Induction Proof and Loop Invariants

Mathematical Induction

- First Principle
- Second Principle

The First Principle of Mathematical Induction

Induction hypothesis

$$\frac{\begin{array}{ll} P(a) & \\ \forall k \geq a & (P(k) \rightarrow P(k+1)) \end{array}}{\forall n \geq a \quad P(n)}$$

Induction basis

Induction step

Example

- Show that $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}, \quad n \geq 1.$

? What is P(n)? What is a?

- – Induction Basis
- – Induction Step
  - Induction hypothesis

The Second Principle of Mathematical Induction

Induction hypothesis

$$\begin{array}{l} \forall a \le k \le b, \quad P(k) \\ \forall k \ge b, \quad (\forall a \le j \le k, P(j) \rightarrow P(k+1)) \\ \hline \forall n \ge a, \quad P(n) \end{array}$$

Induction basis

Induction step

Example

- Define $f_n = \begin{cases} n, & n = 0,1 \\ f_{n-1} + f_{n-2}, & n \ge 2 \end{cases}$
- Show that $f_n = \frac{1}{\sqrt{5}}[\phi^n - (-\phi)^{-n}], \quad \phi = \frac{1+\sqrt{5}}{2}$

? What is P(n)? What are *a* and *b*?

- Induction Basis
- Induction Step
  - Induction hypothesis

Example

```
double fibRecursive(int n)
{
  double ret;

  if (n<2)
    ret = (double)n;
  else
    ret =  fibRecursive(n-1) +
           fibRecursive(n-2);

  return ret;
}
```

How many additions in terms of n?

Assume the total additions is $g(n)$.

$$\begin{cases} g(n) = 0, \; n = 0, 1 \\ g(n) = g(n-1) + g(n-2) + 1, \; n \ge 2 \end{cases}$$

f_n versus $g(n)$

n	0	1	2	3	4	5	6	7	8
f_n	0	1	1	2	3	5	8	13	21
g(n)	0	0	1	2	4	7	12	20	33
$2 f_n$	0	2	2	4	6	10	16	26	42

Prove by induction that

$$\forall n \ge 2, f_n \le g(n)$$

Loop Invariants

- To prove some statement S about a loop is correct. Define S in terms of a series of smaller statements, $S_0, S_1, \ldots, S_k$, where
  - The initial claim, S_0, is true before loop begins
    - Initialization (compared to induction basis)
  - If S_{i-1} is true before iteration i begins, then S_i will be true after iteration i is over
    - Maintenance (compared to induction step)
  - The final statement implies S
    - Termination (conclusion. This step is different from a typical induction proof)

Example: minArray

```
// return the value of the minimum
// element of array a
// n ≥ 1
int minArray(int a[n])
{ int m = a[0];

  for (i=1; i<n; i++)
    if (a[i] < m)
      m = a[i];

  return m;
}
```

Show that the algorithm returns min(a[0..n-1]).

What is S_i?

S_i: m = min(a[0..i])

Prove Fn = fn

```
double fibIterative(int n)
{
  double Fn_1, Fn_2, Fn;
  int i;

  if (n<2)
    return ((double)n);

  Fn_2 = 0;
  Fn_1 = 1;
  Fn = 1;
  for (i=2; i<=n ; i++) {
     Fn = Fn_1 + Fn_2;
     Fn_2 = Fn_1;
     Fn_1 = Fn;
  }
  return Fn;
}
```

S_i: Fn_1=f_i; Fn_2=f_{i-1}; Fn=f_i, i=1,2,….,n

- Initialization: Prove initial claim S_1
- Maintenance: Prove: $S_{i-1} \longrightarrow S_i$
- Termination: S = Sn

Insertion sort

```
void insertionSort(int A[0..n-1], int n)
{
  int i, j, tmp;

  for (i=1; i<n; i++) {
    tmp = A[i];
    j = i-1;
    while (j>=0 && tmp<A[j]) {
      A[j+1] = A[j];
      j--;
    }
    A[j+1] = key;
  }
}
```

S_i: A[0..i] is a permutation of the original sub-array and sorted