EGZAMIN, 8 II 2009

Imię i nazwisko:

Nr indeksu:

Nr grupy:

Uwaga! Sprawdzian jest testem wielokrotnego wyboru, gdzie wszystkie możliwe kombinacje odpowiedzi są dopuszczalne (tj. zarówno wszystkie odpowiedzi poprawne, część odpowiedzi poprawna jak i brak odpowiedzi poprawnych). Pytanie jest uznane za poprawnie rozwiązane (tj. +1pkt) wtedy i tylko wtedy gdy jedynie wszystkie jego poprawne odpowiedzi są wytypowane. Odpowiedzi, tj. litery właściwych podpunktów, należy ostatecznie przepisać do załączonej na końcu testu tabelki. **Tylko zawartość tabelki podlega weryfikacji**. Życzymy powodzenia ...

1. Niech $f(n) = 2^{\lg n!}$, wtedy prawdą jest, że:
   (a) [–] $f(n) = O\left(n^{\lg n} + n^5\right)$,
   (b) [–] $f(n) = \Theta\left(\frac{1}{2}\lg n!\right)$,
   (c) [+] $f(n) = O\left(\frac{n^{n+3}}{n^2}\right)$, dla $n \neq 0$.

2. Rozważmy funkcję $f : \mathbb{N} \to \mathbb{R}^+ \cup \{0\}$ postaci $f(n) = n \lg n^2$, wtedy:
   (a) [–] $f(n) = O\left(\frac{1}{n} \cdot f(n) + c\right)$, gdzie c jest pewną dodatnią stałą mniejszą niż 10^3,
   (b) [+] $f\left(n^2\right) = \Omega\left(\sqrt{n} \cdot f(n)\right)$,
   (c) [+] $f(n) = O(c \cdot f(n) + c)$, gdzie c jest pewną dodatnią stałą.

3. Załóżmy, że złożoność czasową pewnego algorytmu A określa funkcja $T(A, n) = n^2$, gdzie n jest rozmiarem danych wejściowych. Komputer K wykonuje rozważany algorytm dla danych rozmiaru 64 w ciągu 4 sekund, tj. $T_K(A, 64) = 4$. Stąd:
   (a) [–] $T_K(A, 512) = 512$,
   (b) [+] $T_K(A, 1024) = 1024$,
   (c) [–] w ciągu 16 sekund komputer K wykona rozważany algorytm dla danych wejściowych rozmiaru co najmniej 256.

4. Rozważmy następujący algorytm

```
void Algorytm(int n) {
   Alg1(n);
   for (i=0;i<n;i++) {
      Alg2(n);
   }
}
```

   gdzie Alg_1 oraz Alg_2 są algorytmami o złożoności czasowej odpowiednio $A(Alg_1, n) = O(n \lg n)$, $W(Alg_1, n) = \Omega\left(n^2\right)$ oraz $T(Alg_2, n) = \Theta(n \lg n)$, stąd:
   (a) [–] $T(Algorytm, n) = O\left(n^2\right)$,
   (b) [+] $A(Algorytm, n) = O\left(n^3\right)$,
   (c) [+] $W(Algorytm, n) = \Omega\left(n^2 \lg n\right)$.

5. Rozważmy następujący algorytm

```
int Cos(int n) { // wp:  n∈ ℕ i n>1
   int i=1;
   while (i<n) i=sqr(i); // sqr(i) = i^2
   return i; // wk:  i=n
}
```

wtedy:

(a) [–] program Cos jest całkowicie poprawny w strukturze liczb naturalnych,

(b) [–] program Cos jest częściowo poprawny w strukturze liczb naturalnych,

(c) [+] program Cos jest całkowicie poprawny w strukturze liczb naturalnych przy założeniu, że operator sqr zdefiniujemy tak: $sqr\,(i) =_{def} (i + 1)$.

6. Rozważmy następujący algorytm

```
int Cos(int n) { // wp:  n∈ ℕ
   int i=0, p=1, result=1;

   while (i<n) {
      i=i+1;
      p=2*p;
      result=result+p;
   }
   return result;
}
```

wtedy:

(a) [+] niezmiennikiem pętli w programie Cos jest formuła $p = 2^i$,

(b) [–] niezmiennikiem pętli w programie Cos jest formuła $p = 2^i \wedge result = p$,

(c) [+] warunkiem końcowym w programie Cos jest $result = 2^{n+1} - 1$.

7. Co jest warunkiem końcowym poniższego algorytmu rekurencyjnego, jeżeli założymy, że *root* jest dowiązaniem do korzenia pewnego niepustego drzewa binarnego, którego wierzchołki indeksowane są wartościami typu *int*?

```
int Cos(TreeNode root) {
   if (root==NULL) return 0;
   if ((root.et mod 2)==0) return Cos(root.left)+Cos(root.right);
   else return Cos(root.left)+Cos(root.right)+root.et;
}
```

(a) [–] Liczba wierzchołków drzewa o korzeniu *root*, etykietowanych wartościami nieparzystymi.

(b) [+] Suma wszystkich wierzchołków drzewa o korzeniu *root*, etykietowanych wartościami nieparzystymi.

(c) [–] Suma wszystkich wierzchołków drzewa o korzeniu *root*, etykietowanych wartościami parzystymi.

8. Które ze zdań jest prawdziwe:

(a) [–] sprawdzenie, czy dany element należy do nieuporządkowanego uniwersum rozmiaru n^2 wymaga $O(n \lg n)$ porównań,

(b) [–] sprawdzenie, czy dany element należy do uporządkowanego uniwersum rozmiaru n wymaga $\Omega(\sqrt{n})$ porównań,

(c) [–] koszt czasowy, mierzony liczbą porównań, optymalnego algorytmu wyszukania elementu maksymalnego w nieuporządkowanym uniwersum rozmiaru 10^8 wynosi 10^7.

9. Które ze zdań jest prawdziwe:

   (a) [+] sprawdzenie algorytmem BinSearch, czy dany element należy do uporządkowanego uniwersum rozmiaru n^2 wymaga $O(n)$ porównań,

   (b) [–] koszt czasowy algorytmu BinSearch dla poprawnych danych rozmiaru 10^4 wynosi co najmniej 16 porównań,

   (c) [+] koszt czasowy algorytmu BinSearch, w wariancie pesymistycznym, dla poprawnych danych rozmiaru 10^4 nie jest zależny od uporządkowania danych.

10. Który z poniższych ciągów jest poprawnym rezultatem wykonania procedury Split dla danych wejściowych

$$5, 4, 7, 3, 9, 8, 2,$$

   (a) [–] 5,4,3,2,7,8,9,

   (b) [+] 3,4,2,5,9,8,7,

   (c) [–] 3,2,4,5,9,8,7.

11. Rozważmy następujący algorytm wielokrotnego sortowania danych wejściowych rozmiaru n

```
int Sortuj(int A[], int n) { // wp:  n ≥ 1
   QuickSort(A,n); // algorytm w implementacji rekurencyjnej
      //z procedurą podziału Split
   InsertionSort(A,n);
   MergeSort(A,n); // algorytm w implementacji rekurencyjnej
   SelectionSort(A,n);
}
```

   wtedy:

   (a) [+] jeżeli operacją dominującą jest czynność porównania elementów tablicy A, to $W(Sortuj, n) = O(n^2)$,

   (b) [–] jeżeli operacją dominującą jest czynność porównania elementów tablicy A, to $A(Sortuj, n) = O(n \lg n)$,

   (c) [–] $S(Sortuj, n) = O(\lg n)$.

12. Niech T będzie drzewem decyzyjnym dowolnego (czyli każdego możliwego) algorytmu sortowania przez porównania danych rozmiaru n, wtedy:

   (a) [–] liczba liści w drzewie T daje się oszacować przez $O(n^2 \lg n)$,

   (b) [–] wysokość drzewa T daje się oszacować przez $O(n)$,

   (c) [+] długość najdłuższej ścieżki korzeń-liść w drzewie T daje się oszacować przez $\Omega(n \lg n)$.

13. Rozważmy algorytm CountingSort dla danych wejściowych

$$0, 1, 0, 2, 0, 2, 4, 1, 1, 0, 2,$$

   wtedy postać tablicy pomocniczej (tablica TMP) po:

   (a) [+] drugiej części algorytmu (zliczanie) jest następująca: $4, 3, 3, 0, 1$,

   (b) [+] trzeciej części algorytmu (sumowanie) jest następująca: $4, 7, 10, 10, 11$,

   (c) [–] czwartej części algorytmu (wypisanie) jest następująca: $0, 4, 6, 10, 10$.

14. Rozważmy algorytm obliczania wartości 6-cio operatorowego wyrażenia arytmetycznego przy użyciu dwóch stosów (stos operatorów oraz stos argumentów) dla wyrażenia wejściowego

$$((((1+((2+3)\cdot 4))+5)+6)+7)$$

wtedy:

   (a) [+] wyrażenie wejściowe jest poprawnie i w pełni nawiasowane,
   (b) [–] koszt czasowy algorytmu, mierzony liczbą operacji stosowych, jest nie większy niż 15,
   (c) [–] koszt pamięciowy algorytmu, mierzony liczbą elementów zapisanych na obu stosach, wynosi co najwyżej 3.

15. Które z poniższych zdań jest zawsze prawdziwe w dziedzinie kolejek:

   (a) [–] $\neg empty\,(q) \Rightarrow \neg empty\,(out\,(out\,(in\,(q, first\,(q)))))$,
   (b) [+] $empty\,(q) \Rightarrow in\,(out\,(in\,(q,e))\,,f) = out\,(in\,(in\,(q,e)\,,f))$,
   (c) [+] $empty\,(q) \Rightarrow \neg empty\,(in\,(q,e))$.

16. Które z poniższych zdań jest zawsze prawdziwe w dziedzinie słowników:

   (a) [–] $member\,(delete\,(insert\,(d,e)\,,e)\,,e)$,
   (b) [+] $e \neq f \Rightarrow delete\,(insert\,(d,e)\,,f) \neq delete\,(insert\,(d,f)\,,e)$,
   (c) [+] $\neg member\,(delete\,(delete\,(insert\,(d,e)\,,e)\,,e)\,,e)$.

17. Niech (A,h) będzie tablicą haszującą długości m z kolejkami, będącą implementacją słownika d dla uniwersum elementów E, gdzie $|E| = n$. Jeżeli koszt wyznaczenia wartości funkcji haszującej h, dla dowolnego elementu $e \in E$ jest stały, to:

   (a) [+] $W\,(member\,(d,e)\,,m,n) = O\,(n)$,
   (b) [+] $A\,(insert\,(d,e)\,,m,n) = O\left(\frac{n}{m}\right)$,
   (c) [+] $W\,(delete\,(d,e)\,,m,n) = O\,(m+n)$.

18. Niech T będzie drzewem AVL powstałym przez kolejne wstawianie wierzchołków o etykietach $5, 3, 4, 7, 6$ do początkowo pustej struktury, wtedy:

   (a) [–] korzeniem drzewa T jest wierzchołek o etykiecie 5,
   (b) [+] rezultatem działania algorytmu PreOrder dla drzewa T jest ciąg etykiet $4, 3, 6, 5, 7$,
   (c) [–] usunięcie wierzchołka o etykiecie 4 z drzewa T wymaga wykonania co najmniej jednej podwójnej rotacji w celu przywrócenia własności drzewa AVL.

19. Niech drzewo binarne T będzie implementacją n-elementowego słownika d, wtedy złożoność czasowa operacji:

   (a) [–] *member* dla słownika d jest $O\,(\lg n)$, jeżeli T jest drzewem BST,
   (b) [–] *insert* dla słownika d jest $\Omega\,(\lg n)$, jeżeli T jest drzewem BST,
   (c) [+] *delete* dla słownika d jest $\Omega\,(\lg n)$, jeżeli T jest drzewem AVL.

20. Które z poniższych zdań jest zawsze prawdziwe w dziedzinie kolejek priorytetowych:

   (a) [–] $e \neq min(pq) \Rightarrow delmin\,(insert\,(pq,e)) = insert\,(delmin\,(insert\,(pq,e))\,,e)$,
   (b) [–] $delmin\,(insert\,(pq,e)) = pq \Rightarrow \neg empty(pq)$,
   (c) [+] $e = min\,(pq) \Rightarrow min\,(insert\,(pq,e)) = e$.

21. Rozważmy kopiec binarny-drzewo H powstały przez kolejne wstawianie liczb $5, 4, 3, 2, 1, 6, 7$ do początkowo pustej struktury, wtedy:

    (a) [+] etykiety drzewa odczytane w porządku PostOrder tworzą ciąg $5, 3, 2, 6, 7, 4, 1$,

    (b) [–] jeżeli wykonamy ciąg operacji $delmin\,(H)$, $delmin\,(H)$, to etykiety drzewa odczytane w porządku InOrder tworzą ciąg $3, 5, 6, 7, 4$,

    (c) [–] koszt operacji *delmin* na strukturze H jest rzędu liniowego względem liczby wierzchołków liści drzewa.

22. Niech H będzie n-elementowym kopcem binarnym zaimplementowanym w drzewie binarnym T, wtedy:

    (a) [–] wysokość drzewa T jest rzędu $\lg(\sqrt{n})$,

    (b) [+] drzewo T ma co najwyżej 500 wierzchołków wewnętrznych jeśli $n = 1000$,

    (c) [+] liczba wierzchołków liści na przedostatnim poziomie drzewa T jest równa co najwyżej $2^{\lg n - 1}$.

23. Rozważmy ciąg liczb $\alpha = 3, 1, 4, 2, 5$, wtedy:

    (a) [+] tablica reprezentująca kopiec binarny utworzony z elementów ciągu α przez kolejne wstawianie elementów do początkowo pustego kopca ma postać $[1, 2, 4, 3, 5]$,

    (b) [–] tablica reprezentująca kopiec binarny utworzony z elementów ciągu α przez zastosowanie szybkiej procedury budowy kopca (tj. HeapConstruct) ma postać $[1, 2, 3, 5, 4]$,

    (c) [+] tablica reprezentująca kopiec binarny utworzony z elementów ciągu α przez zastosowanie szybkiej procedury budowy kopca (tj. HeapConstruct) ma postać $[1, 2, 4, 3, 5]$.

24. Rozważmy zbiór $E = \{a, b, c, d, e\}$ oraz następujący ciąg operacji na strukturze Find-Union U zaimplementowanej z użyciem list z balansowaniem:

    $$\begin{aligned}&init\,(E),\\ &find\,(U, a),\\ &union\,(U, find\,(U, d)\,, find\,(U, e)),\\ &find\,(U, e),\\ &union\,(U, find\,(U, e)\,, find\,(U, a)),\\ &union\,(U, find\,(U, b)\,, find\,(U, c)),\end{aligned}$$

    wtedy:

    (a) [+] $find\,(U, e) = find\,(U, d)$,

    (b) [+] $find\,(union\,(U, find\,(U, a)\,, find\,(U, b))\,, c) = find\,(U, e)$,

    (c) [–] $find\,(U, c) = find\,(U, d)$?

25. Rozważmy graf-drzewo $G = (V, E)$, gdzie $V = \{1, 2, 3, 4, 5, 6\}$, w którym wszystkie krawędzie mają wagę 20. Wtedy:

    (a) [+] drzewo najkrótszych ścieżek z wierzchołka początkowego 1 grafu G wyznaczone przez algorytm Dijkstry składa się z 5-ciu krawędzi i ma łączny koszt (mierzony wagami krawędzi) równy 100,

    (b) [–] drzewo najkrótszych ścieżek z wierzchołka początkowego 3 grafu G wyznaczone przez algorytm Dijkstry składa się z 4-ech krawędzi i ma łączny koszt (mierzony wagami krawędzi) równy 80,

    (c) [+] koszt algorytmu Dijkstry zastosowanego do rozważanego grafu jest rzędu $O\left(n^2\right)$, gdzie n jest liczbą wierzchołków grafu G.

26. Rozważmy graf spójny G składający się z 10-ciu wierzchołków, którego wszystkie krawędzie mają wagi równe 5. Wtedy:

    (a) [–] koszt algorytmu Kruskala zastosowanego do rozważanego grafu jest rzędu liniowego względem ilości krawędzi grafu G,

    (b) [+] minimalne drzewo rozpinające grafu G wyznaczone przez algorytm Kruskala składa się z 9-ciu krawędzi i ma łączny koszt (mierzony wagami krawędzi) równy 45,

    (c) [–] istnieje drzewo rozpinające grafu G mające 10 krawędzi.

27. Który z poniższych kodów jest optymalnym (w sensie długości) kodem prefiksowym dla alfabetu $\Sigma = \{a,b,c,d,e\}$ i tekstu składającego się z czterech liter "a", dwóch liter "b", trzech liter "c", sześciu liter "d" oraz siedmiu liter "e":

    (a) [–] a – 10, b – 011, c – 100, d – 11, e – 0,

    (b) [+] a – 10, b – 011, c – 010, d – 11, e - 00,

    (c) [+] a – 00, b – 010, c – 011, d – 10, e – 11?

28. Który z podanych poniżej kodów może być kodem Huffmana dla tekstu składającego się odpowiednio z pięciu liter A, siedmiu liter B, sześciu liter C, dziesięciu liter D oraz ośmiu liter E:

    (a) [–] A – 110, B – 00, C – 111, D – 11, E – 01,

    (b) [+] A – 110, B – 00, C – 111, D – 10, E – 01,

    (c) [–] A – 110, B – 00, C – 111, D – 101, E – 01?

29. Które z poniższych zdań jest prawdziwe:

    (a) [–] otoczka wypukła n-kąta wypukłego składa się z co najwyżej $\sqrt{n}$ punktów,

    (b) [–] złożoność algorytmu Jarvisa można oszacować przez $\Omega(n \lg n)$,

    (c) [+] algorytm Grahama jest co najwyżej tak trudny obliczeniowo jak algorytm Jarvisa w przypadku pesymistycznym.

30. Które z podanych zdań jest prawdziwe:

    (a) [+] każdy problem z klasy $\mathbb{P}$ należy do klasy $\mathbb{NP}$,

    (b) [–] problem komiwojażera jest tak samo trudny obliczeniowo jak problem wyszukiwania w danych uporządkowanych,

    (c) [–] problem stopu należy do klasy $\mathbb{NP}$.

TABELKA ODPOWIEDZI

1	2	3	4	5	6	7	8	9	10

11	12	13	14	15	16	17	18	19	20

21	22	23	24	25	26	27	28	29	30