Exercice 1

(1) Insérer un élément au début du tableau demande de décaler tous les éléments du tableau, opération en $O(n)$ si tableau à n éléments (complexité quadratique cf cours)

(2) début des piles

Enfiler → | E | S | → Défiler

Cas S vide : | E | E |

$E : \emptyset$ $S : \emptyset$

First in first out

Enfiler(3) : E : | 3 | S : $\emptyset$

Enfiler(5) : E : | 3 | 5 | S : $\emptyset$

Enfiler(6) : E : | 3 | 5 | 6 | S : $\emptyset$

Défiler() : $E : \emptyset$ S : | 6 | 5 | 3 | → | 3 | 5 | 6 | ↓ | 5 | 6 |

On copie le tableau puis on inverse

Enfiler(12) : E : | 12 | S : | 5 | 6 |

Algo : ENFILER (E, S, x) : on a accès aux fonctions de pile

return EMPILER(x, E);

Algo : DEFILER(E, S) :

si $S = \emptyset$:
  tant que $E \neq \emptyset$:
    x = DEPILER(E)
    EMPILER(x, S)
return DEPILER(S)

Bonus, faire : ELEMENT(i, E, S) qui renvoie le i-ème élement de la file

Algo : ELEMENT(i, E, S) :
  len ← taille(S)
  si len > i :
    renvoyer S[len − i − 1]
  sinon :
    renvoyer E[i − len]

③ Complexité dans le pire des cas (en supposant que EMPILER / DEPILER / ELEMENT sont en temps constant sur E et S)

ENFILER ⇝ $O(1)$
DÉFILER ⇝ $O(1)$

DEFILER : dans le pire des cas, si $S = \emptyset$, on recopie E dans S, complexité $O(n)$ où n est le nombre d'éléments dans la file.

3.1 On effectue n opérations : on suppose qu'on fait k ENFILER et $n-k$ DEFILER avec $k \geq n-k$

$n-k$ éléments (ceux qui sont défilés), sont empilés dans E, dépilés de E, empilés dans S, dépilés de S
au total 4 opérations

Ceux qui ne sont pas défilés : il y en a
$$k-(n-k)=2k-n$$
Pour eux, on les a empilés dans E et peut-être dépilé de E, empilé dans S, 3 opérations au pire

En tout : $4(n-k)+3(2k-n)$ opérations
$= n+2k \leqslant 3n$ opérations

3.2 Méthode comptable (technique compte reste positif)

- Quand on ENFILE, on verse $+3$ au compte, et -1 pour EMPILER car ENFILE appelle EMPILER.
  le compte aura au final $+2$ et la pile E aura gagné un élément.

- Quand on DEFILE un élément, on ajoute $+1$ sur le compte.
  Si $S \neq \emptyset$, on retire -1 pour l'appel à DÉPILER et le compte reste inchangé.
  Si $S=\emptyset$, on doit payer -2 pour chaque élément de E (DEPILER(E) puis EMPILER(x, E)) ce qui vide le compte, puis on ajoute $+1$ pour DEFILER qu'on retire -1 pour le coût de DEPILER (inclu dans DEFILER)
  A la fin le compte vaut $0 = 2e$ (E est vide)

3.3 Méthode du potentiel

On associe l'état obtenu à la i ème étape le potentiel Φ

potentiel Φ_i coût amorti

Le coût amorti est : $a_i = c_i + \phi_i - \phi_{i-1}$

nbre d'op. élém. effectives

On prend ici $\Phi_i = 2 \cdot |E| = 2 \cdot e$ taille de E étape $i-1$

<u>Pour ENFILER</u> : $c_i = 1$, $\phi_i = 2 \cdot e$ $\phi_{i-1} = 2(e-1)$

↳ On a $a_i = 1 + 2e - (2(e-1)) = 3$ ok ($a_i \geqslant 3$)

<u>Pour DEFILER</u>

E reste inchangé

- Si $S \neq \emptyset$: $c_i = 1$, $\phi_i = 2e$ $\phi_{i-1} = 2e$

↳ et $a_i = 1 + 2e - 2e = 1 \leq 3$ ok

- Si $S = \emptyset$: $c_i = 2e' + 1$, $\phi_i = 0$ $\phi_{i-1} = 2e'$

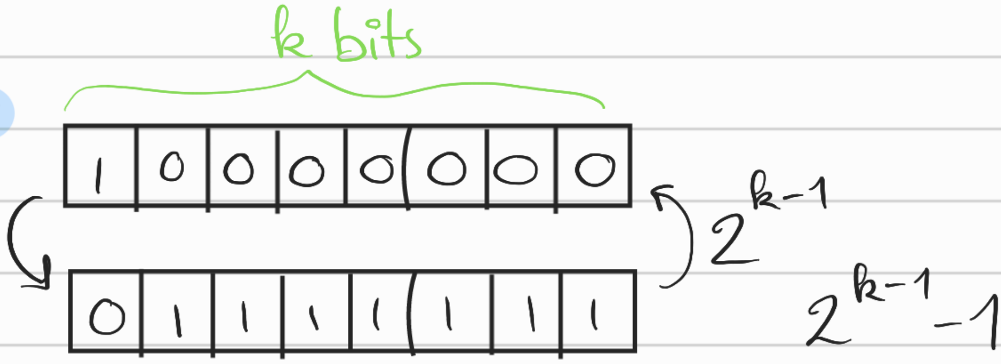
on dépile puis rempile chaque élément de E vers S, puis dépile dernier élément

E dépilé et vide

↳ et $a_i = 2e' + 1 + 0 - 2e' = 1 \leq 3$ ok

Exercice 2 :

①



A chaque INCREMENT / DECREMENT, on fait le changement de bits ⇝ coût amorti non constant

Exemple : $P = 18 = 1\,|\,0\,|\,0\,|\,1\,|\,0_2$ $\quad 18 \wedge 9 = 0$ car aucun 1 et 1

$N = 9 = 0\,|\,1\,|\,0\,|\,0\,|\,1_2$

$v = P - N = 9$

$P = 20 = 1\,|\,0\,|\,1\,|\,0\,|\,0_2$ $\quad 20 \wedge 13 \neq 0$

$N = 13 = 0\,|\,1\,|\,1\,|\,0\,|\,1_2$ $\quad v = P - N = 7$

$20 \wedge 13 = 0\,0\,1\,0\,0_2 = 4$

② $P = 8 = 0\,1\,0\,0\,0$ $\quad v = P - N = 7$ et $P \wedge N = 0$

$N = 1 = 0\,0\,0\,0\,1$

$P = 16 = 1\,0\,0\,0\,0$ $\quad v = P - N = 8$ et $P \wedge N = 0$

$N = 8 = 0\,1\,0\,0\,0$

③ 3.1 $P' = P - 2^k$ et $N' = N - 2^k$ (on passe le k ième bit à 0 dans P et N)

$P' - N' = (P - 2^k) - (N - 2^k) = P - N = v$

3.2 On peut partir de P quelconque $> v$
prendre $N = P - v$

Et tant que $N \wedge P \neq 0$, on effectue l'opération précédente (on passe un bit à 1, à la même position dans P et N, à 0)

3.3 $7 = 9 - 2 = 8 - 1$ chaque nombre a des représentations exclusives infinies

↓

Pour tout v, 2^p et $2^p - v$ avec $p > \lceil \log v \rceil$ sont des représentations exclusives de v

4) 4.1 On augmente P, puis on applique la question précédente pour obtenir une représentation exclusive de v.

4.2 Quand on incrémente P de 1, un seul bit de P passe de 0 à 1. Au pire, à cette position N avait un bit à 1, et on aura un seul bit à changer dans N et P pour conserver une représentation exclusive.

4.3

```
INCRÉMENT (P,N):
    i ← 0
    tant que P[i] = 1 :
        P[i] ← 0
        i ← i+1
    Si N[i] = 1 :          ← quand premier zéro dans P, on fait...
        N[i] ← 0
    Sinon :                  On ne fait pas P[i]+1 directement pour garder N ∧ P = 1
        P[i] ← 1
```

```
DÉCRÉMENT (P,N):
    i ← 0
    tant que P[i] = 1 :
        P[i] ← 0
        i ← i+1
    Si P[i] = 1 :          ← on inverse P et N de
```

$\llcorner$ $P[i] \leftarrow 0$ l'algo INCRÉMENT

Sinon :

$\llcorner$ $N[i] \leftarrow 1$

(5) Dans le pire des cas, on modifie $\lceil \log v \rceil$ bits

maximum nbr bits

(6)

6.1 $a_i = c_i + \phi_i - \phi_{i-1}$

6.2 Supposons que l'on fasse un INCREMENT sur (P, N) où P se termine par ℓ uns consécutifs ($k = \ell$ à la fin du tant que)

$c_i = n+1$ $\qquad \phi_i =$ $\qquad \phi_{i-1} =$

n+1 inversion de bits

nbr bits à 1 dans P a diminué de ℓ dans la boucle "tant que"

Si $N[\ell] = 1$, on passe un bit de N à 0, et $\phi_i = \phi_{i-1} - (\ell+1)$

Si $N[\ell] = 0$, alors on passe un bit de P à 1, et $\phi_i = \phi_{i-1} - \ell + 1$

Dans le premier cas, $a_i = (\ell+1) - (\ell+1) = 0$

Dans le second cas, $a_i = (\ell+1) - \ell + 1 = 2$

6.3 On a : $\sum^{t} c_i = \sum^{t} (a_i - \phi_{i-1} + \phi_i)$

$$\leq \sum_{i=1}^{t} a_i - \Phi_0 + \Phi_t$$

$\Phi_0 = 0$ car $(P,N) = (0,0)$ au début

$$\leq 2t - 0 - \Phi_t$$

Φ_t = nbr bits à 1 dans P + nbr bits à 1 dans N