Temat: Poprawność całkowita i częściowa algorytmu.
Złożoność obliczeniowa algorytmu.
Złożoność czasowa średnia i pesymistyczna.

I. Sprawy organizacyjne

a) Literatura

1. A. V. Aho, J.E. Hopcroft, J. D. Ullman - Projektowanie i analiza algorytmów komputerowych
2. L. Banachowski, K. Diks, W. Rytter – Algorytmy i struktury danych
3. (!)T. H. Cormen, C. E. Leiserson, R. L. Rivest – Wprowadzenie do algorytmów
4. (!)A. Drozdek, D. L. Simon – Struktury danych w języku C
5. D. Harel – Rzecz o istocie informatyki. Algorytmika
6. (!)R. Neapolitan, K. Naimipour, Podstawy algorytmów z przykładami w C++

(!) Literatura podstawowa

b) Forma egzaminu, zasady zwolnienia i dopuszczenia do egzaminu

- test wielokrotnego wyboru z pytaniami otwartymi z konkretną krótką odpowiedzią,
- dopuszczanie do egzaminu (zarówno zasadniczego jak i poprawkowego) wyłącznie po zaliczaniu pracowni specjalistycznej i ćwiczeń,
- osoby, które otrzymają ocenę 5.0 z pracowni i ćwiczeń będą zwolnione z egzaminu z oceną 5.0,
- do wyniku punktowego egzaminu będzie dodawana liczba punktów zdobytych w konkursach* ogłaszanych na wykładach,
- skala ocen z egzaminu jest podana w sylabusie wykładu
- osoby, które wezmą udział w *Mistrzostwach Polski w Programowaniu Zespołowym* lub w eliminacjach *ACM European Central* i rozwiążą poprawnie co najmniej dwa zadania będą zwolnione z egzaminu z oceną 5.0**
- osoby, które wezmą udział w *Mistrzostwach Polski w Programowaniu Zespołowym* lub w eliminacjach *ACM European Central* i i zajmą miejsca od I do III będą zwolnione z egzaminu, pracowni specjalistycznej i ćwiczeń z oceną 5.0**

* konkurs z wykładu: wygrywa pierwsza z osób, która nadeśle poprawne i efektywne rozwiązanie podanego na wykładzie problemu komputerowego. Warunkiem koniecznym udziału w

konkursie jest obecność na wykładzie, na którym został ogłoszony konkurs.

** zastrzega się możliwość przeprowadzenia eliminacji do udziału w zawodach w przypadku dużej liczby chętnych.

nazwa przedmiotu:

ALGORYTMY + STRUKTURY DANYCH

efekt kształcenia przedmiotu:
Umiejętność projektowania oraz implementacji *efektywnych obliczeniowo* rozwiązań *problemów komputerowych*.

II. Problem komputerowy

Analiza algorytmów to dział informatyki zajmujący się szukaniem najefektywniejszych, poprawnych algorytmów dla danych problemów komputerowych.

Problem komputerowy to zadanie przeznaczone do realizacji na maszynie cyfrowej z określonym warunkiem początkowym (WP) i końcowym (WK).

WP – warunek początkowy – specyfikacja (dokładny opis lub formuła logiczna) danych wejściowych problemu

WK – warunek końcowy – specyfikacja (dokładny opis lub formuła logiczna) danych wyjściowych (wynikowych) uzyskanych dla danych wejściowych spełniających **WP**

Rozwiązaniem problemu komputerowego jest algorytm, którego dane wejściowe spełniają WP, a wyniku WK.

III. Poprawność całkowita i częściowa algorytmu

Algorytm **A** jest częściowo poprawny względem danego warunku **WP** i danego warunku **WK** wtedy i tylko wtedy, gdy dla dowolnych danych wejściowych spełniających warunek **WP**, jeżeli algorytm **A** zatrzymuje się, to dane wyjściowe algorytmu spełniają warunek **WK**.

Algorytm **A** jest całkowicie poprawny względem danego warunku **WP** i danego warunku **WK** wtedy i tylko wtedy, gdy dla dowolnych danych wejściowych spełniających warunek **WP** algorytm **A** zatrzymuje się i dane wyjściowe tego algorytmu spełniają warunek **WK**.

Przykład 1

Formalny specyfikacja WP i WK
WP: $n>0 \wedge n \in \mathrm{N}$
WK: $(s=1+3+5+...+n \wedge n \bmod 2 \neq 0) \vee (s=1+3+5+...+n-1 \wedge n \bmod 2=0)$

Nieformalna specyfikacja WP i WK
WP: *n* – liczba naturalna większa od zera
WK: *s* – suma kolejnych liczb nieparzystych nie większych niż *n*

Algorytm (pseudokod)

```
s=0; i=1;
while (i!=n+2)
   {
     s=s + i;
     i+=2;
    };
```

Powyższy algorytm jest poprawny częściowo, ale nie całkowicie. Dla *n* parzystego pętla nie ma stopu, ale dla dowolnego *n* nieparzystego pętla kończy się po skończonej liczbie kroków i wartość końcowa zmiennej *s* spełnia **WK**.

IV. Złożoność obliczeniowa algorytmu

Złożoność obliczeniowa algorytmu to ilość zasobów komputerowych, potrzebnych do jego wykonania. Zasoby komputerowe to **czas** działania i **ilość** pamięci zajmowanej prze dane wejściowe, wynikowe i struktury pomocnicze używane w algorytmie.

Złożoność pamięciowa algorytmu zostanie zdefiniowana na pierwszym z wykładów poświęconym strukturom danych

V. Złożoność czasowa algorytmu

Złożoność czasowa algorytmu określa „czas" realizacji algorytmu.
Złożoność czasowa niezależna od:

- szybkości procesora, który wykonuje zaimplementowany algorytm (program),
- wyboru języka programowania, w którym wykonana jest implementacja algorytmu.

a) Rozmiar problemu komputerowego

Rozmiar problemu komputerowego to rozmiar tych danych wejściowych, których ilość wpływa na czas wykonania algorytmu, tzn. im większa jest ilość tych danych, tym dłużej realizuje się algorytm.

Przykład 2

Problem A1
WP: $a_1, a_2, ..., a_n$- ciąg liczb całkowitych ($n > 0$)
WK: Ciąg $a_1, a_2, ..., a_n$ posortowany niemalejąco

Rozmiar zadania: n – długość ciągu, który należy posortować

Problem A2
WP: $a_0, a_1, ..., a_n$- ciąg liczb rzeczywistych ($n \geq 0$) definiujący współczynniki danego wielomianu W, x- dana liczba rzeczywista
WK: Liczba W(x) – wartość wielomianu W dla argumentu x

Rozmiar zadania: n - stopień wielomianu W(x)

Problem A3
WP: $t_1, t_2, ..., t_n$- ciąg znaków tekstu ($n > 0$)
$w_1, w_2, ..., w_m$- ciąg znaków wzorca ($n \geq m > 0$)

WK: *p*- zmienna logiczna, która przyjmuje wartość 1 (prawda), gdy wzorzec występuje w tekście, a 0 (fałsz), gdy wzorzec nie występuje w tekście

Rozmiar zadania: *n* – długość wzorca, *m* – długość tekstu

b) operacja elementarna

Operacja elementarna (inaczej operacja dominująca) to operacja charakterystyczna dla danego algorytmu. To taka operacja, której łączna liczba wykonań jest proporcjonalna do rozmiaru zadania, tzn. im większy rozmiar zadania, tym więcej razy realizuje się operacja elementarna.

Przykład 3

Problem A1 - operacją elementarną jest operacja porównywania elementów sortowanego ciągu albo operacja przestawiania elementów ciągu w czasie sortowania.

Problem A2 - operacją elementarną jest operacja arytmetyczna mnożenia albo operacja arytmetyczna dodawania realizowana w procesie obliczania wartości wielomianu dla danego *x*.

Problem A3 – operacja porównywania znaków wzorca ze znakami tekstu w procesie sprawdzania, czy wzorzec występuje w tekście.

Za jednostkę złożoności czasowej przyjmuje się wykonanie jednej operacji elementarnej (dominującej). Złożoność czasowa algorytmu jest funkcją parametru (parametrów) rozmiaru zadania.

c) **Złożoność czasowa średnia i pesymistyczna**

Nieformalnie

Pesymistyczna złożoność czasowa to liczba wykonanych operacji elementarnych dla danych „najgorszego" przypadku.

Średnia (oczekiwana) złożoność czasowa to liczba wykonanych operacji elementarnych dla danych „typowego" przypadku.

Formalnie

Oznaczenia

D_n – zbiór zestawów możliwych danych wejściowych rozmiaru n (dla uproszczenia zapisu przyjmujemy, że rozmiar problemu zależy od jednego parametru - n)

$t(d)$ – liczba operacji elementarnych wykonanych dla danych wejściowych d

$pr(d)$ – prawdopodobieństwo, że dane d są danymi wejściowymi algorytmu

Definicje

Pesymistyczna złożoność czasowa algorytmu to funkcja

$$T_{max}(n) = \max\{t(d) : d \in D_n\}$$

Średnia (oczekiwana) złożoność czasowa algorytmu to funkcja

$$T_{śr}(n) = \sum_{d \in D_n} pr(d) \cdot t(d)$$

Przykład 4

Problem wyszukiwania ustalonej liczby w ciągu nieuporządkowanym

WP: A: $a_0, a_2, ..., a_{n-1}$- ciąg liczb całkowitych ($n > 0$). Liczby w ciągu są różne.
x – szukana wartość. x jest liczbą całkowitą.

WK: zmienna logiczna *jest*=1, gdy $\exists_{i \in \{1..n\}} a_i = x$ oraz *jest*=0 w przeciwnym przypadku

Algorytm

$i = 0$;

while ($i<n$ && a_i!=x) i++;

jest=$i<n$;

Operacja elementarna: porównania między elementami ciągu A a liczbą x.
Rozmiar danych: n - długość ciągu

Pesymistyczna złożoność czasowa algorytmu

Dane „najgorszego" przypadku to ciąg, w którym *x* nie występuje albo występuje w indeksie *n*-1.

$$T_{max}(n) = \max\{t(d) : d \in D_n\} = n$$

Średnia złożoność czasowa algorytmu

Zakładamy, że prawdopodobieństwo znalezienia liczby *x* w ciągu *A* jest równe *p*. Prawdopodobieństwo, że *x* występuje na każdej z *n* pozycji w ciągu jest takie samo i wynosi $\frac{p}{n}$.

$$T_{śr}(n) = \sum_{d \in D_n} pr(d) \cdot t(d) = pr(x \notin A) \cdot t(x \notin A) +$$

$$+ \sum_{i=0}^{n-1} pr(x = a_i) \cdot t(x = a_i) = (1 - p) \cdot n + \sum_{i=0}^{n-1} \frac{p}{n} \cdot (i + 1) =$$

$$= (1-p) \cdot n + \frac{p}{n} \cdot \frac{1+n}{2} \cdot n = n + \frac{p}{2}(1-n)$$

VI. Rząd funkcji

Przy formułowaniu ostatecznych wniosków dotyczących efektywności czasowej (złożoności czasowej algorytmu) bierze się pod uwagę nie tyle dokładną funkcję kosztu, ile jej rząd (klasę wzrostu funkcji).

Na przykład, jeżeli dwa algorytmy *A* i *B* rozwiązujące ten sam problem mają złożoność pesymistyczną wyrażającą się odpowiednio wzorami:

$$T_{\max}{}^{A}(n) = n^2 + 2n \quad i \quad T_{\max}{}^{B}(n) = 3n^2$$

to różnica realnego czasu wykonania obydwu algorytmów jest stosunkowo niewielka, nawet dla dużego *n*. Na czas realizacji algorytmu zasadniczo wpływa bowiem w obydwu funkcjach fakt, że są wielomianami stopnia 2.

n	$0{,}1n^2$	$0{,}1n^2+n+100$
10	10	120
20	40	160
50	250	400
100	1000	1200
1000	100000	101100

$$T_{\max}{}^{A}(n) = n^2 + 2n \quad = \Theta(n^2) = T_{\max}{}^{B}(n) = 3n^2$$

Kategorie złożoności:

$$\Theta(\log n) \quad \Theta(n) \quad \Theta(n \log n) \quad \Theta(n^2) \quad \Theta(n^j) \quad \Theta(n^k) \quad \Theta(a^n) \quad \Theta(b^n) \quad \Theta(n!) \quad \Theta(n^n)$$

gdzie $k > j > 2$ oraz $b > a > 1$.

VII. Porównanie czasów realizacji algorytmu wykładniczego na dwóch komputerach o różnej szybkości wykonania operacji elementarnej

Pewien algorytm ma złożoność $\Theta(2^n)$. Załóżmy, ze mamy do dyspozycji dwa komputery:

- wolniejszy – taki, który jedną operację elementarną wykonuje w czasie 10^{-6} *s*,
- szybszy – dokładnie 1000 razy szybszy, który jedną operację elementarną wykonuje w czasie 10^{-9} *s*

W poniższej tabelce zobrazowano wzrost realnego czasu wykonania algorytmu dla rosnącego rozmiaru zadania

n - rozmiar zadania	20	50	100	200
czas realizacji na wolniejszym procesorze $(2^n / 10^6)$	1,04 s	35,7 lat	$4 \cdot 10^{14}$ wieków	$5 \cdot 10^{44}$ wieków
czas realizacji na szybszym procesorze $(2^n / 10^9)$	0,001 s	13 dni	$4 \cdot 10^{11}$ wieków	$5 \cdot 10^{41}$ wieków

Z powyższej tabelki wynika, że „rozsądny" czas wykonania algorytmu o złożoności $\Theta(2^n)$ przestaje być „rozsądny" przy stosunkowo niewielkim *n*, bo *n*=50.