

CMPT-335 Discrete Structures

MATHEMATICAL INDUCTION

Mathematical Induction

Suppose we have a propositional function (predicate), which depends on an integer parameter and which we would like to prove to be true:

$P(1), P(2), P(3), P(4), \dots P(n), \dots$

We can picture each predicate as a domino:



$P(i)$

Mathematical Induction

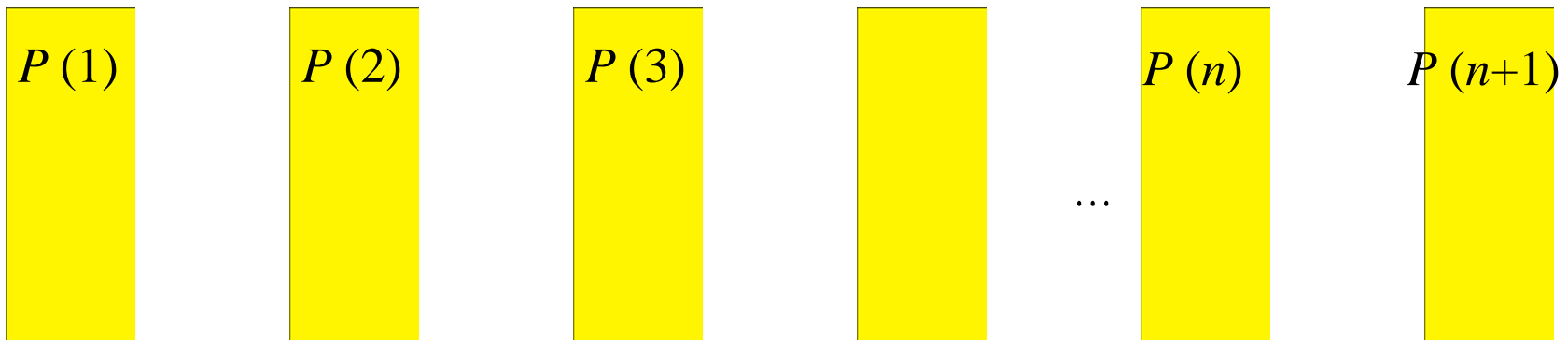
When the domino **falls**, the corresponding predicate is considered **true**:



$P(i)$

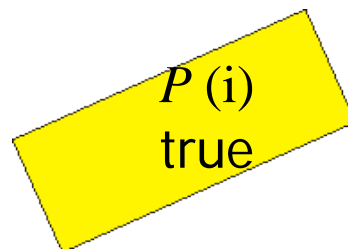
Mathematical Induction

So sequence of predicates is a sequence of dominos.



Mathematical Induction

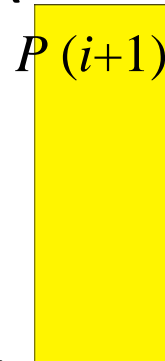
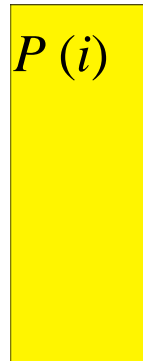
When the domino **falls** (to right), the corresponding predicate is considered **true**:



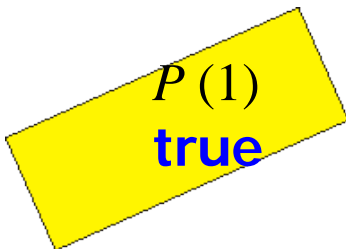
Mathematical Induction

Suppose that the dominos satisfy two constraints.

- 1) Well-positioned: If any domino falls (to right), next domino (to right) must fall as well.



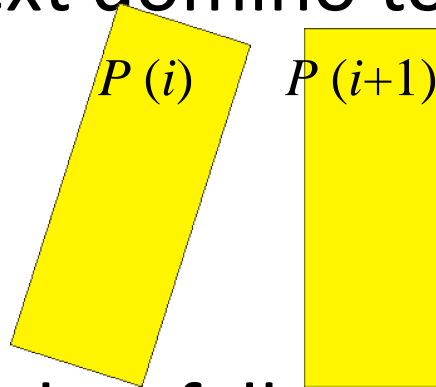
- 2) First domino has fallen to right



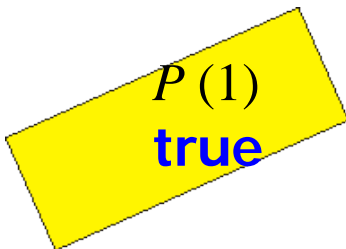
Mathematical Induction

Suppose that the dominos satisfy two constraints.

- 1) Well-positioned: If any domino falls to right, the next domino to right must fall as well.



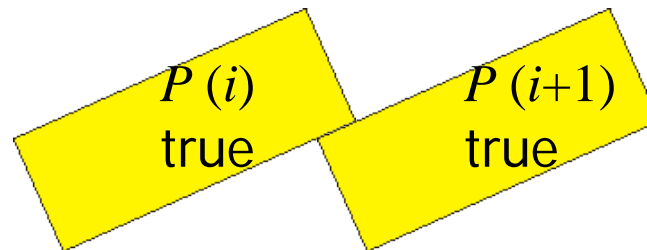
- 2) First domino has fallen to right



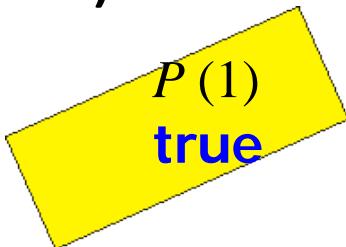
Mathematical Induction

Suppose that the dominos satisfy two constraints.

- 1) Well-positioned: If any domino falls to right, the next domino to right must fall as well.

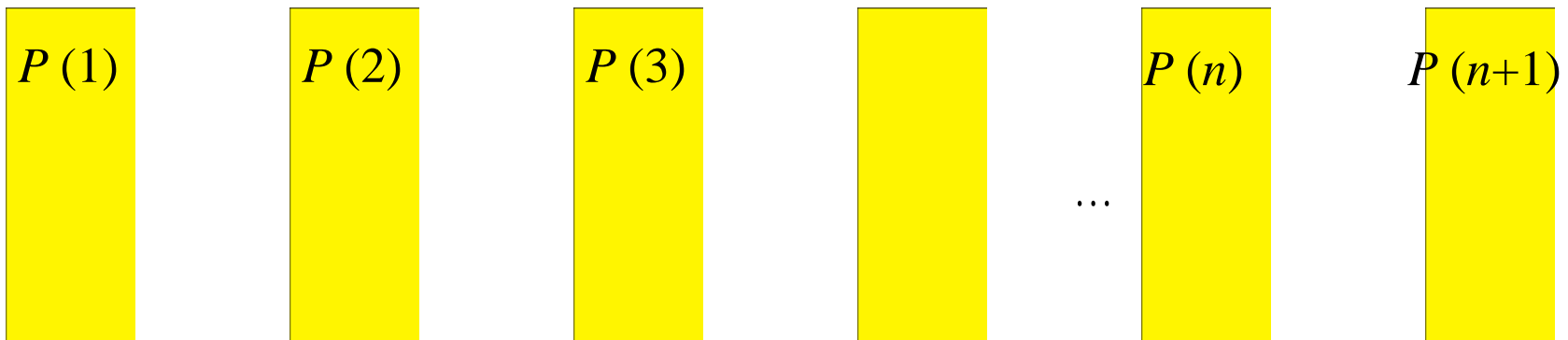


- 2) First domino has fallen to right



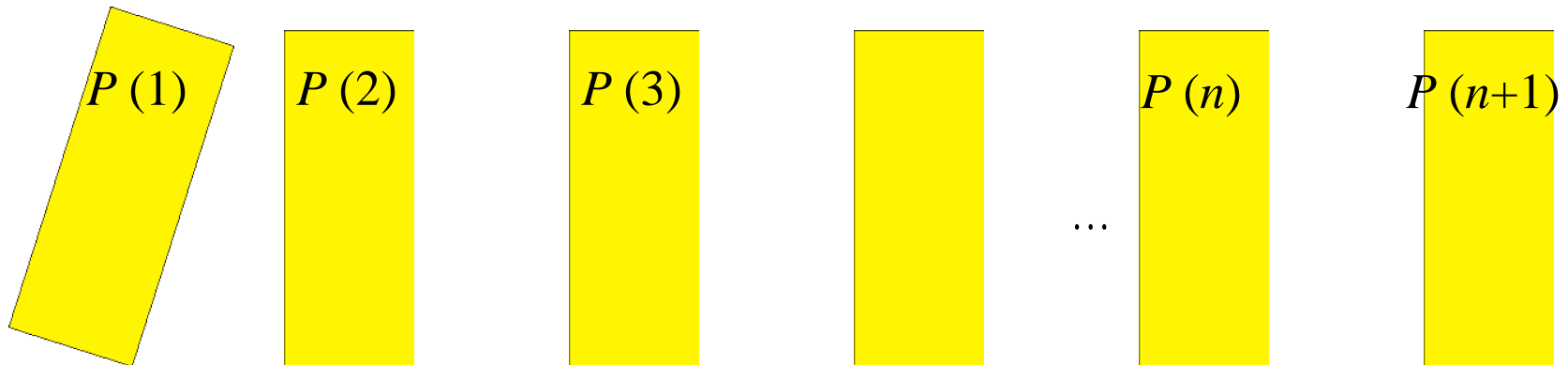
Mathematical Induction

Then we can conclude that all the dominos fall!



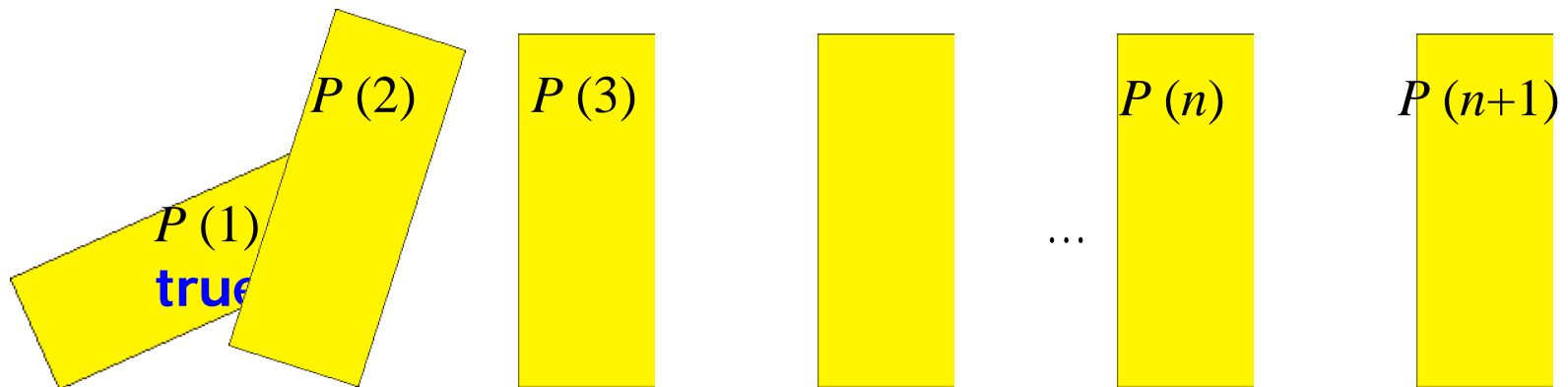
Mathematical Induction

Then we can conclude that all the dominos fall!



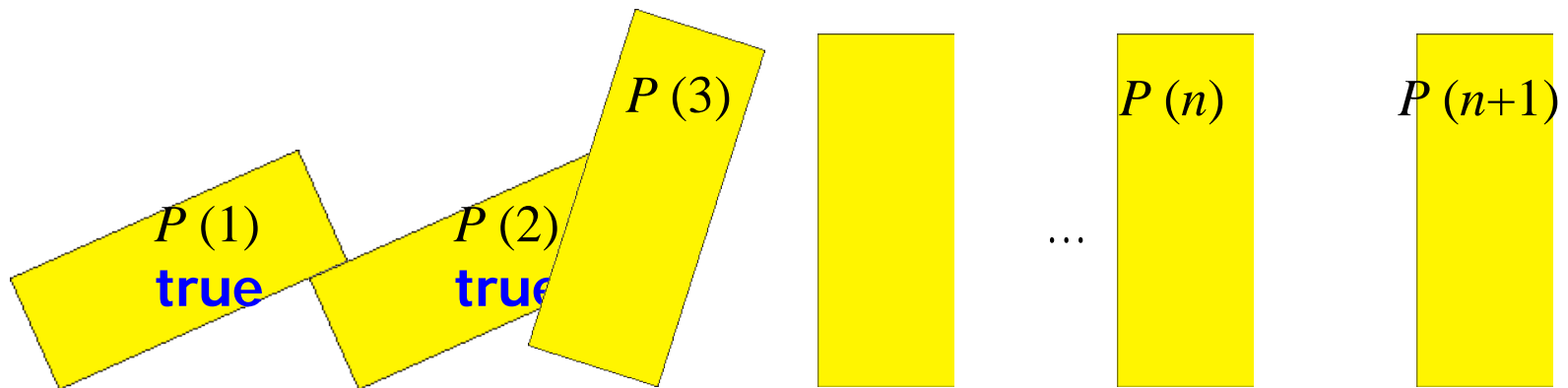
Mathematical Induction

Then we can conclude that all the dominos fall!



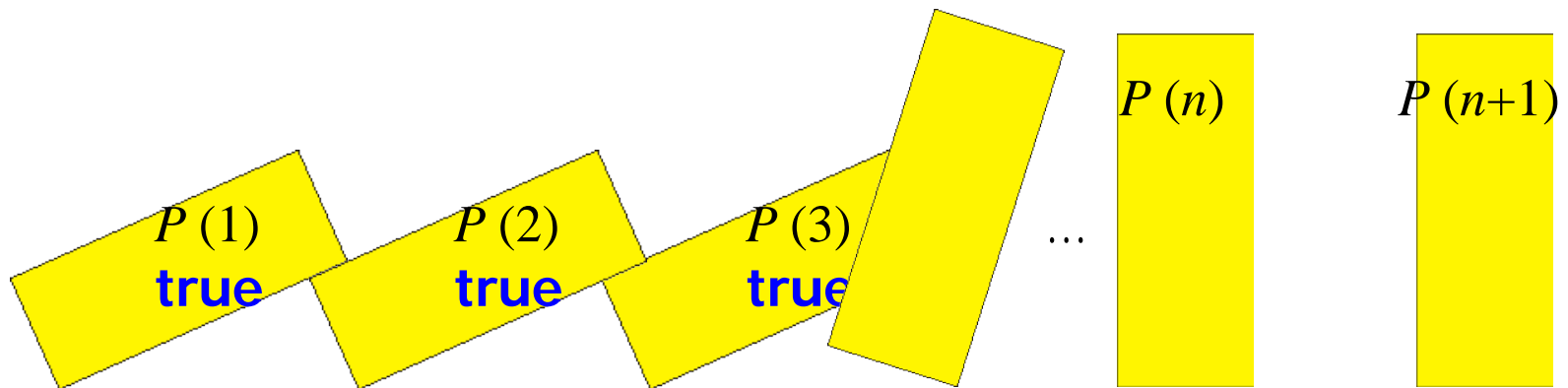
Mathematical Induction

Then we can conclude that all the dominos fall!



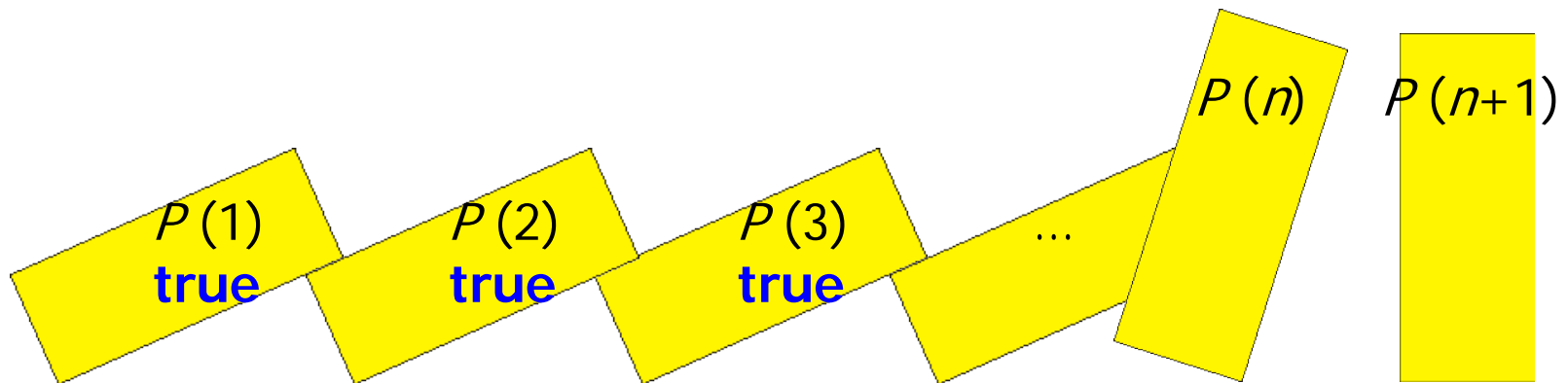
Mathematical Induction

Then we can conclude that all the dominos fall!



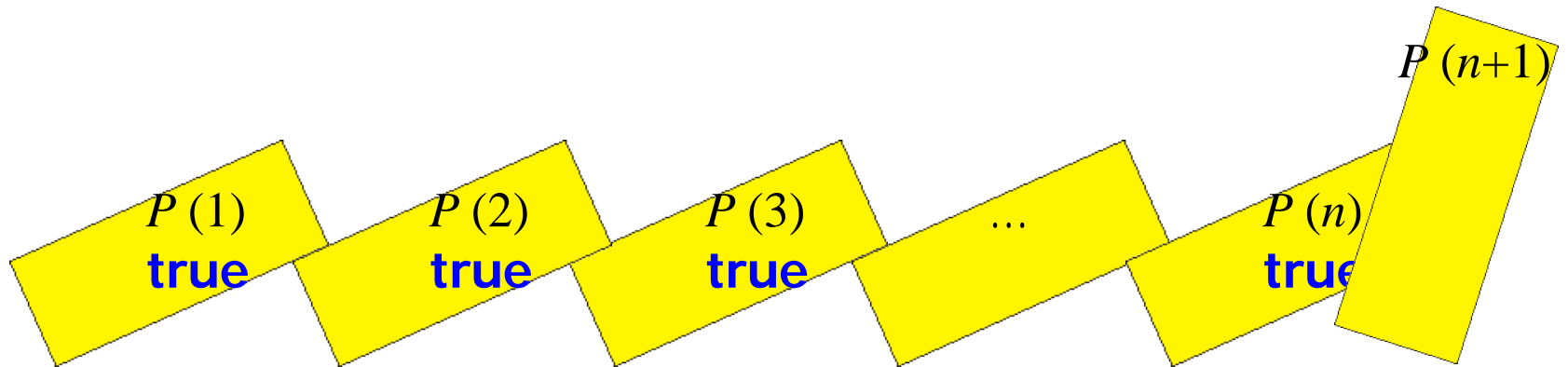
Mathematical Induction

Then we can conclude that all the dominos fall!



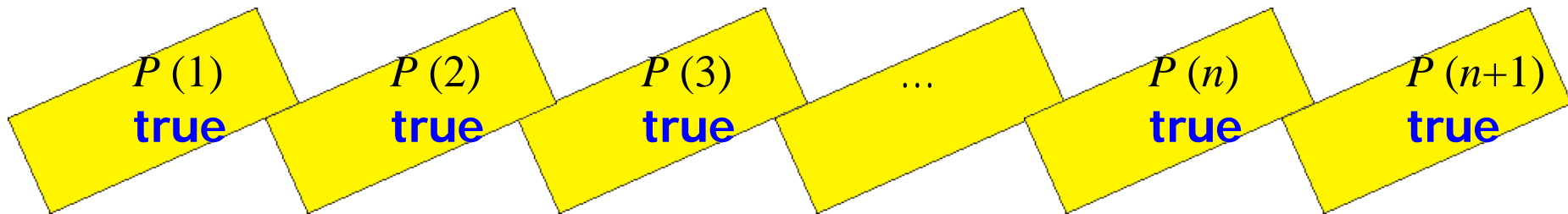
Mathematical Induction

Then we can conclude that all the dominos fall!



Mathematical Induction

Then we can conclude that all the dominos fall!

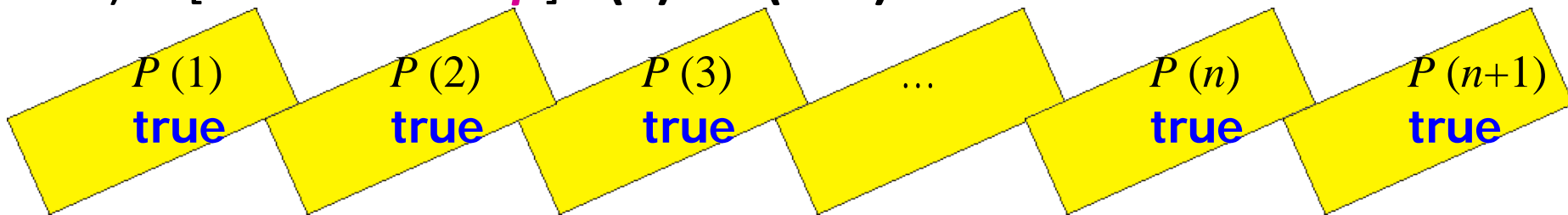


Mathematical Induction

Principle of Mathematical Induction:

If:

- 1) [*basis*] $P(1)$ is true
- 2) [*inductive hypothesis*] $P(n)$ is true
- 3) [*inductive step*] $P(n) \rightarrow P(n+1)$ is true



Then:

$$\forall n \geq 1 \ P(n) \text{ is true}$$

This formalizes what occurred to dominos.

Proof By Induction (Weak Induction): General Approach

- Claim: $P(n)$ is true for all $n \geq k$
- Basis:
 - Show a predicate is true when $n = k$
(usually k is taken equal to 1 or 0 or another small number, reasonable for a certain predicate)
- Inductive hypothesis:
 - Assume the predicate is true for an arbitrary n
- Inductive Step:
 - Show that the **implication $P(n) \rightarrow P(n+1)$** is true

Proof By Induction (Weak Induction): General Approach

- Three steps of weak induction:
 - Show that $P(k)$ is true for fixed constant k
 - Usually $k = 0$ or $k=1$
 - Suppose $P(n)$ is true
 - Show that $P(n) \rightarrow P(n+1)$ is true
- Then $P(j)$ is true for all $j \geq k$
 - This means that if $P(k)$ is true for the fixed constant k and it follows from $P(j)$ is true for $j=n$ that $P(j)$ is true for $j=n+1$, then $P(j)$ is true for all $j \geq k$

Induction Example 1: Arithmetic Progression

- Arithmetic Progression is a sequence of numbers such that the difference of any two successive members of the sequence is a constant:

$$a_1, a_1 + d, a_1 + 2d, \dots, a_1 + nd$$

- d is the difference of the progression

Induction Example 1: Arithmetic Progression

- Prove $a_n = a_1 + (n-1)d$

- Basis:

- If $n = 1$, then $a_1 = a_1 + (1-1)d = a_1 + 0d = a_1$

- Inductive hypothesis (assume true for n):

- Assume $a_n = a_1 + (n-1)d$

- Step (show that the implication $P(n) \rightarrow P(n+1)$ is true):

$$a_{n+1} = a_n + d = a_1 + (n-1)d + d =$$

$$= a_1 + nd - d + d = a_1 + nd = a_1 + ((n+1)-1)d$$

Induction Example 2: Arithmetic Progression

- The sum of the first n members of the arithmetic progression is given by

$$S = \frac{a_1 + a_n}{2} n$$

- Particularly, for $a_1 = 1$, $d=1$, $a_n = 1 + (n-1) = n$ and

$$S = \frac{n(1+n)}{2} = \frac{n(n+1)}{2}$$

Induction Example 2: Arithmetic Progression

- **Prove** $1 + 2 + 3 + \dots + n = n(n+1) / 2$
 - **Basis:**
 - If $n = 1$, then $1 = 1(1+1) / 2 = 2/2 = 1$
 - **Inductive hypothesis (assume true for n):**
 - Assume $1 + 2 + 3 + \dots + n = n(n+1) / 2$
 - **Step (show that the implication $P(n) \rightarrow P(n+1)$ is true):**
$$\begin{aligned} 1 + 2 + \dots + n + n + 1 &= (1 + 2 + \dots + n) + (n+1) = \\ &= n(n+1)/2 + n+1 = [n(n+1) + 2(n+1)]/2 \\ &= (n+1)(n+2)/2 = (n+1)((n+1) + 1) / 2 \end{aligned}$$

Strong Induction

- **Strong induction** means
 - **Basis**: show $P(0)$
 - **Hypothesis**: assume $P(k)$ holds for arbitrary $k \leq n$
 - **Step**: Show $P(n+1)$ follows from $P(n-i), \dots, P(n-1), P(n)$
- **Another variation of Strong induction** :
 - **Basis**: show $P(0), P(1)$
 - **Hypothesis**: assume $P(n)$ and $P(n+1)$ are true
 - **Step**: show $P(n+2)$ follows from $P(n), P(n+1)$

Verification of Correctness of Algorithms

- Mathematical induction is a main method, which is used to verify correctness of iterative algorithms, particularly of those, which are used to process an array
- Such a verification should usually be reduced to the following:
 - 1) prove that an algorithm is correct when an array contains 1 or 2 elements
 - 2) assume that an algorithm is correct when an array contains n elements
 - 3) prove that if an algorithm is correct when an array contains n elements, then it is also correct when an array contains $n+1$ elements

Introduction to Sorting Algorithms

- Sort: arrange values into an order:
 - Alphabetical
 - Ascending numeric
 - Descending numeric
- Two algorithms considered here:
 - Bubble sort
 - Selection sort

Bubble Sort

Concept:

- Compare 1st two elements
 - If out of order, exchange them to put in order
- Move down one element, compare 2nd and 3rd elements, exchange if necessary. Continue until end of array.
- Pass through array again, exchanging as necessary
- Repeat until pass made with no exchanges

Example – First Pass

Array `numlist3` contains:

17	23	5	11
----	----	---	----

compare values
17 and 23 – in correct
order, so no exchange

compare values 23 and
5 – not in correct order,
so exchange them

Example – First Pass

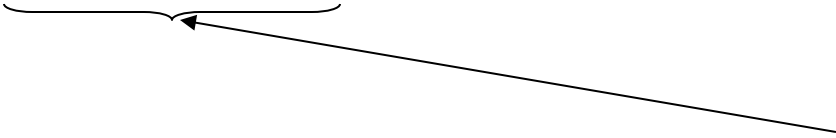
Array `numlist3` contains:

17	5	23	11
----	---	----	----

Example – First Pass

Array `numlist3` contains:

17	5	23	11
----	---	----	----

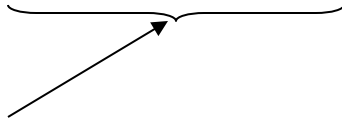


compare values 23 and 11 – not in correct order, so exchange them

Example – Second Pass

After first pass, array `numlist3` contains:

17	5	11	23
----	---	----	----

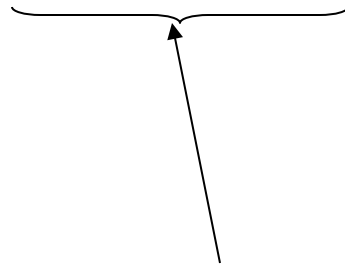


compare values 17 and
5 – not in correct order,
so exchange them

Example – Second Pass

After first pass, array `numlist3` contains:

5	17	11	23
---	----	----	----



compare values 17 and
11 – not in correct order,
so exchange them

Example – Second Pass

After first pass, array `numlist3` contains:

5	11	17	23
---	----	----	----



compare values 17 and 23 – in correct order, so no exchange

Example – Third Pass

After second pass, array `numlist3` contains:

5	11	17	23
---	----	----	----

compare values 5 and 11 – in correct order, so no exchange

compare values 11 and 17 – in correct order, so no exchange

compare values 17 and 23 – in correct order, so no exchange

No exchanges, so array is in order!

Bubble sort Algorithm (Ascending order)

```
for j = 2 to length(A)
  {key = A[j]
    i = j - 1;
    while (i > 0) and (A[i] > key)
      { A[i+1] = A[i]
        i = i - 1
        A[i+1] = key
      }
  }
```

Bubble sort Algorithm – Verification using Induction

- **Basis.** Suppose an array consist of 2 elements. Then $\text{key} = A[2]$, $i=1$. If $A[1] > \text{key}$ (that is $A[1] > A[2]$), then $A[2] = A[1]$ and $A[1] = \text{key}$. An array is sorted. If $A[1] < \text{key}$, an array is initially sorted. Proven.
- **Hypothesis.** Suppose this algorithm may sort an array containing n elements.

Bubble sort Algorithm – Verification using Induction

- **Inductive step.** We have an array containing $n+1$ elements. Suppose first n of them are already sorted (based on the inductive step). Let us consider the last iteration of the for loop, $j=n+1$. Then $i=n$. We may have two cases.
 - 1) $\text{key}=A[n+1]$ and $A[n]>\text{key}$. Then $A[n+1]=A[n]$ and $A[n]=\text{key}$, $i=n-2$ and we repeat the while loop. Since first n elements were already sorted, this repetition will unavoidably be resulted in placement of $A[n+1]$ in a proper place.
 - 2) $\text{key}=A[n+1]$ and $A[n]<\text{key}$. Since first n elements are already sorted, this means that an entire array is already sorted.

Selection Sort

- Concept for sort in ascending order:
 - Locate smallest element in array. Exchange it with element in position 1
 - Locate next smallest element in array. Exchange it with element in position 2.
 - Continue until all elements are arranged in order


Selection Sort - Example

Array `numlist` contains:

11	2	29	3
----	---	----	---

1. Smallest element is 2. Exchange 2 with element in 1st position in array:


2	11	29	3
---	----	----	---



Example (Continued)


2. Next smallest element is 3. Exchange 3 with element in 2nd position in array:

2	3	29	11
---	---	----	----



3. Next smallest element is 11. Exchange 11 with element in 3rd position in array:

2	3	11	29
---	---	----	----



Selection Sort Algorithm (Ascending order)

```
for j = 1 to length (A)-1
{
    for i=j+1 to length (A)
        if A[j]>A[i] // if this is true, swap A[i] and A[j]
            {
                key=A[j]
                A[j]=A[i]
                A[i]=key
            }
}
```