Algorytmy i Struktury Danych

1. (2pkt) Podaj nierekurencyjną wersję procedury *Quicksort*, która
   - poza tablicą z danymi ($A[1..n]$ of integer) używa tylko stałej (niezależnej od n) liczby komórek typu integer (zakładamy, że $\max(n, \max\{A[i] \mid i = 1, .., n\})$ jest największą liczbą jaką może pomieścić taka komórka),
   - czas jej działania jest co najwyżej o stały czynnik gorszy od czasu działania wersji rekurencyjnej.
2. (2pkt) Niech $A = a_1, a_2, \ldots, a_n$ będzie ciągiem elementów oraz niech p i q będą dodatnimi liczbami naturalnymi. Rozważmy p-podciągi ciągu A, tj. podciągi utworzone przez wybranie co p-tego elementu. Posortujmy osobno każdy z tych podciągów. Powtórzmy to postępowanie dla wszystkich $q-$podciągów. Udowodnij, że po tym wszystkie $p-$podciągi pozostaną posortowane.
3. (1pkt) Rozważmy modyfikację podanego na wykładzie algorytmu sprawdzającego izomorfizm drzew, który porządkując wektory przypisane wierzchołkom stosuje sortowanie leksykograficzne ciągów jednakowej długości (po uprzednim wyrównaniu długości wektorów - przez dopisanie symbolu spoza alfabetu). Podaj przykład "złośliwych" danych dla takiego algorytmu. Oszacuj czas działania algorytmu na tych danych.
4. (1pkt) Opracuj wersję algorytmu *Quicksort*, która będzie efektywnie działać na ciągach zawierających wielokrotne powtórzenia kluczy.
5. (2pkt) Opracuj wersję algorytmu *Mergesort*, która działa w miejscu.

Zadania dla grupy zaawansowanej

1. (2pkt) *Serią* w ciągu nazwiemy dowolny niemalejący podciąg kolejnych jego elementów. Seria jest *maksymalna*, jeśli nie można jej rozszerzyć o kolejne elementy. Załóżmy, że algorytm *InsertSort* uruchamiany będzie jedynie na permutacjach zbioru $\{1, 2, \ldots, n\}$, które można rozbić na co najwyżej dwie serie maksymalne. Zbadaj średnią złożoność algorytmu przy założeniu, że dla każdego n, wszystkie takie permutacje n-elementowe są jednakowo prawdopodobne.
2. (2pkt) Rozważmy permutacje liczb $\{1, 2, \ldots, n\}$, których wszystkie 2-podciągi i 3-podciągi są uporządkowane.
   (a) Ile jest takich permutacji?
   (b) Jaka jest maksymalna liczba inwersji w takiej permutacji?
   (c) Jaka jest łączna liczba inwersji w takich permutacjach?

Zadania nadobowiązkowe - nie będą rozwiązywane na ćwiczeniach

1. (0pkt) Pokaż, że *Quicksort* działa w czasie $\Theta(n \log n)$, gdy wszystkie elementy tablicy A mają tę samą wartość.
2. (0pkt) Pokaż, że *Quicksort* działa w czasie $\Theta(n^2)$, gdy tablica A jest uporządkowana niemalejąco.
3. (0pkt) Załóżmy, że na każdym poziomie rekursji procedury *Quicksort* procedura *partition* dzieli daną tablicę na dwie podtablice w proporcji $1 - \alpha$ do α, gdzie $0 < \alpha \leq \frac{1}{2}$ jest stałą. Pokaż, że minimalna głębokość liścia w drzewie rekursji wynosi około $-\frac{\log n}{\log \alpha}$ a maksymalna głębokość liścia wynosi około $-\frac{\log n}{\log(1-\alpha)}$.

4. (2pkt) n-elementowym ciągiem o jednym zaburzeniu nazywamy dowolny ciąg, który może być otrzymany z ciągu $\{1, 2, \ldots, n\}$ poprzez wykonanie jednej transpozycji. Załóżmy, że algorytm *InsertSort* będzie uruchamiany jedynie na ciągach o jednym zaburzeniu. Zbadaj średnią złożoność algorytmu przy założeniu, że dla każdego n, wszystkie takie ciągi n-elementowe są jednakowo prawdopodobne.

5. (1pkt) (Poprawność procedury *Partition*). Rozważ następującą procedurę:

```
Partition(A, p, r)
x ← A[p]
i ← p − 1
j ← r + 1
while true do
    repeat j − −
        until A[j] ≤ x
    repeat i + +
        until A[i] ≥ x
    if i < j
        then zamień A[i] ↔ A[j]
        else return j
```

Udowodnij co następuje

(a) Indeksy i oraz j nigdy nie wskazują na element A poza przedziałem $[p..r]$.

(b) Po zakończeniu *Partition* indeks j nie jest równy r (tak więc podział jest nietrywialny).

(c) Po zakończeniu *Partition* każdy element $A[p..j]$ jest mniejszy lub równy od dowolnego elementu $A[j + 1, r]$.

(1pkt) Ułóż algorytm sortujący w miejscu ciągi rekordów o kluczach ze zbioru $\{1, 2, 3\}$.

(1pkt) Ułóż algorytm sortujący ciąg n liczb całkowitych w czasie $O(n)$ i pamiêci $O(n)$. Przyjmij, że liczby są z zakresu **long long.**