

## 1.1 Types of Number, Well-Ordering, Floors and Ceilings

## 1. Types of numbers and notation

## 2. Well-ordered sets

- Def: A set of numbers is **well-ordered** if every non-empty subset has a least element
- Axiom:  $\mathbb{Z}^+$  is well-ordered
- Related Proofs:
  - Prove  $\sqrt{2}$  is irrational
  - Prove how induction works

## 3. Floors and Ceilings

- $\lfloor x \rfloor = \text{floor}(x) = \text{largest integer} \leq x$
- $\lceil x \rceil = \text{ceil}(x) = \text{smallest integer} \geq x$   
Alternative definition used for proofs:
- for  $x \in \mathbb{R}$  and  $n \in \mathbb{Z}$ ,  $\lfloor x \rfloor = n$  iff  $n \leq x < n+1$
- for  $x \in \mathbb{R}$  and  $n \in \mathbb{Z}$ ,  $\lceil x \rceil = n$  iff  $n-1 < x \leq n$

## 4. Countable and Uncountable Sets

- Def: A set  $S$  is **countably infinite** if it can be placed in 1-1 correspondence with  $\mathbb{Z}^+$
- Def: A set  $S$  is **countable** if it is either **finite** or **countably infinite**
- Theorem: If  $S$  and  $T$  are both countable then so is  $S \oplus T$
- Uncountable set example:  $[0,1]$  Proof using contradiction

## 1.2 Sums and Products

## 1. Notation

(A) The sum:  $\sum_{j=m}^n a_j = a_m + a_{m+1} + \dots + a_n$

(B) The product:  $\prod_{j=m}^n a_j = a_m \cdot a_{m+1} \dots a_n$

## 2. Some sums:

(A)  $\sum_{j=1}^n 1 = n$

(B)  $\sum_{j=1}^n j = \frac{n(n+1)}{2}$

(C)  $\sum_{j=1}^n j^2 = \frac{n(n+1)(2n+1)}{6}$

(D)  $\sum_{j=0}^n r^j = \frac{r^{n+1}-1}{r-1}$

## 3. Some techniques:

- (A) Telescoping: Often partial fractions can help
- (B) Trimming: Change starting index from  $m$  to 1
- (C) Reindexing: Substitution

## 1.3 Induction

## 1. Weak Induction:

Base case: We show  $P(n_0)$  is true (plug it in and show!)

Inductive step:

I.H: Assume it works for  $k$

I.C: Show it works for  $k+1$

## 2. Proof of Induction (using well-ordered)

## 3. Strong Induction:

Inductive step:

I.H: Assume it works for every  $k$ , where  $k \leq n$

I.C: Show it works for  $n+1$

Base case(s):

Analyze the based on I.S, to find how many base cases we need

Write down base cases (Note: Don't use any extra base case)

## 1.5