Név: ..... *Neptun kód:* .....

Algoritmusok és adatszerkezetek II. vizsga, 2016. 05. 18.

1. Mutassuk be a számjegypozíciós („Radix") rendezés működését a $< 11; 20; 10; 23; 21; 30 >$ négyes számrendszerbeli számok tömbjén! Az egyes menetekben leszámláló rendezést alkalmazzunk! Mekkora a Radix rendező algoritmus műveletigénye? A leszámláló rendezés – mint segédprogram – mely tulajdonságaira épül a Radix rendezés? (20p)

2. Adott egy $m = 11$ méretű üres hasító tábla. Nyílt címzést és kettős hasítást alkalmazunk a „$h_1(k) = k \bmod m$" és a „$h_2(k) = 1+(k \bmod (m-1))$" tördelő függvények segítségével. Az alábbi műveletek mindegyikére adjuk meg a próbasorozatot és az eredményül adódó hasító táblát is!

Szúrjuk be a táblába sorban a következő kulcsokat: 10; 22; 31; 4; 15; 28; 17; 88; 59. Ezután töröljük a 17-et, majd próbáljuk megkeresni a 18-at és az 59-et, végül pedig szúrjuk be a 18-at!

Magyarázzuk meg, mi a különbség nyílt címzés esetén a *foglalt*, az *üres* és a *törölt* rések között, az alkalmazható műveletek szempontjából! Mi a kitöltöttségi hányados? Milyen becslést tudunk adni az egyes műveletigényekre, és milyen feltétellel? (20p)

3. Az $\{A, B\}$ ábécével szemléltessük a Lempel–Ziv–Welch (LZW) tömörítő algoritmus működését az $ABABABABA$ szövegen, majd a megfelelő kitömörítő algoritmusét az $1, 2, 2, 3, 6, 4, 7$ kódon! Mindkét esetben adjuk meg a generált szótárat, és a tömörítetlen szövegen a részszavak és a kódok megfeleltetését!

Milyen értelemben optimális a Huffman kód? Hogyan lehetséges, hogy az LZW algoritmus a gyakorlatban gyorsabban és jobban tömörít? (20p)

4. Mit számol ki a Prim algoritmus? Definiálja a súlyozott szomszédossági csúcsmátrix fogalmát! Csak *tömb* adatszerkezeteket használva adja meg a $\mathrm{Prim}(C[1..n, 1..n], d[1..n], \pi[1..n])$ eljárás struktogramját, ahol a gráfot a C súlyozott szomszédossági csúcsmátrix segítségével ábrázoltuk. A segédeljárásokat is részletezze! Az eredményt, a csúcsok d és a π értékeit a megfelelő vektorokban kapjuk. $T(n) \in O(n^2)$, $M(n) \in O(n)$ (20p)

5. Mit számol ki a *Sor alapú (Queue-based) Bellman-Ford* algoritmus? Adja meg a struktogramját! Mit értünk a fenti program futásának *menetei* alatt? Mi a menetekhez kapcsolódó alapvető tulajdonság? Adjon az algoritmus futási idejére aszimptotikus *felső* becslést, és indokolja is állítását! Honnét ismerhető fel, hogy van-e a gráfban a startcsúcsból elérhető negatív kör? Hogyan lokalizálható egy ilyen negatív kör? (20p)