# Egzamin kwalifikacyjny

na uzupełniające studia magisterskie

6 września 2006 r.

1. [**Programowanie**]

   (a) **80 % punktów** Zaprogramuj funkcję o nagłówku

   ```
   static void sort(int *from, int *to);
   ```

   implementującą rekurencyjny algorytm sortowania *Quicksort* liczb typu `int`. Argumenty `from` i `to` są wskaźnikami na, odpowiednio, pierwszy i ostatni element sortowanej tablicy. Jest to funkcja pomocnicza dla następującej implementacji funkcji sortującej:

   ```
   void qsort(int *array, int size) {
      if (size>1)
         sort(array, array+size-1);
   }
   ```

   Przypomnijmy, że algorytm `Quicksort` polega na przestawieniu elementów tablicy w taki sposób, by w jej pierwszej części znalazły się jedynie elementy mniejsze, zaś w drugiej — elementy nie mniejsze od pewnej wartości, zwanej medianą, a następnie na rekurencyjnym wywołaniu procedury, osobno dla pierwszej i drugiej części tablicy. Przyjmij, że medianą jest wartość pierwszego elementu tablicy, tj. wartość `*from`. Funkcja `sort` ma być wywoływana jedynie poprzez funkcję `qsort`, która gwarantuje, że `from` $<$ `to`. Działanie funkcji `sort` dla argumentów nie spełniających tego warunku może być dowolne (może się w szczególności zapętlać).

   (b) **20 % punktów** Wyjaśnij, co oznacza słowo `static` w nagłówku funkcji `sort` i dlaczego zostało w tym nagłówku użyte.

2. [**Algorytmy i struktury danych**] Dany jest zbiór elementów $A = \{a_1, a_2, \ldots, a_n\}$. Elementy te możemy porównywać (mogą to być liczby, słowa, daty lub rekordy z kluczem, według którego możemy elementy uporządkować). Zależy nam na posortowaniu ich w porządku niemalejącym.

   (a) **30 % punktów** Opisz zwięźle w pseudokodzie trzy algorytmy sortowania: sortowanie szybkie (QuickSort), sortowanie przez scalanie (MergeSort) oraz sortowanie kopcowe (HeapSort). Dla każdego wylicz złożoność czasową i pamięciową.

   (b) **20 % punktów** Odpowiedz, w jakich sytuacjach jest najbardziej właściwe zastosowanie każdego z tych algorytmów.

   (c) **30 % punktów** Wybierz jeden z algorytmów i opisz jego możliwe modyfikacje, ich zalety i rozwiązania implementacyjne.

   (d) **20 % punktów** Uzasadnij, że dla algorytmów opartych na porównaniach dolna granica na czas sortowania $n$ elementów wynosi $\Omega(n \log n)$.

3. [**Matematyka dyskretna**] Niech $G = (V, E)$ będzie digrafem, czyli grafem skierowanym.

   (a) **10 % punktów** Podaj definicję digrafu $G^*$, który jest przechodnim domknięciem digrafu $G$.

   (b) **30 % punktów** Podaj algorytm Warshalla, służący do wyznaczania digrafu $G^*$ na podstawie macierzy sąsiedztwa digrafu $G$, i określ jego złożoność. Algorytm zapisz w wybranym pseudojęzyku programowania.

   (c) **10 % punktów** Zastosuj schemat algorytmu Warshalla, by otrzymać algorytm służący do wyznaczania długości najkrótszych dróg między każdą parą wierzchołków. Ten algorytm zapisz w wybranym pseudojęzyku programowania.

   (d) **20 % punktów** Zapisz w wybranym pseudojęzyku programowania algorytm służący do wyznaczania digrafu $G^*$, który jest iteracją algorytmu przeszukiwania digrafu metodą w głąb albo wszerz. Digraf jest pamiętany w postaci list sąsiadów. Określ złożoność podanego algorytmu.

   (e) **30 % punktów** Zapisz w pseudojęzyku programowania algorytm służący do wyznaczania przechodniego domknięcia digrafu symetrycznego. Twój algorytm powinien mieć złożoność $O(n^2)$, gdzie $n$ jest liczbą wierzchołków digrafu.