TÖL203G Tölvunarfræði 2

Heimadæmi 3

Í síðustu viku var lokið við efni kafla 1.4 um greiningu reiknirita og farið í kaflu 1.5 um *Union-find* verkefnið. Í næstu viku verður byrjað á kafla 2 um röðun.

Heimadæmin eru **til þess að þjálfa ykkur** í efninu − nýtið þau vel! Einkunn fyrir þau mun **ekki lækka lokaeinkunn**, þannig að þið fáið mun meira út úr því að glíma við dæmin sjálf en að fá aðstoð frá Hr. Google.

Æfingadæmi fyrir dæmatíma 30. og 31. jan.

1. Leysa á eftirfarandi verkefni: Gefið er N-staka heiltölufylki A og það á að búa til tvívíða fylkið B, sem er þannig að stak (i, j) í B inniheldur summu stakanna $A[i]$, $A[i+1]$, ..., $A[j]$ fyrir öll $i \le j$. Gildi stakanna í öðrum sætum B eru óskilgreind (þ.e. fyrir $i > j$). Hér fyrir neðan er einfalt reiknirit sem leysir þetta verkefni:

```
Fyrir i = 0, 1, ..., N−1
  Fyrir j = i, i+1, ..., N−1
    Reikna A[i]+A[i+1]+...+A[j] og setja í B[i, j]
```

   Útfærið reikniritið að ofan í Java og takið tíma þess á slembnu inntaksfylki (sem forritið býr til sjálft). Metið tvöföldunarhlutfall keyrslutímans með því að keyra forritið fyrir 3 - 4 gildi á N og tvöfalda gildið í hvert sinn. Til að koma ykkur af stað er hér beinagrind að lausn: Hlutsummur.java.

Heimadæmi (skila í Gradescope)

1. [*Reiknirit*] Breyta FourSum.java í `FourSumFast.java` á sama hátt og gert er með ThreeSumFast.java. Skilið kóða fallsins `count` (sem texta, ekki skjáskoti) og skjáskoti af keyrslu `FourSum` og `FourSumFast` á gagnaskránni 1Kints.txt. Þá eiga að finnast 13654 ferndir. Athugið að þið þurfið að aðlaga kóðann aðeins, því `FourSum.java` notar `long` fylki í stað `int` fylkis og innlestur gagnanna er aðeins ólíkur.

2. [*Reiknirit*] Framhald af æfingadæminu að ofan.

   a. Finnið raunhæf neðri mörk á vaxtarhraða keyrslutíma reiknirits sem leysir þetta verkefni, þ.e. hversu margar aðgerðir þurfa öll reiknirit að nota til þess að leysa þetta verkefni (sem fall af N)? Rökstyðjið svarið í nokkrum orðum.

   b. Það er hægt að leysa verkefnið á mun hraðvirkari hátt en gert var í æfingadæminu, með því að nýta sér fyrri útreikninga í B. Hugmyndin er að þegar þið eruð að reikna út B[i, j] þá eruð þið nýbúin að reikna út $B[i, j\text{-}1]$. Er ekki hægt að nota það gildi?

   Útfærið þetta reiknirit í Java og keyrið það fyrir sömu gildi á N og gert var í æfingadæminu. Hver er vaxtarhraði þessa nýja reiknirits? Skilið kóðanum (sem texta, ekki skjáskoti) og svarinu.

3. [*Reiknirit*] Tiltekið hótel hefur *N* herbergi, sem eru í röð á löngum gangi. Herbergi 0 er næst móttökunni, en herbergi *N*−1 er lengst í burtu. Öll herbergin frá 0 til *F*−1 eru tekin, en herbergi *F* til *N*−1 eru laus. Við viljum sjálf vera í herbergi *F*, en við vitum ekki gildið á *F* (aðeins að $F < N$). Til þess að finna fyrsta lausa herbergið getum við aðeins kannað eitt herbergi í einu með því að banka á hurðina og kíkja inn. Við viljum lágmarka fjölda skipta sem við bönkum á hurðir í versta tilfelli.

   a. Hver er versta tilfellis tími (sem fall af *N*) á reikniriti sem byrjar á herbergi 0 og rekur sig út eftir ganginum þar til fyrsta lausa herbergið er fundið?

   b. Lýsið reikniriti sem notar í versta falli $\log N$ tíma til að finna fyrsta lausa herbergið.

   c. Ef *N* er mikið stærra en *F*, þá er hægt að gera betur og nota aðeins $\sim 2 \log F$ tíma til að finna fyrsta lausa herbergið. Lýsið þessari aðferð og rökstyðjið vaxtarhraða þess.

4. [*Union-find*] Geta eftirfarandi `id[]` fylki komið út þegar reikniritið viktað *Quick-union* án vegþjöppunar er keyrt með *N*=8? Teiknið upp trén í hvoru tilfelli og útskýrið hvers vegna þetta fylki er ómögulegt eða sýnið röð *union*-aðgerða sem enda í þessu fylki.

   a. 0, 1, 0, 3, 1, 1, 1, 3

   b. 5, 0, 6, 6, 5, 6, 6, 4

5. [*Union-find*] Í aðalútfærslunni á *Union-find* í kennslubókinni, UF.java, er notuð vegþjöppun með helmingun (*path halving*). Rekjið ykkur í gegnum kóðann í UF.java fyrir eftirfarandi sameiningar. Fjöldi staka er 6 og aðgerðirnar eru:

   union(0, 1), *union*(2, 3), *union*(4, 5), *union*(1, 3), *union*(3, 5).

   Teiknið trén eftir tvær síðustu aðgerðirnar og bendið á hvar raunveruleg vegþjöppun á sér stað (þ.e. vegur í rótina styttist). Athugið að inni í hverri *union*-aðgerð eru tvær *find*-aðgerðir.

Skilið PDF-skjali með lausnum ykkar á þessum dæmum fyrir **kl. 23:59 fimmtudaginn 2. febrúar** í Gradescope. Munið eftir að gefa upp á hvaða blaðsíðum svör við einstökum dæmum eru.