMIT بعنوان **Strong Induction.**

Invariant

واحد من أهم الإستخدامات للـ induction في الـ computer science هي انك تثبت وتتأكد من ان الـ Program او الـ Algorithm بتاعك بياحفظ على Properties معينة طول ما هو شغال، الخاصية اللي بيفضل الـ Program بتاعك محافظ عليها طول ما هو شغال بتسمى Invariant، تعالى ناخد أمثلة كده:
لو بتبني سوفتوير لإدارة مفاعل نووي، في حالة انت هتكون مش عايز توصلها وهي الـ Meltdown وكمان هتكون عايز تثبت أن أي تسلسل من العمليات اللي بيعملها السوفتوير بتاعك مش هيوصل المفاعل للـ meltdown state يعني مثلا هتكون عايز درجة حرارة المفاعل متتخطاش حد معين، كذلك الأمر لو بتبني سوفتوير لطائرة فأنت مش عايز توصل لإن الطيارة تتحطم مثلا زي انها متنزلش تحت أرتفاع ألف قدم من غير ما تنشر معدات الهبوط، او لو بتبني جهاز أشعة فأنت مش عايز جهاز الأشعة ده يضرب المريض بالإشعاع لحد ما يموته، (أبحث عن جهاز Therac-25) ، او بشكل عام لو عندك variable معين وعايز القيمة بتاعته متتخطاش حد معين، الكلام ده كله بيتعمل عن طريق الـ Invariants.

طيب ايه دخل الـ induction بالليلة دي؟ الـ induction بستخدمه عشان أثبت ان proposition معينة اللي هي هنسميها invariant تكون true في البداية (base case)، كمان true بعد عدد معين من الخطوات وليكن t وكمان هتكون true بعد t+1 ودي كده (inductive step) ومن هنا نقدر بإستخدام الـ induction نثبت ان فعلا الـ proposition دي invariant يعني هتكون دائما true!

بطريقة تانية الـ invariant مفيد لما يكون عندي system له حالة إبتدائية start state ، وله عدد خطوات well defined ممكن تغير حالته وأنا عايز أتأكد ان في خاصية معينة كانت موجودة في الـstart state وعدد الخطوات اللي بتحصل للـsystems ده مش هتغير الخاصية دي.

ناخد مثال عشان نفهم الدنيا:

Example 1: The Diagonally-Moving Robot

لو أفترضنا عندنا روبوت بيتحرك على الأقطار فقط على سطح غير نهائي ثنائي الأبعاد ومكان الروبوت بيتحدد عن طريق قيمتين (x, y) ومكان اللي بيبدأ الروبوت منه الحركة هو نقطة الأصل (0 ,0) والروبوت ده في كل حركة له بيتحرك one unit فقط وبما انه بيمشي على الأقطار وبدأ من (0 ,0) فبعد خطوة واحدة ممكن يكون عند (1 ,1) , (1- ,1-) , (1- ,1) , (1 ,1-)، السؤال هنا هل ممكن للروبوت انه يوصل للمكان ده (0 , 1)؟

لو قعدت تجرب بنفسك كده بالورقة والقلم هتلاقي ان مستحيل الروبوت يوصل للمكان ده بسبب ان الروبوت يقدر بس يوصل لمكان فيه مجموع أحداثياته x + y عدد زوجي وده اللي احنا هنحاول نثبته، يعني نقدر نعمل predicate بالشكل ده

p(t): if the robot is in state (x, y) after t steps, then x + y = even.

Theorem 1: The sum of robot’s coordinates is always even.

تعالى نثبت الكلام ده عن طريق الـ induction، طب ليه الـ induction؟ عشان انا عندي start state اهو للروبوت وهي ان مجموع أحداثياته في البداية عدد زوجي 0 + 0 والسؤال بيقول هل ينفع انه الروبوت بتاعي يوصل لإحداثيات (1, 0) ؟ اللي هم مجموعهم عدد فردي، وبالتالي لو انا أثبت ان مجموع أحداثيات الروبوت هيفضل دايما زوجي او بمعنى اخر أثبت ان الـ p(t) دايما بـ true إذا وقتها نقدر نستنتج ان مستحيل يوصل للإحداثيات دي (1, 0).

نبدأ أول بأننا نحدد للي هيقرأ ن ده إثبات بالـ induction وبعدين الـ base case وبعدها الـ inductive step:

Proof by induction

Base case: P(0) is true since the robot start at (0, 0) and after 0 steps x + y = 0 which is even.
Inductive step:

زي ما احنا متعودين في الحالة دي أنا بحاول أثبت أن p(t) implies p(t+1) وبالتالي بفترض ان الـ p(t) بـ true وبحاول أثبت ان الـ p(t+1) بـ true، طيب لو أفترضنا ان الـ p(t) بـ true ده كده معناه ان الـ x + y عدد زوجي، طيب عشان اثبت p(t+1) فأنا عندي 4 حالات:

الحالة الأولى: الروبوت هيتحرك للمكان ده (x + 1, y + 1) وقتها مجموع الإحداثيات هيكون x + y + 2 وده عدد زوجي لأن الـ x + y عدد زوجي ولو جمعت عليهم 2 هيفضل زوجي وبالتالي p(t+1) هتكون true هنا.

الحالة الثانية: الروبوت هيتحرك للمكان ده (x - 1, y - 1) وقتها مجموع الإحداثيات هيكون x + y - 2 وده عدد زوجي لأن طرح 2 من اي عدد زوجي هيفضل زوجي وبالتالي p(t+1) هتكون true هنا.

الحالة الثالثة: الروبوت هيتحرك للمكان ده (x - 1, y + 1) وقتها مجموع الإحداثيات هيكون x + y وده عدد زوجي زي ما أفترضنا وبالتالي p(t+1) هتكون true هنا.

الحالة الرابعة: الروبوت هيتحرك للمكان ده (x + 1, y - 1) وقتها مجموع الإحداثيات هيكون x + y وده عدد زوجي وبالتالي p(t+1) هتكون true هنا.

وبالتالي هنا في كل الحالات الممكنة p(t + 1) بـ true يعني معنى كده اننا أثبتنا خلاص ان

وبالتالي عن طريق الـ induction نقدر نقول أن الـ p(t) بـ true لكل قيم الـ t الأكبر p(t) implies p(t+1) من او تساوي 0.

P(t) is true for all t >= 0.

وبالتالي عن طريق الإثبات ده نقدر نقول ان الروبوت مستحيل يوصل لـ (1, 0) لان مجموع الإحداثيات هنا فردي، واحنا اثبتنا اننا لو بدأنا من (0, 0) فمجموع الإحداثيات لأي مكان ممكن يوصله الروبوت هيكون دائما زوجي وأنتهى الإثبات.

تعالى نشوف مثال تاني أصعب شوية:

Example 2: The 8-Puzzle

Figure 1

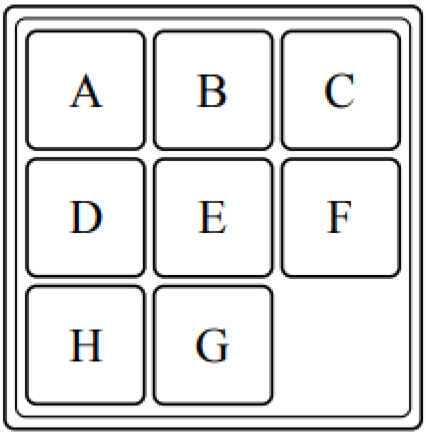
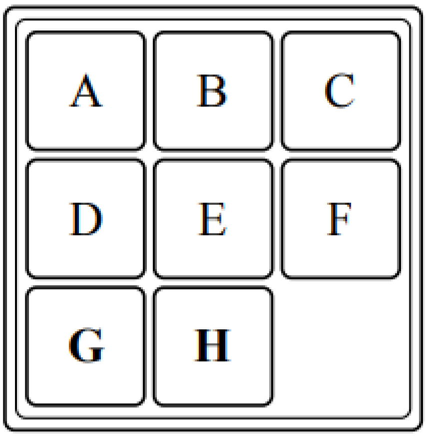


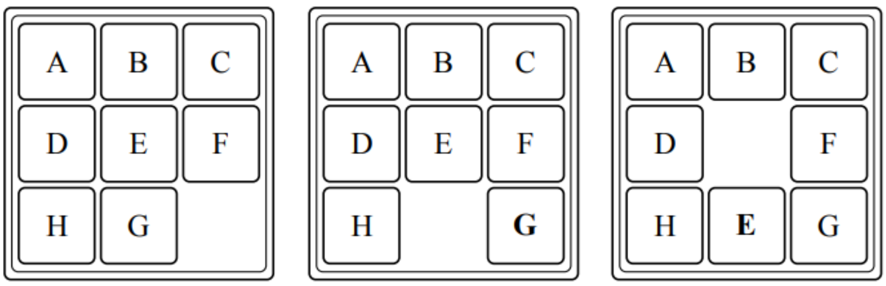
Figure 2



عندي grid متقسم 3 * 3 عبارة عن 9 مربعات، وموجود عليه 8 حروف مترتبة ترتيب أبجدي بإستثناء اخر حرفين فقط اللي هم الـ G, H زي ما باين في الصورة اللي على الشمال Figure 1 هتلاقي الـ H قبل الـ G والمفروض يكون العكس، والترتيب الصحيح هو اللي موجود في الصورة اللي على اليمين، وبالتالي عندك كده مربع واحد فاضي تقدر تحرك فيه باقي الحروف التانية ومسموحلك بحركتين، اول حركة هي حركة صف row move وده معناه انك تقدر تحرك حرف في المكان الفاضي سواء كان على يمينه او شماله، الحركة التانية column move هي حركة عموديا ودي مسموحلك تحرك الحرف في المكان الفاضي سواء لفوق او لتحت، ومش مسموحلك بحركة قطرية، من الأخر فوق وتحت شمال ويمين فقط، طيب ايه المطلوب؟ **المطلوب هو انك تبدل الـ G مع الـ H بحيث يكون الحروف كلها مترتبة ترتيب أبجدي، يعني متبدلش الـ H, G وتبوظ باقي الترتيب.**

الصورة اللي تحت بتوضح اللعبة في الوضع الإبتدائي والصورتين اللي جمبها بتوضح الـ row move والـ column move:

Figure 1



(a) (b) (c)

ملحوظة: حاول ترسم المربعات على ورقة وقطع ورقة مربعة صغيرة لكل حرف وحاول تلعب اللعبة مع نفسك كده وشوف هتقدر توصل لأيه، بس حرقا للإحداث مستحيل تقدر تبدل الحرف الـ G مكان الـ H وفي نفس الوقت تحافظ على ترتيب باقي الحروف وهنستخدم الـ invariant والـ induction عشان نثبت الكلام ده.

طيب ليه الـ invariant؟ لو انت عندك system وعايز تثبت انه مستحيل يوصل لـ special state فأنت كل اللي محتاج تعمله هو أنك تلاقي property معينة (invariant) وأثبت انها بتكون true في البداية وكمان true مع كل legal move يعني الخاصية دي هتكون true معاك دايما ضمن الـconditions اللي انت حاطتها وبعدين شوف هل الخاصية دي هتكون true ولا لأ في الـ special state ، لو كانت true يبقى تقدر توصلها لو كانت بـ false يبقى مستحيل توصل للـ state دي.

Theorem 2: No sequence of legal moves transforms the configuration in Figure 1 into the configuration in Figure 2.

عشان نثبت الموضوع ده فأحنا هنقسم الإثبات على كذا حته اصغر اسمها Lemma ، طب يعني ايه Lemma ؟ دي بتكون proposition بردو بثبتها عشان بتساعدني في النظرية اللي انا بحاول أثبتها، الموضوع بالظبط كده عامل زي في البرمجة لما بتقسم الـ program بتاعك لأكثر من function كل واحدة بتقوم بوظيفة معينة وفي الأخر كلهم بيساعدوك في تنفيذ الـ program ككل، كذلك نفس الكلام كده احنا بنثبت شوية حاجات وبنسميهم Lemma ودول هيساعدونا في الإثبات النظرية رقم 2 ككل.

طيب عشان نلاقي الخاصية الـ invariant دي هنعمل analysis للعبة ونشوف في الـ row move والـ column move ايه اللي بيحصل.

<u>**Row move**</u>

Lemma1: row move does not change the order of the item
proof:

في حالة الـ row move لنفترض ان الحرف اللي انت عايز تحركه وليكن في الـ position هنشير له بالحرف j، اذا انت مسمحولك تحركه يا اما j + 1 او j – 1 ، لو بصينا هنلاقي انك في الـ row move مبتغيرش الترتيب بتاع اي حرف من الحروف بالنسبة للحروف التانية! يعني الترتيب بيفضل زي ما هو وبالتالي نقدر نستنتج lemma1 بـ true.

<u>**Column move**</u>

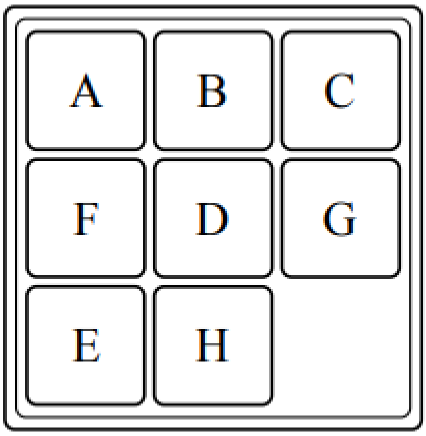
هنا بقى اللي فيه شغل، لو بصيت في Figure 3 فوق في الصورة b, c هتلاقي ان فعلا الـ column move بيغير الترتيب بتاع الحروف وبالتحديد العنصر اللي بحركه بيغير مكانه مع عنصرين تانيين يعني اقدر اعمل lemma جديدة بالشكل ده:

Lemma2: a column move changes the relative order or precisely two pairs of items.
Proof:
لما العنصر بيتحرك حركة عمودية فهو بيتحرك من position وليكن j إلى position تاني بيكون j + 2 او j – 2 وبالتالي لو بصيت على Figure 3 (a) هتلاقي فعلا انه بيبدل مكانه مع عنصريين تانيين اللي هم مكانهم بيكون j – 1 , j – 2 او j + 1 و j + 2.

عايزين نعمل تعريف يوضحلنا حاجه مهمة وهي ان الترتيب عندي بيكون فيه مشكلة لما حرفين يكون واحد فيهم قبل التاني في الترتيب الأبجدي لكن في اللعبة هو موجود بعده.

Def: a pair of letters call L1, L2 is an inversion (inverted pair) if L1 precede L2 in alphabet but L1 is after L2 in puzzle.

يعني الصورة اللي تحت دي تحتوى على تلاتة inversions ، طب مين هم؟
(D, F) , (E, G) , (E , F)



طيب في المشكلة بتاعتنا الأصلية اللي متبدل فيها الـ G, H اللي هي Figure 1 فوق، عندي كام inversion ؟ واحد بس لان الـG متبدله مع الـ H فقط (H, G).

من الكلام ده نقدر نستنتج حاجة خطيرة جدا وهي ان خلال اي حركة عدد الـ inversion يا أما هيزيد بمقدار 2 او هيقل بمقدار 2 او هيفضل زي ما هو! طب تعالى بقى نعمل Lemma جديدة ونثبتها.

Lemma 3: during a move the number of inversions can only increase by 2 or decrease by 2 or stay the same.
proof

أولا مفيش حاجه بتتغير في عدد الـ inversions عند الـ row move وده عن طريق Lemma 1 اللي أثبتناها فوق، أما في حالة الـ Column move ففي 3 حالات:

احنا قولنا ان في حالة الـ column move ففي 2 pairs بيتغير مكانهم

الحالة الأولى انهم كانوا مترتبين: وبالتالي هنا عدد الـ inversions هيزيد بمقدار أتنين لأنهم كانوا مترتبين وانا اما عملت الـ column move بوظت الترتيب.

الحالة التانية أنهم مكنوش مترتبين: وبالتالي هنا عدد الـ inversions هيقل بمقدار أتنين لأنهم كانوا مش مترتبين وانا اما عملت column move رتبتهم.

الحالة الثالثة أن واحد منهم كان مترتب ولكن التاني مكنش مترتب: وبالتالي هنا عدد الـ inversions هيفضل زي ما هو لأن اللي كان مترتب هيبوظ ترتيبه واللي كان مش مترتب هيترتب.

من هنا نقدر نستنتج أستنتاج كبير ومهم جدا وهو ان خلال اي حركة الـ parity الخاصة بالـ number of inversions مبتتغيرش! يعني لو كان عدد الـ inversion زوجي هيفضل زوجي ولو كان فردي هيفضل فردي!

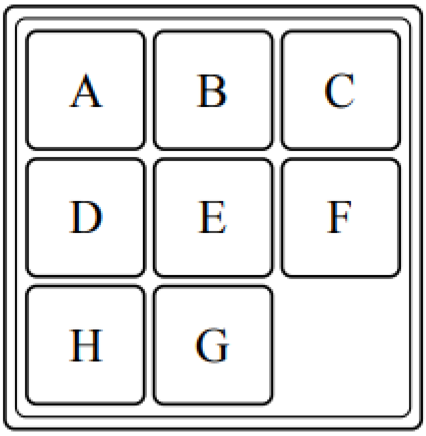
Corollary 1: during a move the parity of number of inversions does not change.
Proof:

الإثبات بسيط وهو طرح او إضافة 2 على اي رقم لا يغير الـ parity بتاعته!

- ثانية واحدة مين corollary ده ؟ دي عبارة عن نظرية بستنتجها على طول من نظرية احنا لسا ثابتينها ، زي ما انت لاحظت ان احنا في lemma 3 أثبتنا ان عدد الـ inversions يا أما بيزيد ب2 او بيقل ب2 او بيفضل زي ما هو، فأستنتجنا من هنا corollary بتقولك ان الـ parity الخاصة بعدد الـ inversions مبيتغيرش.

احنا كده وصلنا للـ invariant اهو، ومن الكلام اللي فوق ده كله نقدر نستنتج ان أي state اقدر أوصلها من الـ state اللي في Figure 1 الـ parity الخاصة بالـ number of inversions هتفضل odd! ليه؟ لأن عدد الـ inversions عندي في البداية واحد فقط ومن الـ corollary 1 اللي وصلنا ليها فالـ parity هتفضل odd وثابته زي ما هي، بس تعالى نشتغل رسمي ونكتب lemma 4 ونثبتها بالـ induction.

Lemma 4: in every state reachable from the state in Figure 1, the parity of the number of inversions is odd.



مرة تانية ليه induction؟ عشان خلاص لقينا invariant قيمتها true في البداية وقيمتها هتفضل true بعد اي move مسموحة ليا وانا عايز أثبت الـ invariant دي قيمتها false عند الstate اللي هو عايزني اوصلها اللي هي انهم ارتبهم كلهم يعني.

Proof by Induction

في البداية زي ما متعودين هنختار predicate

p(n): after any sequence of n moves from the start (Figure 1) , the parity of the number of inversions is odd.

base case: p(0), the number of inversions equal to 1 which is odd, then p(0) is true.

الـ base case عندي هي اني معملتش اي حركة واصلا اللعبة عندي في الحالة الإبتدائية عدد الـ inversions بواحد فهيفضل واحد زي ما هو وبالتالي هو عدد فردي.

Inductive step: let p(n) be true and then show that p(n+1) is true

عدد الـ moves اللي هيحصل عندي في حالة الـ p(n+1) هو n + 1 يعني هيكون عامل بالشكل ده:

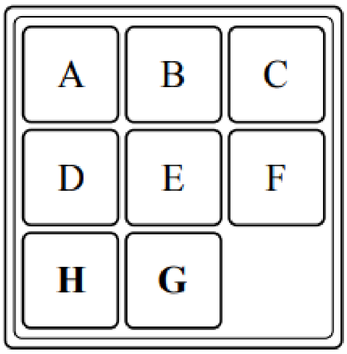
M1, M2, M3, …, Mn, Mn+1

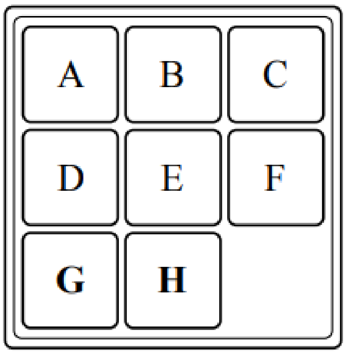
واحنا لسا مفترضيش ان الـ p(n) بـ true وبالتالي من M1 لحد M2 عدد الـ inversion عندي الـ parity بتاعته فردي، واحنا عارفين من الـ corollary 1 اللي أثبتناها فوق ان اي حركة هعملها الـ parity مش هتتغير يبقى اذا بعد ما اعمل الحركة Mn+1 هتفضل الـ parity بتاعة الـ number of inversion زي ما هي فردي وبالتالي p(n+1) كمان بـ true وبكدة نكون أثبتنا أن الـ p(n) implies p(n+1) ومن خلال الـ induction نقدر نقول ان الـ p(n) هتكون true لكل n >= 0.

P(n) is true for all n >= 0

النظرية رقم 2 اللي فوق اللي حاولنا نثبتها هرجع أكتبها هنا مرة تانية:

Theorem 2: No sequence of legal moves transforms the board below on the left into the board below on the right.





الإثبات: ان عدد الـ inversions في الصورة اللي على الشمال بواحد (عدد فردي) وعدد الـ inversion في الصورة اللي على اليمين 0 (عدد زوجي) وبالتالي من خلال الـ lemma 4 مستحيل أقدر اوصل من الشكل اللي على الشمال للشكل اللي على اليمين.

Strong Induction

لو تفتكر في الـ induction اللي خدناه قبل كده (بيسموه Ordinary induction) كنا في الـ inductive step بنفترض ان الـ p(n) بـ true وبعدين بنحاول نثبت ان الـ p(n+1) بـ true وكنا دايما واحنا بنحاول نثبت الـ p(n+1) كنا بنستخدم الـ p(n) وكانت بتساعدنا بشكل كبير جدا وكانت الـ p(n+1) معتمدة على الـ p(n)، طيب أفرض على سبيل المثال أن الـ p(n+1) مش معتمدة على الـ p(n) ولكنها معتمدة على حاجه تانية ولتكن p(j) بحيث ان j أقل من n فهل وقتها أقدر بدل ما أفترض أن الـ p(n) بـ true ، أفترض أن الـ p(j) بـ true؟ هل يحق ليا اعمل الكلام ده ؟ اه تقدر تعمل كده ودي فايدة الـ strong induction بتستخدمه لو الـ p(n+1) بتعتمد على الـ p(j) بحيث ان j اقل من n طب ايه الدليل أن ينفع اعمل كده ؟ الدليل زي ما شرحنا المرة اللي فاتت ان الـ induction ده عامل زي Machine انا بثبت الـ p(0) وبعدين بدخلها للـ machine وهي تثبت الـ p(1) وهكذا لحد ما أوصل لـ p(n) و p(n+1) وبالتالي وبما ان j أقل من n إذا اكيد الـ machine مرت بيها وأثبتت ان الـ p(j) بـ true قبل ما توصل للـ p(n).

Principle of Strong Induction.

Let P(n) be a predicate. If
P(0) is true, and
for all $n \in N$, p(0), p(1), …, p(n) together imply p(n+1),
then p(n+1) is true for all $n \in N$.

ناخد مثال على الكلام ده:

Example: Any positive integer greater than 1 is either a prime or a product of primes.

Proof by strong induction

let p(n): n is product of primes.

Base case: p(2) is true because 2 is prime.
Inductive step:

زي ما قولنا بفضل الـ strong induction مش بس هنفترض ان الـ p(n) هتكون بـ true لا كمان نقدر نفرض كل اللي قبلها لحد الـ Base case وبالتالي دلوقتي هنفترض أن p(3) , p(4) , … p(n) كل دول بـ true وهنحاول نثبت أن الـ p(n+1) كمان بـ true، طيب ازاي نعمل كده؟ احنا معندناش غير حالتين يا أما الـ n+1 تكون عدد أولي وفي الحالة دي p(n+1) هتكون true وخلصنا، يا اما تكون عدد غير أولي ودي الحالة اللي هيكون فيها شغل شوية لأن وقتها لو هي عدد غير أولي فأنا أقدر أمثلها برقمين مضروبين في بعض وليكن k و m بحيث ان أي رقم من الرقمين دول لازم يكون اكبر من او يساوي 2 وأقل من الـ n + 1

$n+1 = km$, where $2 \le k, m < n + 1$

طيب ما احنا لسا مفترضين ان p(n) بـ true وبناء عليه كل الأرقام من n لحد 2 يا أما prime او product of primes وبالتالي الـ k, m ينطبق عليهم نفس الكلام لانهم اقل من n + 1! إذا نستنتج من الكلام ده ان الـ n + 1 هو كمان يا اما هيكون prime او هيكون product of primes وبالتالي p(n+1) هنا بـ true وعن طريق الـ induction نقدر نستنتج ان الـ p(n) هتكون true لكل n اكبر من او تساوي 2.

P(n) holds for all $n \ge 2$

وبكده بفضل الله ده يكون ملخص المحاضرة الثالثة أتمنى يكون مفيد، والسلام عليكم ورحمة الله وبركاته، مع تحياتي: محمد صلاح.

https://www.linkedin.com/in/mohamed-salah-039b35109