# Koszt zamortyzowany

Czas potrzebny na wykonanie ciągu operacji dzielony jest równo między operacje.

*Koszt zamortyzowany* – średni koszt w najgorszym przypadku. (nie to samo co oczekiwany koszt).

Metody:

**Metoda kosztu sumarycznego:** Szacujemy koszt $T(n)$ ciągu $n$ operacji. Koszt zamortyzowany $= T(n)/n$.

**Metoda księgowania (accountig):** początkowe operacje umieszczają "nadpłatę" w elementach struktury danych, której używają późniejsze operacje na tych elementach.

**Metoda potencjału:** "Nadpłata" ("Potencjał") – związana z całą strukturą danych (nie z elementami).

# Operacje na stosie

Wcześniej omówione:

- `Push(`$S$`, `$x$`)` – czas: $O(1)$.
- `Pop(`$S$`)` – czas: $O(1)$.

```
Multipop(S,k)
1 while not Stack-Empty(S) and k ≠ 0 do
2    Pop(S)
3    k ← k − 1
```

Czas: $O(\min(s,k))$ gdzie $s$ – liczba elementów na stosie (przed wywołaniem).
Przeanalizujemy ciąg $n$ operacji: `Push`, `Pop` i `Multipop`.
Może się zdarzyć, że niektóre operacje (`Multipop`) wymagają czasu $O(n)$.

# Metoda kosztu sumarycznego

Łączny koszt $n$ opreacji można oszacować przez $O(n)$:
Każdy obiekt po włożeniu na stos, może być zdjęty co najwyżej raz. Stąd liczba wywołań `Pop` włączając wywołania wewnątrz `Multipop` jest $\leq n$. Koszt zamortyzowany jednej operacji: $O(n)/n = O(1)$.

# licznik binarny

Licznik – tablica bitów: $A[0 \ldots k-1]$.

$A[0]$ – najmniej znaczący bit.

Zwiększanie licznika o $1 \pmod{2^k}$:

```
Increment(A)
i ← 0
while i < length[A] and A[i] = 1 do
    A[i] ← 0
    i ← i + 1
if i < length[A] then
    A[i] ← 1
```

Czas (najgorszy przypadek): $\Theta(k)$.

Koszt jest równy liczbie zmienianych bitów. (każda iteracja `while` – zmiana innego bitu.)

# Metoda kosztu sumarycznego

Oszacowanie na czas wykonania $n$ operacji `Increment`:
Spostrzeżenie: bit $A[i]$ ulega zmianie raz na $2^i$ operacji.
($A[0]$ zmienia się w każdym kroku, $A[1]$ – co drugi krok, . . . )
Bity o numerach $> \lfloor \lg n \rfloor$ nie zmieniają się, bo
$\lfloor n/2^{\lfloor \lg n \rfloor + 1} \rfloor = 0$.
Sumaryczny koszt:

$$\sum_{i=0}^{\lfloor \lg n \rfloor} \lfloor n/2^i \rfloor < n \sum_{i=0}^{\infty} 1/2^i = 2n$$

Stąd zamortyzowany koszt pojedynczej operacji: $O(1)$.

# Metoda księgowania (stos)

Faktyczne koszty:

| | | |
|---|---|---|
| `Push` | $1$ | |
| `Pop` | $1$ | |
| `Multipop` | $\min(k, s)$ | |

Koszty zamortyzowane:

| | | |
|---|---|---|
| `Push` | $2$ | 1 zł - za siebie + 1 zł - zostawione na wstawionym elemencie |
| `Pop` | $0$ | płaci za siebie 1 zł zabranym ze zdejmowanego elementu |
| `Multipop` | $0$ | płaci za siebie pieniędzmi zabranymi ze zdejmowanych elementów |

Starcza na opłacenie dowolnego ciągu $n$ operacji.

# Metoda księgowania (licznik binarny)

Koszt faktyczny: zmiana 1 bitu – 1 zł.
Koszt zamortyzowany:

- zmiana 1 bitu z $0$ na $1$ – 2 zł (1 zł zmiana bitu + 1 zł zostawione na zmienionym bicie)
- zmiana 1 bitu z $1$ na $0$ – 0 zł (płacimy 1 zł zabranym ze zmienianego bitu)

(Pieniędzy nigdy nie brakuje.)
Koszt zamortyzowany `Increment`:
za zerowanie bitów w pętli `while` płacimy pieniędzmi zabranymi z tych bitów.
Co najwyżej 1 bit zmieniamy na $1$ (w wierszu 6) – koszt 2 zł.
Stąd zamortyzowany koszt `Increment`: $2 = O(1)$.

# Metoda potencjału

$D_0$ – początkowa struktura danych,
$D_i$, dla $i \geq 1$ – struktura po $i$-tej operacji.
Dla $i \geq 0$, $\Phi(D_i)$ – *potencjał* $D_i$.
Dla $i \geq 1$, $c_i$ – faktyczny koszt $i$-tej operacji.
Dla $i \geq 1$, $\hat{c}_i$ – zamortyzowany koszt $i$-tej operacji:

$$\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1})$$

Całkowity koszt zamortyzowany ciągu $n$ operacji:

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} (c_i + \Phi(D_i) - \Phi(D_{i-1})) = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0)$$

Jeśli $(\forall n \geq 1)$ $\Phi(D_n) - \Phi(D_0) \geq 0$, to (zawsze)
$\sum_{i=1}^{n} \hat{c}_i \geq \sum_{i=1}^{n} c_i$ (koszt zamort. $\geq$ koszt fakt.)

# Metoda potencjału (stos)

Potencjał – wysokość stosu.
Mamy $\Phi(D_0) = 0$ (stos początkowo pusty) oraz
$(\forall i)\Phi(D_i) \geq 0 = \Phi(D_0)$ (wysokość nieujemna).

# Metoda potencjału (stos)

Potencjał – wysokość stosu.
Mamy $\Phi(D_0) = 0$ (stos początkowo pusty) oraz
$(\forall i)\Phi(D_i) \geq 0 = \Phi(D_0)$ (wysokość nieujemna).
Jeśli stos $D_{i-1}$ zawiera $s$ elementów i $i$-ta operacja – Push,
to przyrost potencjału: $\Phi(D_i) - \Phi(D_{i-1}) = (s+1) - s = 1$.
Koszt zamortyzowany Push:
$\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1}) = 1 + 1 = 2$.

# Metoda potencjału (stos)

Potencjał – wysokość stosu.
Mamy $\Phi(D_0) = 0$ (stos początkowo pusty) oraz $(\forall i)\Phi(D_i) \geq 0 = \Phi(D_0)$ (wysokość nieujemna).
Jeśli stos $D_{i-1}$ zawiera $s$ elementów i $i$-ta operacja – `Push`, to przyrost potencjału: $\Phi(D_i) - \Phi(D_{i-1}) = (s+1) - s = 1$.
Koszt zamortyzowany `Push`:
$\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1}) = 1 + 1 = 2$.
Przyrost potencjału `Multipop(S,k)` przy zdejmowaniu $k' = \min(s,k)$ elementów: $\Phi(D_i) - \Phi(D_{i-1}) = -k'$. Stąd koszt zamortyzowany `Multipop(S,k)`:
$\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1}) = k' - k' = 0$
Podobnie – koszt zamortyzowany `Pop`: $0$.
Koszt zamortyzowany każdej operacji jest $O(1)$.

# Metoda potencjału (licznik binarny)

Potencjał $b_i$ – liczba bitów równych $1$.
(Początkowo: $b_0 = 0$ i zawsze $b_i \geq 0 = b_0$.)
Niech $i$-ta operacja `Increment` zeruje $t_i$ bitów. Jej faktyczny koszt: $c_i \leq t_i + 1$.
Potencjał po jej wykonaniu: $b_i \leq b_{i-1} - t_i + 1$.
Przyrost potencjału:
$\Phi(D_i) - \Phi(D_{i-1}) \leq (b_{i-1} - t_i + 1) - b_{i-1} = 1 - t_i$.
Koszt zamortyzowany:
$\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1}) \leq (t_i + 1) + (1 - t_i) = 2$.

# Metoda potencjału (licznik binarny)

Potencjał $b_i$ – liczba bitów równych $1$.
(Początkowo: $b_0 = 0$ i zawsze $b_i \geq 0 = b_0$.)
Niech $i$-ta operacja `Increment` zeruje $t_i$ bitów. Jej faktyczny koszt: $c_i \leq t_i + 1$.
Potencjał po jej wykonaniu: $b_i \leq b_{i-1} - t_i + 1$.
Przyrost potencjału:
$\Phi(D_i) - \Phi(D_{i-1}) \leq (b_{i-1} - t_i + 1) - b_{i-1} = 1 - t_i$.
Koszt zamortyzowany:
$\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1}) \leq (t_i + 1) + (1 - t_i) = 2$.

---

Niech początkowo $b_0 \neq 0$ jedynek i po $n$ operacjach $b_n$ jedynek, gdzie $0 \leq b_0, b_n \leq k$. Koszt faktyczny: $\sum_{i=1}^{n} c_i = \sum_{i=1}^{n} \hat{c}_i - (\Phi(D_n) - \Phi(D_0)) \leq \sum_{i=1}^{n} 2 - b_n + b_0 = 2n - b_n + b_0$.
Wiadomo: $b_0 \leq k$. Stąd jeśli $n = \Omega(k)$, to koszt $n$ operacji jest $O(n)$.

# Tablice dynamiczne

Table-Insert($T$,$x$)

if *size*$[T] = 0$ then
    *przydziel 1 komórkę do table*$[T]$
    *size*$[T] \leftarrow 1$
if *num*$[T]$ $=$*size*$[T]$ then
    *przydziel* $2\cdot$*size*$[T]$ *komórek do new-table*
    *wstaw elementy z table*$[T]$ *do new-table*
    *zwolnij table*$[T]$
    *table*$[T]$ $\leftarrow$*new-table*
    *size*$[T] \leftarrow 2\cdot$*size*$[T]$
*wstaw* $x$ *do table*$[T]$
*num*$[T]$ $\leftarrow$*num*$[T] + 1$

# Koszt ciągu wstawień

Koszt faktyczny $i$-tej operacji w ciągu `Table-Insert`:

$$c_i = \begin{cases} i, & \text{jeśli } i-1 \text{ – całkowita potęga } 2 \\ 1, & \text{w p.p.} \end{cases}$$

Całkowity koszt $n$ operacji:

$$\sum_{i=1}^{n} c_i \leq n + \sum_{j=0}^{\lfloor \lg n \rfloor} 2^j < n + 2n = 3n$$

bo jest $\leq n$ operacji o koszcie $1$, a pozostałe tworzą szereg geometryczny $2^{\lfloor \lg n \rfloor} + \ldots + 2^0$.
Stąd koszt zamortyzowany jednej operacji: $3$.

# Inny sposób: księgowanie

Każdy element płaci za 3 elementarne wstawienia:

- pierwsze wstawienie siebie samego do bieżącej tablicy
- pierwsze przeniesienie siebie samego do nowej tablicy przy powiększeniu
- przeniesienie jednego z elementów przenoszonych po raz kolejny (jest ich tyle samo co elementów przenoszonych po raz pierwszy)

# Inny sposób: metoda potencjału

$\Phi(T) = 2$*num*$[T]-$*size*$[T]$.
Zawsze: $\Phi(T) \geq 0$.
Zaraz po każdym wstawieniu powiększającym tablicę (i początkowo): $\Phi(T) \geq 0$, bo *num*$[T] \geq$*size*$[T]/2$.
Wstawienie niepowiększające tablicy zwiększa $\Phi(T)$.
Bezpośrednio przed powiększeniem: $\Phi(T) =$*num*$[T]$, bo *num*$[T]$=*size*$[T]$.
Jeśli $i$-ta operacja nie zwiększa tablicy, to $size_{i-1} = size_i$ oraz:
$\hat{c}_i = c_i + \Phi_i - \Phi_{i-1} = 1 + (2num_i - size_i) - (2num_{i-1} - size_{i-1}) = 1 + (2num_i - size_i) - (2(num_i - 1) - size_i) = 3$.
Jeśli $i$-ta operacja zwiększa tablicę, to $size_i/2 = size_{i-1} = num_i - 1$, oraz: $\hat{c}_i = c_i + \Phi_i - \Phi_{i-1} = num_i + (2num_i - size_i) - (2num_{i-1} - size_{i-1}) = num_i + (2num_i - (2num_i - 2)) - (2(num_i - 1) - (num_i - 1)) = num_i + 2 - (num_i - 1) = 3$.

# Wstawianie i usuwanie elementów

`Table-Delete`

*usuń element i zmniejsz rozmiar tablicy o połowę gdy współczynnink zapełnienia tablicy $\alpha(T) =$num$[T]/$size$[T]$ spada poniżej stałej $c$. (Jeśli num$[T]$ spada do $0$ to pamięć zwalniana i size$[T] = 0$, $\alpha[T] = 1$)*

$c = 1/2$ – zły wybór: Niech $n/2$ – potęga $2$ i pierwsze $n/2$ operacji `Insert` a potem ciąg:
`I,D,D,I,I,D,D,I,I,...` (co 2 kroki kosztowna zmiana rozmiaru – łączny czas $n$ operacji: $O(n^2)$)
Wybieramy $c = 1/4$.
Potencjał (zawsze $\geq 0$ i równy $0$ dla pustego $T$):

$$\Phi(T) = \begin{cases} 2\textit{num}[T] - \textit{size}[T] & \text{jeśli } \alpha(T) \geq 1/2 \\ \textit{size}[T]/2 - \textit{num}[T] & \text{jeśli } \alpha(T) < 1/2 \end{cases}$$

# Wstawianie i usuwanie elementów

Niech $i$-ta operacja – `Table-Insert`.
Przypadek $\alpha_{i-1} \geq 1/2$: taka sama analiza jak przy ciągu samych wstawień: koszt zamortyzowany: $3$.
Przypadek $\alpha_{i-1} < 1/2$: Tablica nie może ulec powiększeniu (powiększanie tylko dla $\alpha_{i-1} = 1$).

# Wstawianie i usuwanie elementów

Niech $i$-ta operacja – `Table-Insert`.
Przypadek $\alpha_{i-1} \geq 1/2$: taka sama analiza jak przy ciągu samych wstawień: koszt zamortyzowany: $3$.
Przypadek $\alpha_{i-1} < 1/2$: Tablica nie może ulec powiększeniu (powiększanie tylko dla $\alpha_{i-1} = 1$).
Podprzypadek $\alpha_{i-1} < 1/2$ i $\alpha_i < 1/2$:
$\hat{c}_i = c_i + \Phi_i - \Phi_{i-1} = 1 + (\textit{size}_i/2 - \textit{num}_i) - (\textit{size}_{i-1}/2 - \textit{num}_{i-1}) = 1 + (\textit{size}_i/2 - \textit{num}_i) - (\textit{size}_i/2 - (\textit{num}_i - 1)) = 0$

# Wstawianie i usuwanie elementów

Niech $i$-ta operacja – `Table-Insert`.
Przypadek $\alpha_{i-1} \geq 1/2$: taka sama analiza jak przy ciągu samych wstawień: koszt zamortyzowany: $3$.
Przypadek $\alpha_{i-1} < 1/2$: Tablica nie może ulec powiększeniu (powiększanie tylko dla $\alpha_{i-1} = 1$).
Podprzypadek $\alpha_{i-1} < 1/2$ i $\alpha_i < 1/2$:

$$\hat{c}_i = c_i + \Phi_i - \Phi_{i-1} = 1 + (\mathit{size}_i/2 - \mathit{num}_i) - (\mathit{size}_{i-1}/2 - \mathit{num}_{i-1}) =$$
$$1 + (\mathit{size}_i/2 - \mathit{num}_i) - (\mathit{size}_i/2 - (\mathit{num}_i - 1)) = 0$$

Podprzypadek $\alpha_{i-1} < 1/2$ i $\alpha_i \geq 1/2$:

$$\hat{c}_i = c_i + \Phi_i - \Phi_{i-1} = 1 + (2\mathit{num}_i - \mathit{size}_i) - (\mathit{size}_{i-1}/2 - \mathit{num}_{i-1}) =$$
$$1 + (2(\mathit{num}_{i-1} + 1) - \mathit{size}_{i-1}) - (\mathit{size}_{i-1}/2 - \mathit{num}_{i-1}) =$$
$$3\mathit{num}_{i-1} - (3/2)\mathit{size}_{i-1} + 3 = 3\alpha_{i-1}\mathit{size}_{i-1} - (3/2)\mathit{size}_{i-1} + 3 <$$
$$< (3/2)\mathit{size}_{i-1} - (3/2)\mathit{size}_{i-1} + 3 = 3$$

# Wstawianie i usuwanie elementów

Niech $i$-ta operacja – `Table-Delete`.
Przypadek: $\alpha < 1/2$ i tablica nie ulega zmniejszeniu:
$\hat{c}_i = c_i + \Phi_i - \Phi_{i-1} = 1 + (\mathit{size}_i/2 - \mathit{num}_i) - (\mathit{size}_{i-1}/2 - \mathit{num}_{i-1}) = 1 + (\mathit{size}_i/2 - \mathit{num}_i) - (\mathit{size}_i/2 - (\mathit{num}_i + 1)) = 2$

# Wstawianie i usuwanie elementów

Niech $i$-ta operacja – `Table-Delete`.
Przypadek: $\alpha < 1/2$ i tablica nie ulega zmniejszeniu:
$\hat{c}_i = c_i + \Phi_i - \Phi_{i-1} = 1 + (\mathit{size}_i/2 - \mathit{num}_i) - (\mathit{size}_{i-1}/2 - \mathit{num}_{i-1}) =$
$1 + (\mathit{size}_i/2 - \mathit{num}_i) - (\mathit{size}_i/2 - (\mathit{num}_i + 1)) = 2$
Przypadek: $\alpha < 1/2$ i tablica ulega zmniejszeniu (t.j. $\mathit{size}_i/2 = \mathit{size}_{i-1}/4 = \mathit{num}_{i-1} = \mathit{num}_i + 1$):
$\hat{c}_i = c_i + \Phi_i - \Phi_{i-1} =$
$(\mathit{num}_i + 1) + (\mathit{size}_i/2 - \mathit{num}_i) - (\mathit{size}_{i-1}/2 - \mathit{num}_{i-1}) =$
$(\mathit{num}_i + 1) + ((\mathit{num}_i + 1) - \mathit{num}_i) - (2(\mathit{num}_i + 1) - (\mathit{num}_i + 1)) = 2$

# Wstawianie i usuwanie elementów

Niech $i$-ta operacja – `Table-Delete`.
Przypadek: $\alpha < 1/2$ i tablica nie ulega zmniejszeniu:
$\hat{c}_i = c_i + \Phi_i - \Phi_{i-1} = 1 + (\mathit{size}_i/2 - \mathit{num}_i) - (\mathit{size}_{i-1}/2 - \mathit{num}_{i-1}) = 1 + (\mathit{size}_i/2 - \mathit{num}_i) - (\mathit{size}_i/2 - (\mathit{num}_i + 1)) = 2$
Przypadek: $\alpha < 1/2$ i tablica ulega zmniejszeniu (t.j. $\mathit{size}_i/2 = \mathit{size}_{i-1}/4 = \mathit{num}_{i-1} = \mathit{num}_i + 1$):
$\hat{c}_i = c_i + \Phi_i - \Phi_{i-1} =$
$(\mathit{num}_i + 1) + (\mathit{size}_i/2 - \mathit{num}_i) - (\mathit{size}_{i-1}/2 - \mathit{num}_{i-1}) =$
$(\mathit{num}_i + 1) + ((\mathit{num}_i + 1) - \mathit{num}_i) - (2(\mathit{num}_i + 1) - (\mathit{num}_i + 1)) = 2$
Przypadek $\alpha_{i-1} \geq 1/2$ i $\mathit{size}_{i-1} \leq 2$: koszt stały.
Przypadek $\alpha_{i-1} \geq 1/2$ i $\mathit{size}_{i-1} > 2$ (t.j. $\mathit{size}_{i-1} \geq 4$, bo rozmiary – potęgi $2$): Tablica nie może ulec zmniejszeniu bo $\alpha_i \geq 1/4$. Koszt stały (ćw.).