

Heimadæmi 3 - Tölvunarfræði 2

Magnús Daníel Einarsson

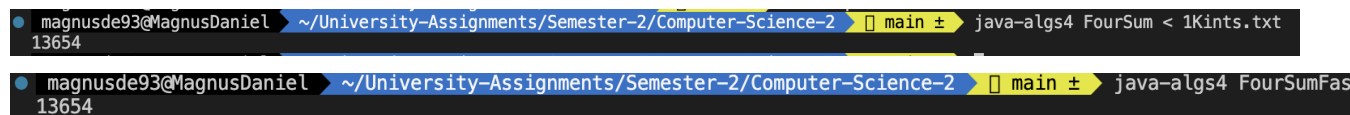
February 2023

Verkefni 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.

Lausn:

```
public static int count(int[] a) {
    int N = a.length;
    Arrays.sort(a);
    int cnt = 0;
    for (int i = 0; i < N; i++) {
        for (int j = i+1; j < N; j++) {
            for (int k = j+1; k < N; k++) {
                int z = Arrays.binarySearch(a, -(a[i] + a[j] + a[k]));
                if (z > k) cnt++;
            }
        }
    }
    return cnt;
}
```



```
magnusde93@MagnusDaniel ~/University-Assignments/Semester-2/Computer-Science-2 [main] java -algs4 FourSum < 1Kints.txt
13654
```

```
magnusde93@MagnusDaniel ~/University-Assignments/Semester-2/Computer-Science-2 [main] java -algs4 FourSumFast
13654
```

Verkefni 2

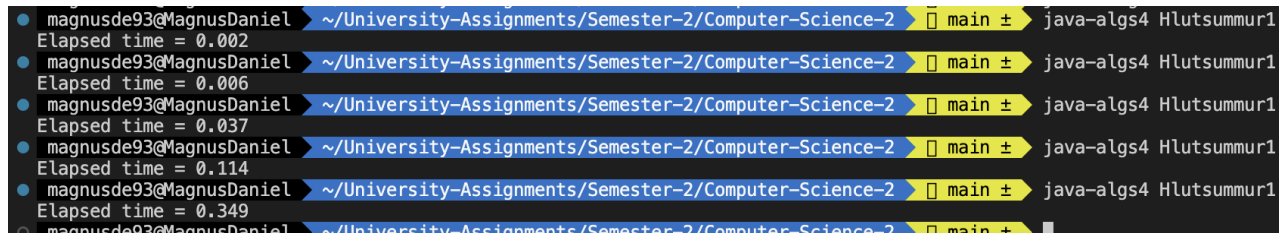
- Finnið raunhæf neðri mörk á vaxtarhraða keyrslutíma reiknirits sem leysir þetta verkefni, þ.e. hversu margar aðgerðir þurfa öll reiknirit að nota til þess að leysa þetta verkefni (sem fall af N)? Rökstyðjið svarið í nokkrum orðum.
- Það er hægt að leysa verkefnið á mun hraðvirkari hátt en gert var 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 $B[i, j-1]$. 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.

Lausn:

- a) Með þessari tvöföldu for lykkju erum við að bæta við staki í tvívítt fylki. Þetta fylki hefur lengdina N og dálkana N , $B[N][N]$. Þegar við erum að bæta við stökum í tvívíða fylkið erum við ekki að nota öll stökin í því. Í fyrstu línu notum við öll stök en í síðustu línu notum við engin stök. $O(N^2)$ gildir fyrir tvívítt fylki. Hinsvegar er síðasta línan ekki með nein stök, þá getum við tekið eitt N í burtu. Svo sést að hver lína notar einum dálki minna þar til engin dálkur er notaður, þess vegna notum við bara helminginn af þessu fylki, eftir að við höfum tekið frá síðustu línuna sem er tóm. Þess vegna er formúlan fyrir þetta dæmi $\frac{N^2-N}{2}$

```
b) public static int[] [] reiknaHS(int[] A) {
    int N = A.length;
    int[] [] B = new int[N][N];

    for (int i=0; i<N; i++) {
        int hsum = 0;
        for (int j=i; j<N; j++) {
            hsum += A[j];
            B[i][j] = hsum;
        }
    }
    return B;
}
```



Input Size	Elapsed time
0	0.002
1	0.006
2	0.037
3	0.114
4	0.349

Höfum hér að $0.349/0.114=3.06$. Líka $0.114/0.037=3.08$. Þegar inntaksstærðin tvöfaldest þá u.þ.b. þrífaldest tíminn. Það er því líklegt að vaxtahraðinn er tilda N^2

Verkefni 3

[Reiknirit] Tiltekið hótél 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 tilfalli.

- 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?
- Lýsið reikniriti sem notar í versta falli $\log N$ tíma til að finna fyrsta lausa herbergið.
- Ef N er mikið stærra en F , þá er hægt að gera betur og nota aðeins $2\log F$ tíma til að finna fyrsta lausa herbergið. Lýsið þessari aðferð og rökstyðjið vaxtarhraða þess.

Lausn:

- Í versta falli þá er þetta $\mathcal{O}(n)$ því manneskja þarf að banka og opna allar dyr í röð þar til herbergi er laust.
- Log N er helmingunarleitun. Við byrjum á að helminga herbergin og opnum þar, ef herbergið er í notkun þá finnum við helming af restinni hægra megin og skoðum aftur. Ef það herbergi er laust þá helmingum við til vinstri. Þetta er gert þar til herbergi er fundið sem laust við hliðina á herbergi sem er í notkun.
- Til þess að finna fyrsta lausa herbergi með þessari jöfnu erum við ekki með helmingunarleitun heldur 2^F og ástæðan fyrir því að við erum með 2 fyrir framan er til þess að koma í veg fyrir að við byrjum á fyrsta herberginu, því $2\log 1$ eru 2, ef herbergi 2 er laust þá vitum við strax að það er bara 1 herbergi tekið. Við hækku mun hraðar dyr á milli herbergja sem við tjékkum ekki á þannig það er auðveldara að komast í hærri tölu ef N er t.d. 10.000 eða hærri.

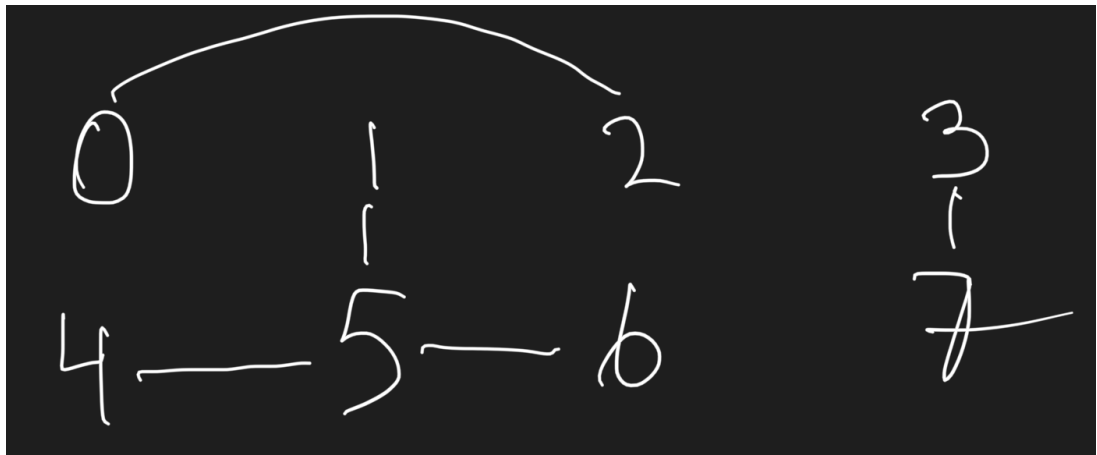
Verkefni 4

[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.

- 0, 1, 0, 3, 1, 1, 3
- 5, 0, 6, 6, 5, 6, 6, 4

Lausn:

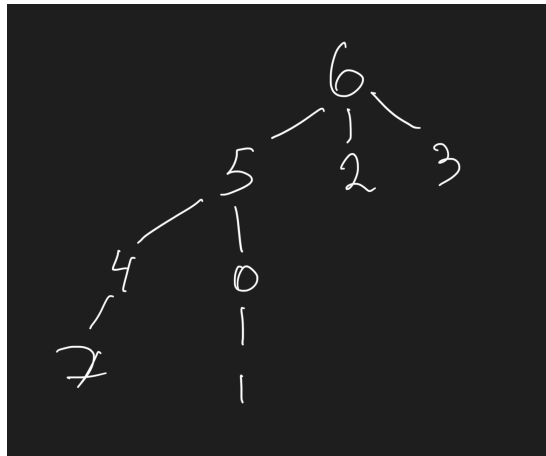
-



```
union(0,2)
union(1,4)
union(1,5)
union(1,6)
union(3,7)
```

- Þetta mun ekki ganga upp því það skiptir ekki máli í hvernig röð þetta er sett upp. Með viktaðri Quick-union þá verður alltaf vesen að ná þessu tré því ef við byrjum með union(6,2) svo union(6,3) svo union(6,5) þá erum við strax komin í vesen með restina. Því öll önnur mengi verða of lítil til að tengjast við 5 þannig 4 og 7 munu tengjast við 6 beint en ekki í gegnum 5. Sama á við um 0 og 1. Þeir tengjast beint

við 6 ekki í gegnum 5 útaf því að þeir eru léttari en hitt mengið.



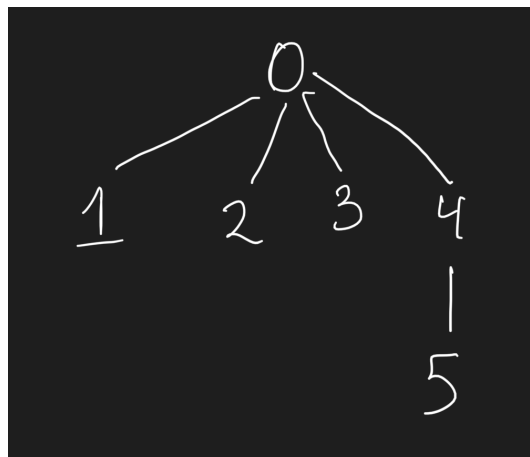
Verkefni 5

[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 styttest). Athugið að inni í hverri union-aðgerð eru tvær find-aðgerðir.

Lausn:



Vegþjöppun á sér stað þegar `union(1,3)` er framkvæmd. Í stað þess að 3 tengist við einn þá tengjast báðar tölur í mengi með 3 í við rótina á 1 sem er 0. Sama gerist þegar `union(3,5)` er framkvæmd. Þá tengist rót fyrir 5 við rót fyrir 3.