Institut for Matematik og Datalogi
Syddansk Universitet, Odense

12. marts 2014
RF

DM507 – Opgaver uge 12

Eksaminatorier

1. Eksamen juni 2008, opgave 4 b.
2. Cormen et al. øvelse 12.2-3 (side 293).
3. Cormen et al. øvelse 2.1-3 (side 22). Kun spørgsmålet om invariant, resten er lavet i uge 7.
4. Cormen et al. øvelse 2.2-2 (side 29).
5. Cormen et al. opgave 2-2 (side 40).
6. Eksamen juni 2010, opgave 3.
7. Cormen et al. opgave 2.3 (side 41). Hint: for at overskue situationen, prøv algoritmen på konkrete instanser, og skriv summen i spørgsmål **c** ud.
8. Implementer Countingsort i Java ud fra bogens pseudokode (side 171). Husk at sætte en øvre grænse k på de `int`'s, du sorterer. Test at din kode fungerer ved at generere arrays med forskelligt indhold og sortere dem. Tilføj tidtagning af din kode (kun selve sorteringen, ikke den del af programmet som genererer array'ets indhold).

   Kør derefter din kode med input, som er random `int`'s i intervallet $[0; k]$ for $k = n/20$, n, og $20n$ (dvs. tre kørsler for hver værdi af n). Brug f.eks. `java.util.Random.nextInt(k+1)` til generering af tallene. Gør dette for mindst tre forskellige værdier af n (antal elementer at sortere), vælg værdier som får programmet til at bruge fra ca. 100 til ca. 5000 millisekunder for $k = n$ kørslen. Gentag hver enkelt kørsel tre gange og find gennemsnittet af antal millisekunder brugt ved de tre kørsler. Dividér de fremkomne tal med $n + k$ og check derved hvor godt analysen passer med praksis – de resulterende tal burde ifølge analysen være konstante.

Sammenlign med dine køretider for det tilsvarende forsøg (samme n) med Quicksort fra opgaverne i uge 8. Er Quicksort eller Countingsort hurtigst? Afhænger det af k (for fastholdt n)?

Studiegrupper

Forslag til fokus for arbejde i studiegrupper: Forbered dele af opgaverne til eksaminatorietimer, f.eks. på nedenstående måde.

- Forsøg at lave de mere simple af opgaverne (f.eks. opgave 1–2) individuelt især på forhånd, og sammenlign svar i studiegruppen.
- Forsøg at løse de mere udfordrende af opgaverne i fælleskab (opgave 3–7). Arbejd med at forstå 1) hvordan invarianten er sand før første løkke gennemløb, 2) hvordan invarianten bliver vedligeholdt sand af løkkens handlinger under hver ny iteration af løkken, og 3) hvordan den betingelse, som fik løkken til at stoppe, sammen med invarianten beviser det ønskede om algoritmen ved dens afslutning.