ćwiczenia (licencjat wieczorowy): lista zadań nr 3 *23 października 2006 r.*

ALGORYTMY I STRUKTURY DANYCH

SORTOWANIE I WYBÓR

Instytut Informatyki Uniwersytetu Wrocawskiego *Paweł Rzechonek*

1. [$*$] Czasami mamy do czynienia z danymi, których wartości często się powtarzają. Jak zmodyfikować procedurę podziału, aby dokonywała ona tak zwanego *trójpodziału* względem wybranego elementu x: na początku ma się znaleźć blok z elementami $< x$, potem blok z elementami $= x$, a na końcu blok z elementami $> x$. Czas działania twojego algorytmu powinien być liniowy.

2. [$*$] Jak zaimplementować algorytm *quick–sort*, aby w pesymistycznym przypadku działał on w czasie $O(n \log n)$ na danych rozmiaru n?

3. [$***$] Jak zmodyfikować algorytm *quick–sort*, aby w każdym przypadku głębokość wywołań rekurencyjnych była nie większa niż $\log n$ dla danych rozmiaru n?

4. [$**$] Mamy dostęp do algorytmu *czarna skrzynka*, który w liniowym czasie znajduje medianę nieuporządkowanego ciągu liczb. W jaki sposób, korzystając z *czarnej skrzynki*, wyznaczyć dowolny k-ty co do wielkości element w nieuporządkowanym ciągu liczb. Twój algorytm także powinien działać w czasie liniowym.

5. [$**$] Przedstawiony poniżej algorytm dokonuje jednoczesnego wyboru minimum i maksimum spośród nieuporządkowanych elementów. Określ ile porównań będzie wykonywał ten algorytm dla danych rozmiaru n. Jak bardzo różni się ten wynik od dolnej granicy $\lceil \frac{3}{2} n - 2 \rceil$? Jak zachowuje się ten algorytm dla danych, których rozmiar jest potęgą 2?

```
function MINMAX–REC (key T[0 ... n − 1]) ↦ (key, key)
{
    if n = 1 then return (T[0], T[0]);
    if n = 2 then
        if T[0] ≤ T[1] then return (T[0], T[1]); else return (T[1], T[0]);
    m ← ⌊n/2⌋;
    (m_l, M_l) ← MINMAX–REC (T[0 ... m − 1]);
    (m_r, M_r) ← MINMAX–REC (T[m ... n − 1]);
    return (min{m_l, m_r}, max{M_l, M_r});
}
```