Correction du TD N°2

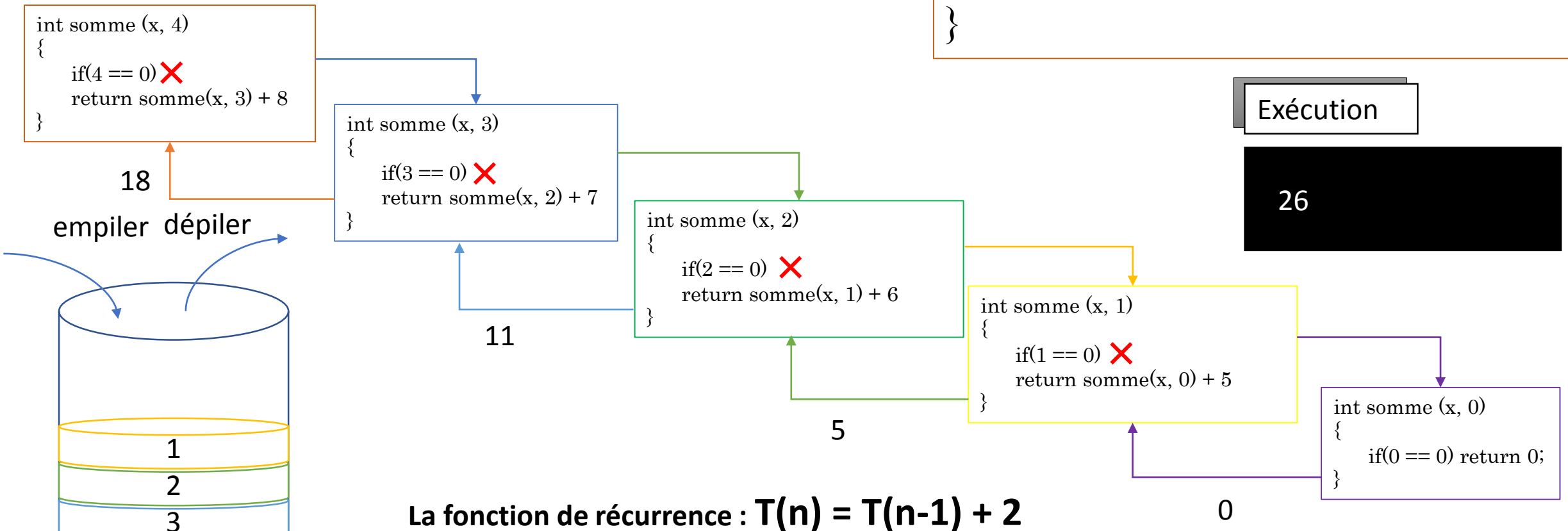
Algorithmes récursifs

TD2 Exercice 5:

5	6	7	8

X, n= 4

```
int somme (int x[ ], int n)
{
    if(n== 0) return 0;
     return (somme(x, n-1) + x[n-1]);
}
```



La fonction de récurrence : $T(n) = T(n-1) + 2$

TD2 Exercice 6.1:

- 5, 7, 13, 4, 7, 0, 5, 15, 50, 0

```
void traiter()
{ int x;
cin >> x;
if (x != 0)
    traiter();
cout << " , " << x;
}
void main () { traiter (); }
```

Exécution

```
0 7 4 13 7 5
```

```
Void traiter( )
    {
    cin >> 5
    if (5!=0)
        traiter( );
     cout << « , » << 5;
    }
```

```
Void traiter ( )
    {
    cin >> 7
    if (7!=0)
        traiter( );
     cout << « , » << 7;
    }
```

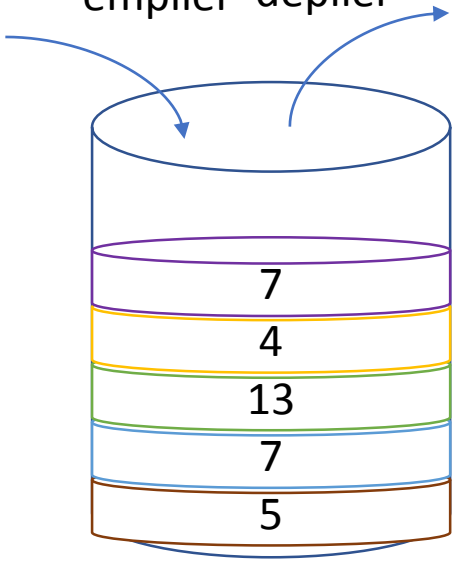
```
Void traiter ( )
    {
    cin >> 13
    if (5!=0)
        traiter( );
     cout << « , » << 13;
    }
```

```
Void traiter ( )
    {
    cin >> 4
    if (4!=0)
        traiter( );
     cout << « , » << 4;
    }
```

```
Void traiter ( )
    {
    cin >> 7
    if (7!=0)
        traiter( );
     cout << « , » << 7;
    }
```

```
Void traiter ( )
    {
    cin >> 0
    if (0!=0)
     cout << « , » << 0;
    }
```

empiler dépiler



Pile d'exécution

TD2 Exercice 6.2:

- La version désécurisée de la fonction **traiter** ?

```
void traiter()
{ int x;
cin >> x;
if (x != 0)
    traiter();
cout << " , " << x;
}
void main () { traiter (); }
```

```
void traiter()
{ Pile.init ( );
  cin >> x;
  while (x != 0){
     Pile.push(x);
     cin >> x;
   }
   cout << x;
   while(! Pile.empty){
        y = Pile.pop( );
        cout << " , " << y;
    }
}
void main () { traiter (); }
```

TD2 Exercice 7:

- n = 13

```
void Binaire (int n)
{
if (n != 0){
    Binaire (n/2);
     cout << n%2;
     }
}
```

Exécution

1 1 0 1

```
Void Binaire (13)
   {
   if (13 != 0)
       { Binaire(6);
       cout << 1; }
   }
```

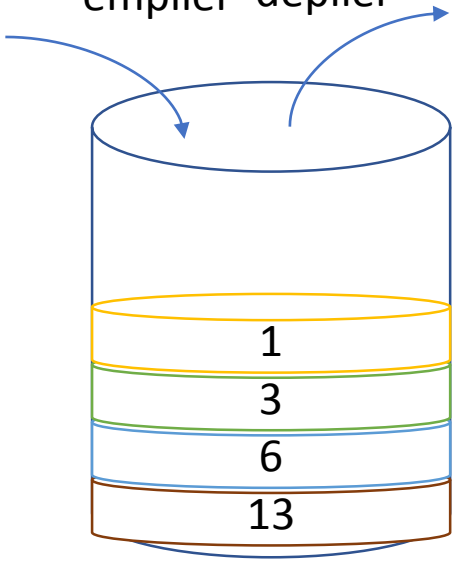
```
Void Binaire (6)
   {
   if (6 != 0)
       { Binaire(3);
         cout << 0; }
   }
```

```
Void Binaire (3)
   {
   if (3 != 0)
       { Binaire(1);
         cout << 1; }
   }
```

```
Void Binaire (1)
   {
   if (1 != 0)
       { Binaire(0);
        cout << 1; }
   }
```

```
Void Binaire (0)
   {
   if (0 != 0) ✗
    { …..}
}
```

empiler dépiler



Pile d'exécution

La fonction de récurrence : $T(n) = T(n/2) + 2$

TD2 Exercice 8:

- Palindrome = RADAR, PAAP

```
Int Palindrome ( char* m, int g, int d)
{
if (m[g] != m[d])
      return 0;
Else{
      if(g >= d)
            return 1;
       return Palindrome(m, g+1, d-1);
}
}
```

A chaque appel récursif, on diminue la longueur de la chaine m de 2

La fonction de récurrence : T(n) = T(n-2) + 2