BROUILLON - CANDIDAT - COMPTER EN PLACE LE NOMBRE DE BITS NON NULS D'UNE ÉCRITURE BINAIRE

CHRISTOPHE BAL

Document, avec son source $\LaTeX$, disponible sur la page
https://github.com/bc-writing/drafts.

Mentions « légales »

En informatique, il existe une méthode astucieuse pour compter le nombre de `1` dans une écriture binaire. Expliquons comment faire avec l'écriture binaire `1101.1001.0110.1111` qui contient $16 = 2^4$ chiffres. Ci-dessous, toutes les opérations se font en base 2. **Notez bien que l'on note les résultats dans une écriture binaire de taille 16 et c'est là la belle astuce du point de vue informatique.**

- On additionne les chiffres voisins puis on conserve le résultat sur deux chiffres.

```
[1+1 | 0+1 | 1+0 | 0+1 | 0+1 | 1+0 | 1+1 | 1+1]
[10  | 01  | 01  | 01  | 01  | 01  | 10  | 10 ]
```

- On additionne les paires voisines puis on conserve le résultat sur quatre chiffres.

```
[10  +  01 | 01  +  01 | 01  +  01 | 10  +  10]
[  0011    |   0010    |   0010    |   0100   ]
```

- On additionne les quadruplets voisins puis on conserve le résultat sur huit chiffres.

```
[  0011    +   0010    |   0010    +   0100   ]
[      00000101        |       00000110       ]
```

- On additionne les octuplets voisins puis on conserve le résultat sur seize chiffres et on s'arrête pour analyser le résultat.

```
[      00000101        +       00000110       ]
[             0000000000001011                ]
```

- Le dernier nombre a pour valeur décimale 11 qui est exactement le nombre de `1` contenus dans `1101.1001.0110.1111`.

Très joli ! Non ? Très bien ! Mais pourquoi, ou comment cela marche-t-il ? En fait c'est tout simple. Pour le voir reprenons les étapes en expliquant l'information qu'elles construisent.

- La 1re étape redonnée ci-dessous compte le nombre de `1` par paire de voisins. Il est important de noter qu'il n'y a pas de perte d'information car au maximum il y aura deux `1`, or deux s'écrit `10` en base 2.

Date: 31 Mars 2019.

```
[1+1 | 0+1 | 1+0 | 0+1 | 0+1 | 1+0 | 1+1 | 1+1]
[10  | 01  | 01  | 01  | 01  | 01  | 10  | 10 ]
```

- La 2^e^ étape redonnée ci-dessous compte le nombre de `1` par quadruplets de voisins sans perdre d'information car au maximum il y aura quatre `1`, or 4 s'écrit `0100` en base 2.

```
[10  +  01 | 01  +  01 | 01  +  01 | 10  +  10]
[  0011    |   0010    |   0010    |   0100   ]
```

- La 3^e^ étape redonnée ci-dessous compte le nombre de `1` par octuplets de voisins sans perdre d'information.

```
[  0011    +   0010    |   0010    +   0100   ]
[       00000101       |       00000110       ]
```

- La 4^e^ étape redonnée ci-dessous compte le nombre de `1` pour 16 voisins, c'est à dire tous les chiffres, sans perdre d'information. Simple mais efficace !

```
[       00000101       +       00000110       ]
[              0000000000001011               ]
```

Vous noterez le caractère général des explications précédentes. Ainsi il est immédiat d'étendre la méthode à toute écriture binaire contenant 2^p chiffres où $p \in \mathbb{N}^*$.