# Algorytmy i Struktury Danych
# egzamin na studia uzupełniające
# luty 2013 r.

**Zadanie 1 (36 punktów)**

Jakie jest dolne ograniczenie na liczbę porównań w *problemie sortowania*? Zakładamy, że sortujemy tylko w oparciu o porównywanie kluczy.

1. Oszacuj asymptotycznie od dołu liczbę porównań jaką musi wykonać w najgorszym przypadku algorytm sortujący dane rozmiaru $n$; wykorzystaj do tego celu drzewa decyzyjne.
2. W oparciu o przedstawioną granicę dolną wylicz, ile porównań musi wykonać w najgorszym przypadku algorytm sortujący 5 elementów; następnie skonstruuj taki algorytm (sortujący optymalnie) i zapisz go w pseudokodzie.
3. Wybierz i opisz jeden asymptotycznie optymalny co do liczby porównań algorytm sortujący; opisz krótko ideę jego działania; przeanalizuj jego złożoność czasową.

**Zadanie 2 (32 punkty)**

Dana jest lista $n$ wykładów, które mają się odbywać w tej samej sali. Każdy z wykładowców upiera się przy swoim terminie przeprowadzenia zajęć: $[p_1, k_1), [p_2, k_2), \ldots, [p_n, k_n)$. Nie można wszystkich usatysfakcjonować, gdyż dysponujemy tylko jedną salą wykładową.

1. Podaj algorytm, który wybierze wykłady w taki sposób, aby odbyło się ich jak najwięcej.
2. Koniecznie uzasadnij poprawność Twojego rozwiązania (dowód nie wprost).
3. Czy algorytm ten będzie skuteczny, gdybyśmy chcieli zmaksymalizować czas wykorzystania sali?

**Zadanie 3 (32 punkty)**

Zaprojektuj strukturę danych umożliwiającą efektywne wykonywanie (najlepiej w czasie logarytmicznym względem bieżącej długości ciągu) następujących operacji na początkowo pustym ciągu liczbowym $S$:

- *S.insert(x,j)* : wstawienie liczby $x$ na $j$–tą pozycję do ciągu $S$;
- *S.sum(j)* : obliczenie sumy $j$ początkowych elementów w ciągu $S$ ($\sum_{i=0}^{j-1} s_i$).

W powyższych operacjach parametr $j$ spełnia warunek $0 \leq j \leq |S|$.

1. Opisz ideę rozwiązania tego problemu. Zaprojektuj strukturę umożliwiającą efektywne wykonywanie na niej wymienionych operacji, wykorzystując jakąś znaną strukturę danych realizującą efektywnie operację *insert*.
2. Opisz dokładnie modyfikacje wprowadzone do tej struktury i do procedury *insert*; uzasadnij, że nie zmieni to poprawnego i efektywnego działania tej procedury.

3. Procedurę *sum* zapisz w pseudokodzie, opisz jej działanie i przeanalizuj złożoność obliczeniową (czas i pamięć).

*Uwaga.* Ciąg $S$ jest indeksowany od zera, czyli $S = (s_0, s_1, \ldots, s_{n-1})$, dla $|S| = n$.

*Przykład.* Przy wstawianiu nowej liczby na $j$–tą pozycję, wszystkie liczby od pozycji $j$–tej włącznie są przesuwane o jedną pozycję dalej. Niech $S = (a, b, c)$ będzie 3–elementowym ciągiem liczbowym. Wówczas:

- wykonując operację *S.insert(x,1)*, czyli wstawiając liczbę $x$ na pozycję 1, otrzymamy ciąg $(a, x, b, c)$;
- wykonując operację *S.insert(y,0)*, wstawimy $y$ na początek ciągu otrzymując $(y, a, b, c)$;
- a wykonując operację *S.insert(z,3)*, wstawimy $z$ na koniec ciągu otrzymując $(a, b, c, z)$.