Preuves d'algorithmes

Informatique pour tous

Preuves de programme

Prouver un programme c'est faire deux choses:

1. Montrer qu'il **termine**, sans «planter» (pas de boucle infinie, pas de division par 0...)

Preuves de programme

Prouver un programme c'est faire deux choses:

1. Montrer qu'il **termine**, sans «planter» (pas de boucle infinie, pas de division par 0...)
2. Montrer qu'il donne le résultat attendu

Preuves de programme

Prouver un programme c'est faire deux choses:

1. Montrer qu'il **termine**, sans «planter» (pas de boucle infinie, pas de division par 0...)
2. Montrer qu'il donne le résultat attendu, souvent grâce à un **invariant de boucle**: une propriété vraie initialement, qui reste vraie à chaque itération de boucle et qui montre que l'algorithme renvoie bien le bon résultat.

Preuves de programme

Prouver un programme c'est faire deux choses:

1. Montrer qu'il **termine**, sans «planter» (pas de boucle infinie, pas de division par 0...)
2. Montrer qu'il donne le résultat attendu, souvent grâce à un **invariant de boucle**: une propriété vraie initialement, qui reste vraie à chaque itération de boucle et qui montre que l'algorithme renvoie bien le bon résultat.

Montrer un invariant de boucle $\approx$ preuve par récurrence.

Preuve de terminaison

Pour montrer qu'une boucle `while` s'arrête, on trouve en général une quantité en rapport avec les variables du programme:

1. qui est un entier positif
2. qui décroît strictement à chaque itération

Preuve de terminaison

Pour montrer qu'une boucle `while` s'arrête, on trouve en général une quantité en rapport avec les variables du programme:

1. qui est un entier positif
2. qui décroît strictement à chaque itération

Comme il n'existe pas de suite d'entiers positifs $(u_n)_{n\in\mathbb{N}}$ strictement décroissante, la boucle termine.

Preuve de terminaison

Pour montrer qu'une boucle `while` s'arrête, on trouve en général une quantité en rapport avec les variables du programme:

1. qui est un entier positif
2. qui décroît strictement à chaque itération

Comme il n'existe pas de suite d'entiers positifs $(u_n)_{n\in\mathbb{N}}$ strictement décroissante, la boucle termine.

(Remarque: une boucle `for` termine toujours!)

Preuve de correction

Pour montrer qu'une boucle (`for` ou `while`) produit le bon résultat, on trouve une propriété P, appelée **invariant de boucle**, telle que:

1. P est vraie avant la première itération de boucle
2. Si P est vraie à la $n$ème itération, P est vraie à l'itération $n + 1$
3. Une fois la boucle terminée, P prouve que le résultat est le bon

Exemple

```python
def f(a, b):
    q, r = 0, a
    while r >= b:
        r = r - b
        q = q + 1
    return [q, r]
```

Question

Soient `a` et `b` des entiers tels que $a \geq b \geq 0$.
Est-ce que `f(a, b)` termine?

Exemple

```python
def f(a, b):
    q, r = 0, a
    while r >= b:
        r = r - b
        q = q + 1
    return [q, r]
```

Question

Deviner à quoi sert la fonction `f` et le prouver.

Division euclidienne

```python
def f(a, b):
    q, r = 0, a
    while r >= b:
        r = r - b
        q = q + 1
    return [q, r]
```

Montrons par récurrence l'**invariant de boucle** suivant:

$$P\text{: « } a = bq + r \text{ »}$$

Division euclidienne

```
def f(a, b):
    q, r = 0, a
    while r >= b:
        r = r - b
        q = q + 1
    return [q, r]
```

Montrons par récurrence l'**invariant de boucle** suivant:

$$P\text{: « } a = bq + r \text{ »}$$

1. P est vraie initialement car $r = a$ et $q = 0$.

Division euclidienne

```python
def f(a, b):
    q, r = 0, a
    while r >= b:
        r = r - b
        q = q + 1
    return [q, r]
```

Montrons par récurrence l'**invariant de boucle** suivant:

$$P: \text{« } a = bq + r \text{ »}$$

1. P est vraie initialement car $r = a$ et $q = 0$.
2. Lors d'une itération, on augmente q de 1 et on diminue r de b. Donc $bq + r$ reste le même, ce qui montre que P reste vraie.

Exemple

```python
def f(a, b):
    q, r = 0, a
    while r >= b:
        r = r - b
        q = q + 1
    return [q, r]
```

Question

Deviner à quoi sert la fonction `f` et le prouver.

Exemple

```python
def f(a, b):
    q, r = 0, a
    while r >= b:
        r = r - b
        q = q + 1
    return [q, r]
```

Question

Deviner à quoi sert la fonction `f` et le prouver.

Les entiers `q` et `r` sont donc tels que `a = bq + r` et `0 <= r < b`.

Exemple

```python
def f(a, b):
    q, r = 0, a
    while r >= b:
        r = r - b
        q = q + 1
    return [q, r]
```

Question

Deviner à quoi sert la fonction `f` et le prouver.

Les entiers `q` et `r` sont donc tels que `a = bq + r` et `0 <= r < b`.
`f(a, b)` renvoie donc le quotient et le reste de la division euclidienne de `a` par `b`.

Exemple

```python
def f(L):
    res = L[0]
    for i in range(1, len(L)):
        if L[i] > res:
            res = L[i]
    return res
```

Exemple

```python
def f(L):
    res = L[0]
    for i in range(1, len(L)):
        if L[i] > res:
            res = L[i]
    return res
```

Question

Est-ce que `f(L)` termine?

Exemple

```python
def f(L):
    res = L[0]
    for i in range(1, len(L)):
        if L[i] > res:
            res = L[i]
    return res
```

Question

Est-ce que `f(L)` termine?

Une boucle `for` termine toujours!

Exemple

```python
def f(L):
    res = L[0]
    for i in range(1, len(L)):
        if L[i] > res:
            res = L[i]
    return res
```

Question

Deviner à quoi sert la fonction `f` et le prouver.

Exemple

```python
def f(L):
    res = L[0]
    for i in range(1, len(L)):
        if L[i] > res:
            res = L[i]
    return res
```

Question

Deviner à quoi sert la fonction `f` et le prouver.

On montre par récurrence:

P_i: « au début de la $i$ème itération de la boucle `for`, `res` contient le maximum de `L[0]`, ..., `L[i - 1]` »

Exemple

```python
def f(L):
    res = L[0]
    for i in range(1, len(L)):
        if L[i] > res:
            res = L[i]
    return res
```

Question

Deviner à quoi sert la fonction `f` et le prouver.

On montre par récurrence:

P_i: « au début de la $i$ème itération de la boucle `for`, `res` contient le maximum de `L[0]`, ..., `L[i - 1]` »

L'initialisation et l'hérédité sont facilement vérifiées.

Exemple

```python
def f(a, b):
    x, y = a, b
    while y != 0:
        x, y = y, x % y
    return x
```

Question

1. Est-ce que `f` termine? Pour quelles valeurs de `a` et `b`?
2. Deviner à quoi sert `f` et le prouver.

Exemple

```
def f(a, b):
    while a > 0 and b > 0:
        if a % 2 == 0:
            a = a + 1
            b = b - 3
        else:
            a = a - 2
            b = b + 1
    return 1
```

Question

Est-ce que `f` termine? Pour quelles valeurs de `a` et `b`?

Exemple

```python
def f(L, e):
    i, j = 0, len(L)-1
    while i < j:
        m = (i + j) // 2
        if L[m] < e:
            i = m + 1
        else:
            j = m
    return L[i] == e
```

Question

1. Est-ce que `f` termine?
2. Deviner à quoi sert `f` et le prouver.

Exemple

```python
def f(x, n):
    res = x
    m = n
    while m > 1:
        if m % 2 == 0:
            res = res * res
        else:
            res = x * res * res
        m = m // 2
    return res
```

Question

1. Est-ce que `f` termine?
2. Deviner à quoi sert `f` et le prouver.