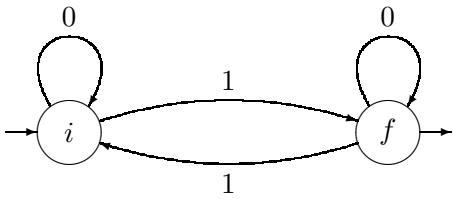
## Exercice 1

Soit l'alphabet $B = \{0, 1\}$.

1) a) Soit l'ensemble $L$ des mots sur $B$ contenant un nombre impair d'occurrences de 1 et l'automate $\mathcal{A}$ défini par :




L'état $i$ est initial.
L'état $f$ est final.

Montrer que $\mathcal{A}$ reconnaît le langage $L$.

b) Montrer que le langage $M$ constitué des mots sur $\{0, 1\}$ contenant un nombre pair de 0 et un nombre impair de 1 est reconnaissable.

2) On souhaite maintenant concevoir un circuit logique qui renvoie le nombre modulo 2 de 0 présents dans le mot passé en entrée au circuit.

On pourra utiliser les portes logiques élémentaires *et*, *ou*, *non* et *xor* (ou exclusif), que l'on représente simplement par :


Porte Non — Porte Et — Porte Ou — Porte Xor

a) Commençons par un mot de longueur 1 : $u = a$, avec $a \in \{0, 1\}$.

Construire un circuit $\mathcal{P}_0$ à une seule entrée $a$ et une sortie $s$ qui vaut 0 si $u$ contient un nombre pair de 0 et 1 sinon.

b) Supposons conçu un circuit $\mathcal{P}_n$ qui prend en entrée un mot $u$ de longueur $2^n$ et possède une sortie $s$ qui vaut 0 si $u$ contient un nombre pair de 0 et 1 sinon.

Construire, à l'aide de $\mathcal{P}_n$, un circuit $\mathcal{P}_{n+1}$ qui résout le même problème pour un mot de longueur $2^{n+1}$ en entrée.

c) Déterminer, lorsque $n$ tend vers $+\infty$, un équivalent du nombre de portes logiques élémentaires utilisées dans $\mathcal{P}_n$.

3) Soit $N$ l'ensemble des mots $u$ sur $\{0, 1\}$ tels que :

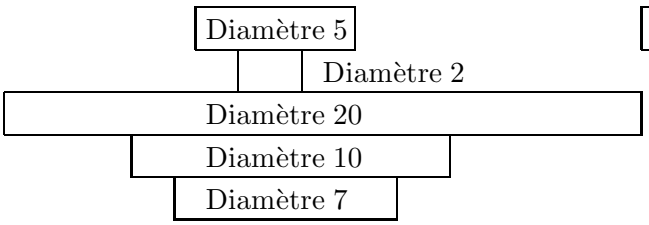
$$|u|_0 = 2\,|u|_1$$

où $|u|_0$ et $|u|_1$ désignent respectivement le nombre d'occurrences de 0 et de 1 dans $u$.

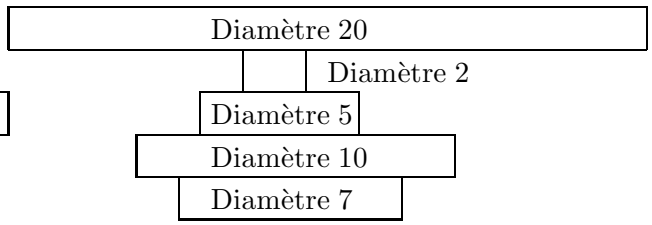
Ce langage est-il reconnaissable ?

## Exercice 2

On dispose d'une pile de rondelles de diamètres variés, que l'on souhaite ranger par ordre décroissant de taille, la plus large devant être en bas de la pile. Malheureusement, on ne peut manipuler la pile que « par morceaux » : la seule opération autorisée consiste à prendre la partie de la pile surmontant une certaine rondelle et à retourner d'un bloc tout le paquet.
Par exemple, si une pile contient des rondelles de diamètres 7, 10, 20, 2 et 5 et que l'on retourne le bloc commençant à la rondelle de diamètre 20, on obtient la pile de diamètres successifs 7, 10, 5, 2 et 20.



Pile au départ



Pile après retournement

Nous modéliserons la pile de rondelles par un tableau (ou vecteur) $d$ d'entiers, indicé de 0 à $n-1$, où $n$ est le nombre de rondelles, la case numéro $i$ contenant le diamètre en millimètres de la rondelle $i$. La case numéro 0 correspond au bas de la pile.
En Caml comme en Pascal, nous supposerons la constante $n$ définie.
En Caml, $d$ est de type `int vect`, et en Pascal de type `Pile` défini par :

```
Pile = ARRAY[0..n-1] OF INTEGER
```

1) Écrire :
   - en Caml une fonction `retourne : int vect -> int -> unit` telle que `retourne d i` modifie le vecteur `d` pour « retourner » la partie supérieure de la pile à partir de la rondelle d'indice $i$ (incluse) ;
   - en Pascal une procédure `retourne(VAR d : Pile ; i : INTEGER)` telle que `retourne(d,i)` modifie le tableau `d` pour « retourner » la partie supérieure de la pile à partir de la rondelle d'indice $i$ (incluse).
2) Quel est, en nombre d'affectations dans `d`, le coût de ce retournement ?
3) Comment faire pour mettre en bas de pile la rondelle la plus large ?
4) Écrire
   - en Caml une fonction `place : int vect -> int -> unit` telle que `place d i` modifie le vecteur `d` pour mettre à l'indice $i$ la rondelle la plus large de la partie de la pile surmontant la rondelle d'indice $i$ (incluse) ;
   - en Pascal une procédure `place(VAR d : Pile ; i : INTEGER)` telle que `place(d,i)` modifie le tableau `d` pour mettre à l'indice $i$ la rondelle la plus large de la partie de la pile surmontant la rondelle d'indice $i$ (incluse) ;
5) En déduire, selon le langage choisi, une fonction ou une procédure qui trie `d`.
6) Quel est le coût au pire de cette méthode, en nombre de retournements d'un morceau de pile ?