Złożoność czasowa

W przypadku złożoności czasowej, z reguły wyróżniamy pewną **operację dominującą**, a czas będziemy traktować jako liczbę wykonanych operacji dominujących.
W ten sposób analiza będzie zależna jedynie od algorytmu, a nie od implementacji i sprzętu.

W przypadku sortowania, operacją dominującą jest przeważnie porównanie dwóch elementów.
W przypadku przeglądania drzewa operacją dominującą jest jedno przejście w drzewie między wierzchołkami.

- Zazwyczaj określamy pewien parametr *n*, będący rozmiarem problemu wejściowego i określamy złożoność jako funkcję *T(n)*, której argumentem jest rozmiar problemu.

- Z reguły będziemy przyjmować, że każda operacja arytmetyczna na małych liczbach daje się wykonać w jednym kroku.

- Złożoność algorytmu może być rozumiana w sensie złożoności najgorszego przypadku lub złożoności średniej.

- Złożoność najgorszego przypadku nazywamy **złożonością pesymistyczną** - jest to maksymalna złożoność dla danych tego samego rozmiaru *T(n)*.

- W praktyce ważniejsza może się okazać **złożoność średnia** lub **oczekiwana.** W tym przypadku *T(n)* jest średnią (oczekiwaną) wartością złożoności dla wszystkich problemów rozmiaru *n*. Tego typu złożoność zależy istotnie od tego, jaka się pod tym kryje przestrzeń probabilistyczna danych wejściowych. Z reguły zakładamy, że wszystkie dane wejściowe tego samego rozmiaru mogą się pojawić z tym samym prawdopodobieństwem.

Oznaczenia:

- ***D_n*** - zbiór zestawów danych wejściowych rozmiaru *n;*
- *t(d)* - liczba operacji dominujących dla zestawu danych wejściowych ***d;***
- ***X_n*** - zmienna losowa, której wartością jest *t(d)* dla $d \in D_n$;
- ***p_{nk}*** - rozkład prawdopodobieństwa zmiennej losowej X_n, tzn. prawdopodobieństwo, że dla danych rozmiaru *n* algorytm wykona *k* operacji dominujących ($k \geq 0$).
- Rozkład prawdopodobieństwa zmiennej losowej ***X_n*** wyznacza się na podstawie informacji o zastosowaniach rozważanego algorytmu.
- Gdy zbiór D_n jest skończony, przyjmuje się często model probabilistyczny, w którym każdy zestaw danych rozmiaru *n* może się pojawić na wejściu do algorytmu z jednako-wym prawdopodobieństwem.

Pesymistyczna złożoność czasowa algorytmu to funkcja:

$$W(n) = \sup \{ t(d): \mathbf{d} \in \mathbf{D_n} \},$$

gdzie sup oznacza kres górny zbioru.

Oczekiwana złożoność algorytmu to funkcja:

$$A(n) = \sum_{k \geq 0} k p_{nk}$$

(wartość oczekiwana zmiennej losowej X_n - $E(X_n)$)

Aby stwierdzić, na ile funkcje *W(n)* oraz *A(n)* są reprezentatywne dla wszystkich danych wejściowych rozmiaru *n,* uwzględnia się miary wrażliwości algorytmu.

Miara wrażliwości pesymistycznej to funkcja

$$\Delta(n) = \sup\{t(d_1) - t(d_2) : d_1, d_2 \in D_n\}$$

Przykład.

Przypuśćmy, że chcemy znaleźć pierwszą jedynkę w n-elementowej tablicy zerojedynkowej i nasz algorytm przegląda tablicę od strony lewej sprawdzając kolejne elementy. (zakładamy, że jedynka występuje w tablicy).
Niech operacją dominującą będzie sprawdzenie jednego elementu.

```
int j=0;

while (a[j]!=1) j++;

p=j// numer komórki w której występuje pierwsza jedynka
```

Rozmiar danych wejściowych: n

Operacja dominująca: *porównanie a[j]!=1*

Pesymistyczna złożoność czasowa: $W(n)=n$ (gdy „1" w ostatniej komórce)

Oczekiwana złożoność czasowa: $A(n)=?$

Załóżmy, że prawdopodobieństwo znalezienia 1 na każdym z n możliwych miejsc jest takie samo i wiadomo, że a jest w tablicy:

$$p_{nk}=1/N, \text{ dla } k=1, 2, \ldots, n$$

Wtedy:

$$A(n)=\sum_{k=1}^{n} kp_{nk} = \frac{1}{n} * \sum_{k=1}^{n} k = \frac{1}{n} * \frac{n(n+1)}{2} = \frac{n+1}{2}$$

Pesymistyczna wrażliwość czasowa: $\Delta(n)=n-1$

W notacji używanej do opisu asymptotycznego czasu działania algorytmów korzysta się z funkcji, których zbiorem argumentów jest zbiór liczb naturalnych.

Zad.1.

Przypuśćmy, że chcemy obliczyć ile jedynek występuje *w* *n*-elementowej tablicy zerojedynkowej i nasz algorytm przegląda tablicę od strony lewej sprawdzając kolejne elementy. Niech operacją dominującą będzie porównanie A[j]==1.

```
int j=0;int ile=0;

while (j<n)

{   if (A[j]==1) ile++;

    j++;}
```

Zad.2.

Dana jest tablica A o rozmiarze n. Oblicz złożoność pesymistyczną, średnią i miarę wrażliwości pesymistycznej algorytmu:

```
for (int i=0;i<n;i++)

   for (int j=0;j<n;j++)
        A[i]=i+j;
```

Zad.3.

Dana jest tablica A o rozmiarze n. Oblicz złożoność pesymistyczną, średnią i miarę wrażliwości pesymistycznej algorytmu:

```
int suma =0

for (int i=0;i<n;i++)

   suma=suma+A[i];
```

Zad.4.

Dana jest tablica A o rozmiarze n. Oblicz złożoność pesymistyczną, średnią i miarę wrażliwości pesymistycznej algorytmu:

```
bool istnieje()

{for (int i=0;i<n;i++)
     {if (A[i]<0) return true;} return false}
```

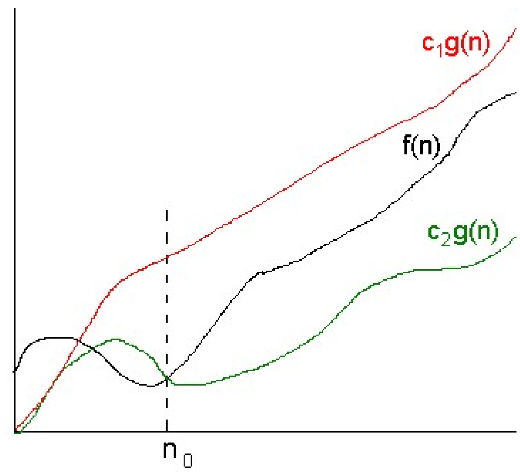
Notacja Θ

Dla danej funkcji *g(n)* przez ***Θ(g(n))*** *(„duże theta od g od n")*

oznaczamy zbiór funkcji:

$$\Theta(g(n)) = \{ f(n) : \exists c_1, c_2, n_0 > 0:\ 0 \le c_1 g(n) \le f(n) \le c_2 g(n)\ \ \forall n > n_0 \}$$

(funkcja f(n) należy do ***Θ(g(n))*** jeśli istnieją dodatnie stałe c_1 i c_2 takie, że funkcja *f(n)* może być „wstawiona " pomiędzy $c_1 g(n)$ a $c_2 g(n)$).



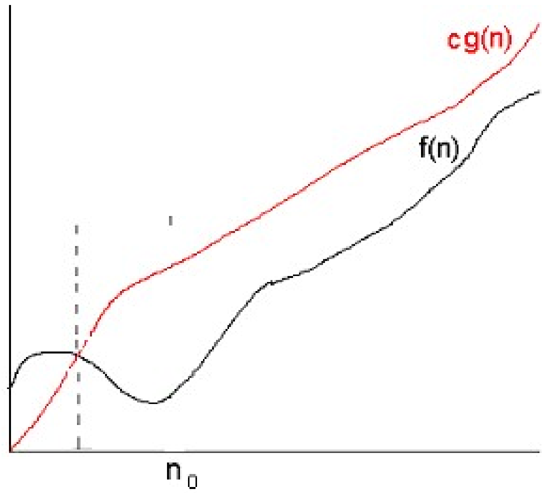
f(n)=Θ(g(n))

Notacja O

Notacja Θ asymptotycznie ogranicza funkcję od góry i od dołu. Kiedy mamy tylko **ograniczenie górne**, używamy **notacji *O***.

$$O(g(n)) = \{ f(n) : \exists c, n_0 > 0:\ f(n) \le c_2 g(n)\ \ \forall n > n_0 \}$$

Z notacji ***O*** korzystamy, gdy chcemy oszacować funkcję z góry z dokładnością do stałej.



f(n)=O(g(n))

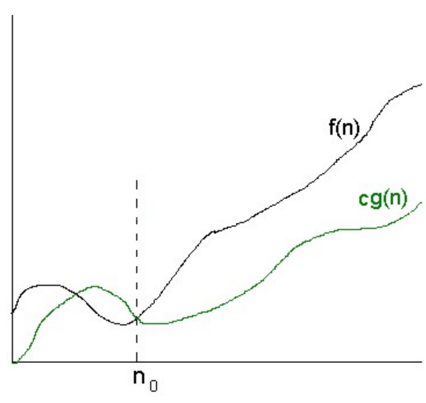
Uwaga: Notacja *Θ* jest silniejsza niż notacja *O*

Ponieważ notacja *O* odpowiada ograniczeniu górnemu, stosując ją do oszacowania pesymistycznego czasu działania algorytmu, uzyskujemy górne ograniczenie czasu działania tego algorytmu dla wszystkich danych wejściowych.

Notacja Ω

Notacja Ω asymptotycznie ogranicza funkcję od dołu.

$$\Omega(g(n)) = \{ f(n) : \exists c, n_0 > 0 :\ 0 \le cg(n) \le f(n)\ \ \forall n > n_0 \}$$



f(n)=Ω(g(n))

Zad.5. Czy $T(n) = 3n^3 + 2n^2$ jest $O(n^3)$?

Zad.6. Czy $T(n)=2n^4+3n^2+100n$ jest $O(n^4)$?

Dodawanie i mnożenie w notacji O

Reguła sumowania:

Załóżmy, że $T_1(n)$ i $T_2(n)$ są czasami wykonania fragmentów programu P_1 i P_2.

$T_1(n)$ jest $O(f(n))$ a $T_2(n)$ jest $O(g(n))$.

Wtedy czas wykonania fragmentów P_1 i P_2 jest :

$T_1(n)+T_2(n)$ jest $O(\max(f(n), g(n))$.

Reguła mnożenia:

Załóżmy, że $T_1(n)$ i $T_2(n)$ są czasami wykonania fragmentów programu P_1 i P_2.

$T_1(n)$ jest $O(f(n))$ a $T_2(n)$ jest $O(g(n))$.

Wtedy:

$T_1(n)*T_2(n)$ jest $O(f(n)* g(n)$.

Zad.7. Oblicz złożoność czasową algorytmu. Skorzystaj z reguły mnożenia lub dodawania.

```
for (int i=0;i<n;i++)

   for (int j=0;j<n;j++)
        C[i]=…;
```

Zad.8. Oblicz złożoność czasową algorytmu. Skorzystaj z reguły mnożenia lub dodawania.

```
for (int i=0;i<n;i++)

   for (int j=i;j<n;j++)
        C[i]=…;
```

Zad.9. Dana jest tablica n- elementowa A. Oblicz złożoność pesymistyczną, oczekiwaną oraz miarę wrażliwości pesymistycznej poniższego algorytmu. Co oblicza poniższy algorytm?

```
IsMember()

{int i=0;

while ((i<n) &&(A[i]!=10)) i++;

return i;}
```

Zad.10. Dana jest tablica A o rozmiarze *n* i tablica *B* o rozmiarze *m*. Co oblicza ten algorytm? Oblicz złożoność pesymistyczną, średnią, miarę wrażliwości pesymistycznej algorytmu.
Zaimplementuj funkcję o lepszej złożoności czasowej, obliczającej to samo. Oblicz złożoność pesymistyczną i średnią, miarę wrażliwości pesymistycznej tej funkcji.

```
Dane: x
int s=0
for (int i=0;i<n;i++)
    {for (int j=0;j<m;j++)
       {if ((i==0)&&(B[j]==x)) s=s++;
        if ((j==0)&&(A[j]==x)) s=s++;
        }
    }
return s;
```