Zadania na ćwiczenia

Zadanie 1

Uporządkować rosnąco wg rzędu następujące funkcje:
$2^{n!}$, n^2 , $(2009!)n^2$, $n^3 - n^2$, n^3 , $n^3 + n^2$, 2^n , $n^{2009!}$, $n!$, $n \log n$, $\log n$.
Czy któreś z tych funkcji są równoważne według rzędu (są asymptotycznie podobne)?

Znajdź i wykaż (udowodnij) zależności pomiędzy funkcjami w notacji asymptotycznej:
— $2n$ w odniesieniu do $(20!)n$,
— $n^2 - n$ w odniesieniu do $n^2 + n$,
— $\log_2 n$ w odniesieniu do n,
— 3^n w odniesieniu do 2^n
— $n!$ w odniesieniu do 2^n

Rozwiązanie

Uporządkowanie:
$\log n$, $n \log n$, n^2 , $(2009!)n^2$, $n^3 - n^2$, n^3 , $n^3 + n^2$, $n^{2009!}$, 2^n , $n!$, $2^{n!}$

a) $2n \in \Theta((20!)n)$. Ponieważ $\exists\ c_1 = \frac{1}{(20!)}, c_2 = 1$ takie, że $\forall n \geqslant 1$

$$c_1 \cdot (20!)n = \frac{1}{(20!)} \cdot (20!)n = n \leqslant 2n \leqslant 1 \cdot (20!)n = c_2 \cdot ((20!)n.$$

b) $n^2 - n \in \Theta(n^2 + n)$. Ponieważ $\exists\ c_1 = \frac{1}{2}, c_2 = 1$ takie, że (podanie warunku na n jest bardzo ważne!) $\forall n \geqslant 3$

$$c_1 \cdot (n^2 + n) = \frac{1}{2} \cdot (n^2 + n) \overset{(1)}{\leqslant} n^2 - n \leqslant 1 \cdot (n^2 + n) = c_2 \cdot (n^2 + n).$$

Gdzie nierówność (1) jest prawdziwa $\forall n \geqslant 3$ gdyż:

$$\begin{aligned}
\frac{1}{2} \cdot (n^2 + n) &\leqslant n^2 - n \\
\frac{1}{2}n^2 + \frac{1}{2}n &\leqslant n^2 - n \\
\frac{3}{2}n &\leqslant \frac{1}{2}n^2 \ , \ (n \geqslant 0) \text{ stąd} \\
\frac{3}{2} &\leqslant \frac{1}{2}n \\
3 &\leqslant n
\end{aligned}$$

c) $\log_2 n \in O(n)$. Ponieważ $\exists\ c_1 = 1$ takie, że $\forall n \geqslant 1$

$$\log_2 n \overset{(1)}{\leqslant} n = 1 \cdot n = c_1 \cdot n.$$

Gdzie nierówność (1) jest prawdziwa $\forall n \geqslant 1$ gdyż:

$n = \log_2 2^n \geqslant \log_2 n$, (jest to równoważne do nierówności) $2^n \geqslant n$, która jest prawdziwa, gdyż dla $n = 1$ mamy $2^n = 2, n = 1$, a dla każdego kolejnego $n > 1$ w ciągu 2^n przemnażamy wartość poprzedniego elementu przez 2 a dla ciągu n dodajemy do poprzedniego elementu 1. Stąd elementy z ciągu 2^n muszą mieć większą wartość niż elementy z ciągu n.

Można też w tym miejscu narysować wykres.

d) $3^n \in \Omega(2^n)$. Ponieważ $\exists\ c_1 = 1$ takie, że $\forall n \geqslant 1$

$$c_1 \cdot 2^n = 1 \cdot 2^n \overset{(1)}{\leqslant} 3^n$$

Gdzie nierówność (1) jest prawdziwa $\forall n \geqslant 1$, gdyż dla $n = 1$ mamy $2^n = 2, 3^n = 3$, a dla każdego kolejnego $n > 1$ w ciągu 2^n przemnażamy wartość poprzedniego elementu przez 2 a dla ciągu 3^n przemnażamy wartość poprzedniego elementu przez 3. Stąd dla każdego $n > 1$ elementy z ciągu 3^n muszą mieć większą wartość niż elementy z ciągu 2^n.

e) $n! \in \Omega(2^n)$ Ponieważ $\exists\ c_1 = \frac{1}{2}$ takie, że $\forall n \geqslant 1$

$$c_1 \cdot 2^n = \frac{1}{2} \cdot 2^n \overset{(1)}{\leqslant} n!$$

Gdzie nierówność (1) jest prawdziwa $\forall n \geqslant 1$, gdyż dla $n = 1$ mamy $\frac{1}{2} \cdot 2^n = 1, n! = 1$, a dla każdego kolejnego $n > 1$ w ciągu 2^n przemnażamy wartość poprzedniego elementu przez 2 a dla ciągu $n!$ przemnażamy wartość poprzedniego elementu przez $n - 1 \geqslant 2$. Stąd dla każdego $n > 1$ elementy z ciągu $n!$ muszą mieć większą wartość niż elementy z ciągu 2^n.

Zadanie 2

Dla danej liczby $n \in \mathbb{N}$ należy wyznaczyć wszystkie liczby pierwsze $p \leqslant n$. Porównać złożoność obliczeniową dwóch algorytmów realizujących to zadanie:

— sprawdzamy kolejno dla każdej liczby naturalnej $2 \leqslant k \leqslant n$ czy jest liczbą pierwszą,

— sito Eratostenesa z listy $2, ..., n$ wykreślamy kolejno: liczby podzielne przez 2, liczby podzielne przez 3,...,

Rozwiązanie

Odp : a) n^2. b) $n \ln \ln n$ ($\ln n = \log_e n$)

Odpowiedź jak powyżej jest wystarczająca na kolokwium. Poniżej uzasadnienie czemu jest to n^2 i $n \ln \ln n$.

a) Dla każdej liczy od $2, .., n$ musimy wykonać jakieś działanie. Stąd w programie musi być pętla:

```
for i:=2 to n
```

W tej pętli naszym zadaniem jest sprawdzić czy dana liczba jest liczbą pierwszą. Czyli inaczej mówiąc sprawdzić czy dzieli się przez jakąś liczbę z przedziału $1, ..., n - 1$ (lub optymalniej z przedziału $1, ..., \sqrt{n}$) (w zadaniu powinna być podana dokładnie metoda sprawdzania podzielności - założymy że sprawdzamy podzielność przez wszystkie liczby od 1 do $n - 1$, gdyż jest to prostsze do analizy). Stąd w obrębie pierwszej pętli wystąpić musi druga pętla sprawdzająca podzielność:

```
for j:=1 to i-1
sprawdź czy się dzieli
```

Sumarycznie nasz program będzie wyglądał następująco (Uwaga! To nie jest pseudokod algorytmu, tylko szkic jego postaci, inaczej mówiąc tego co poniżej nie można nazwać pseudokodem algorytmu) :

```
for i:=2 to n {
  for j:=2 to i-1 {
    sprawdź czy się dzieli
  }
{
```

Stąd program wykonuje $1+2+3+...+n-1$ operacji (odpowiednio sprawdzenie podzielności dla liczb $3,4,5,...,n$). Ciąg $1+2+3+...+n-1$ sumuje się do $\frac{n(n-1)}{2} = \frac{n^2-n}{2}$. Wiemy, że $\frac{n^2-n}{2} \in \Theta(n^2)$, stąd złożoność algorytmu jest równa n^2.

b) Mamy daną tablicę po której przechodzimy wykreślając wielokrotności liczb pierwszych. Dla każdej liczy od $2,..,n$ musimy wykonać jakieś działania. Stąd w programie musi być pętla:

```
for i:=2 to n
```

W tej pętli naszym zadaniem jest sprawdzić czy dana liczba jest zaznaczona jako złożona („wykreślona z tablicy"). Jeżeli tak to przechodzimy dalej jeżeli nie to wykreślamy z tablicy wszystkie liczby będące wielokrotnością danej liczby (począwszy od jej kwadratu), dla każdej liczby pierwszej i wykonujemy więc $\frac{n}{i}$ operacji. Podany algorytm wyglądałby mniej więcej następująco:

```
jeżeli i jest liczbą pierwszą
for (j:= i*i; j<=n; j+=i)
 oznacz j-tą liczbę jako złożoną
```

Sumarycznie nasz program będzie wyglądał następująco (Uwaga! To nie jest pseudokod algorytmu, tylko szkic jego postaci.) :

```
for i:=2 to n
  jeżeli i jest liczbą pierwszą
  for (j:= i*i; j<=n; j+=i)
    oznacz j-tą liczbę jako złożoną
```

Stąd program dla każdej liczby pierwszej i wykona $\frac{n}{i}$ operacji. Suma operacji będzie w tym wypadku równa:

$$\sum_{i \leqslant \sqrt{n}} \frac{n}{i}$$

Wyłączając n przed sumę otrzymamy

$$n \cdot \sum_{i \leqslant \sqrt{n}} \frac{1}{i}, \text{ gdzie } i \text{ jest liczbą pierwszą.}$$

Z Twierdzeń z teorii liczb wiemy, że $\sum_{i \leqslant \sqrt{n}} \frac{1}{i} = \ln \ln n$. Stąd złożoność obliczeniowa tego algorytmu wynosi $n \ln \ln n$.

Uwaga! Jako, że rozwiązanie tego zadania wymaga wiedzy nadprogramowej, zadanie tego typu na pewno nie pojawi się na kolokwium. Zadanie

z oszacowaniem złożoności będzie zadaniem mniej więcej takiej klasy jak zadanie 3.

Zadanie 3

Dany jest wielomian $W(x) = a_0 + a_1x + a_2x^2 + \ldots + a_{n-1}x^{n-1} + a_nx^n$. Chcemy obliczyć wartość tego wielomianu w punkcie x_0, możemy zrobić to na dwa sposoby:

— obliczając wprost wartość $W(x_0) = a_0 + a_1x_0 + a_2x^2 + \ldots + a_nx^n$

— korzystając z tzw. *schematu Hornera* przekształcając wielomian do postaci

$$\begin{aligned} W(x) &= a_0 + x(a_1 + a_2x + \ldots + a_{n-1}x^{n-2} + a_nx^{n-1}) = \\ &= a_0 + x(a_1 + x(a_2 + \ldots + a_{n-1}x^{n-3} + a_nx^{n-2})) = \\ &= a_0 + x(a_1 + x(a_2 + \ldots + x(a_{n-1}x + a_n)\ldots)) \end{aligned}$$

Wyznaczyć złożoność obliczeniową algorytmu w stosunku do tego, gdy jako operacje elementarne przyjmiemy a) liczę dodawań, b) liczbę mnożeń, c) liczbę wszystkich operacji arytmetycznych.

Rozwiązanie

Interesuje nas tylko to jakiego rzędu to są funkcje a nie ich dokładne wartości:

a) W I przypadku n. W II przypadku n.

b) W I przypadku n^2 (znów jako suma $1 + 2 + ... + n$). W II przypadku n.

c) W I przypadku n^2. W II przypadku n.

Zadanie 4

Sortowania n^2.

Wyszukiwanie binarne $\log_2 n$ (gdyż w kazdym kroku dzielimy przedział wyszukiwania na dwa).

Wyznaczenie F_n - iteracyjnie n operacji, nie będziemy się zajmować szacowaniem złożoności alg. rekurencyjnych.

Informacje końcowe

Jedno zadanie na kolokwium będzie obejmowało uporządkowanie funkcji wg rzędu i wskazanie zależności asymptotycznej pomiędzy wybranymi parami oraz jedno zadanie na kolokwium będzie obejmowało oszacowanie złożoności obliczeniowej jakiegoś algorytmu/algorytmów.