

Opgaver Uge 10

DM507/DS814/T510040101

I.A: Løses i løbet af de første øvelsestimer i uge 10

1. Eksamen juni 2008, opgave 4 b.
2. Cormen et al. øvelse 12.2-1 (side 293), spørgsmål a–c.
3. Cormen et al. øvelse 12.2-3 (side 293).
4. (*) Cormen et al. øvelse 12.3-3 (side 299). Bemærk at opgaven mener ubalancerede søgetræer (kapitel 12). Besvar bagefter opgaven igen, men nu med rød-sorter træer i stedet for ubalancerede binære søgetræer.
5. Cormen et al. øvelse 12.1-5 (side 289)
6. Cormen et al. øvelse 13.1-2 (side 311).

I.B: Løses hjemme inden de næste øvelsestimer i uge 10

1. Cormen et al. øvelse 12.1-2 (side 289)
2. (*) Cormen et al. øvelse 14.2-4 (side 348). Opgaven kan løses uden at læse kapitel 14. Operationen kaldes langt oftere `RANGESEARCH` end `ENUMERATE`. Hint: lade dig inspirere af `INORDER-TREE-WALK` (side 288). Dette giver ret nemt algoritmen, og det udfordrende i opgaven er så at finde et argument for køretiden.

II.A: Løses i løbet af de næste øvelsestimer i uge 10

1. Eksamen jan 2005, opgave 1.
2. Cormen et al. øvelse 13.1-6 (side 312).
3. Eksamen juni 2013, opgave 5.
4. Eventuelle opgaver fra tidligere ugesedler, som mangler at gennemgås.

II.B: Løses hjemme inden øvelsestimerne i uge 11

1. Eksamen juni 2011, opgave 1.
2. Implementer Countingsort i Java eller Python ud fra bogens pseudokode (side 195). Husk at sætte en øvre grænse k på de `int`'s, som du genererer som input. 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 $k = n/50$, n , og $50n$ (dvs. tre værdier af k for hver værdi af n). Brug f.eks. `java.util.Random.nextInt(k+1)` i Java og `random.randint(0,k)` i Python 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. Divider 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 (eller evt. Mergesort fra uge 7). Er Quicksort eller Countingsort hurtigst? Afhænger det af k (for fastholdt n)?