LaTeX Typesetting for Discrete Mathematics SIT192

Nirosh Ravindran BSCP|CS|63|134

January 29, 2024

1. Prove that $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$ for $n \geq 0$, by induction.

Solution:

Proof.

Base case. When $n = 0$, LHS = 0 and RHS = $\frac{0 \cdot 1}{2} = 0$. Thus RHS = LHS.
Inductive step. We assumed that $\sum_{i=1}^{q} i = \frac{q(q+1)}{2}$ is true for an arbitrary fixed integer q and attempt to prove the validity of the formula for $(q+1)$. Thus

$$\begin{aligned}
\sum_{i=1}^{q+1} i &= \underbrace{\sum_{i=1}^{q} i}_{\frac{q(q+1)}{2}} +(q+1) \\
&= \frac{q(q+1)}{2} + (q+1) \\
&= \frac{q(q+1) + 2(q+1)}{2} \\
&= \frac{(q+1)(q+1)}{2}.
\end{aligned}$$

which is precisely the right-side of $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$ when $n = (q+1)$. Therefore, by the principle of mathematical induction, $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$. ∎

[3]

1 Integrals

Integral $\int_a^b x^2 dx$ inside text.

The same integral on display:

$$\int_a^b x^2 \, dx$$

and multiple integrals:

$$\iint_V \mu(u,v) \, du \, dv$$
$$\iiint_V \mu(u,v,w) \, du \, dv \, dw$$
$$\iiiint_V \mu(t,u,v,w) \, dt \, du \, dv \, dw$$
$$\int \cdots \int_V \mu(u_1,\dots,u_k) \, du_1 \dots du_k$$
$$\oint_V f(s) \, ds$$

[1]

2 Sums and products

Sum $\sum_{n=1}^{\infty} 2^{-n} = 1$ inside text.
The same sum on display:

$$\sum_{n=1}^{\infty} 2^{-n} = 1$$

Product $\prod_{i=a}^{b} f(i)$ inside text.
The same product on display:

$$\prod_{i=a}^{b} f(i)$$

[2]

Highlighted Code

```
import numpy as np

def incmatrix(genl1,genl2):
m = len(genl1)
n = len(genl2)
M = None #to become the incidence matrix
VT = np.zeros((n*m,1), int)  #dummy variable

#compute the bitwise xor matrix
M1 = bitxormatrix(genl1)
M2 = np.triu(bitxormatrix(genl2),1)

for i in range(m-1):
for j in range(i+1, m):
[r,c] = np.where(M2 == M1[i,j])
for k in range(len(r)):
VT[(i)*n + r[k]] = 1;
VT[(i)*n + c[k]] = 1;
VT[(j)*n + r[k]] = 1;
VT[(j)*n + c[k]] = 1;

if M is None:
M = np.copy(VT)
else:
M = np.concatenate((M, VT), 1)

VT = np.zeros((n*m,1), int)

return M
```

3 Algorithm Description

In this section, we describe a simple algorithm for finding the maximum element in an array.

Algorithm 1: Find Maximum Element in Array

Data: Array A of size n
Result: Maximum element in the array

for $i \leftarrow 1$ **to** n **do**
  if $A[i] > maxElement$ **then**
    maxElement $\leftarrow A[i]$;

return *maxElement*;

The algorithm iterates through each element in the array and updates the maximum element if a larger element is encountered. The final result is the maximum element in the array.

4 Analysis

The time complexity of this algorithm is $O(n)$, where n is the size of the array. This is because it iterates through the entire array once.

References

[1] Ivan K Lifanov. *Singular integral equations and discrete vortices*. Vsp, 1996.

[2] Seymour Lipschutz and Marc Lars Lipson. *Theory and problems of discrete mathematics*. McGraw-Hill, 2007.

[3] Gabriel J Stylianides, Andreas J Stylianides, and George N Philippou. Preservice teachers' knowledge of proof by mathematical induction. *Journal of Mathematics Teacher Education*, 10:145–166, 2007.