Greining reiknirita vor 2024
Heimaverkefni 1 – drög að lausnum

Skila skal þessu verkefni á vefnum Gradescope.

Gradescope tekur við .pdf skjölum. Frágangur á þeim skiptir máli.

Telji nemandi að mistök hafi verið gerð við yfirferð skal tilkynna slíkt á Gradescope. **Skilafrestur er til kl. 22:00 miðvikudaginn 17. janúar.** Gangi þér vel!

1. Aðfellutáknun ($\star$)

Í þessu dæmi á að ákvarða efri-neðri aðfellumörk (e. asymptotic bound) $g(n)$ fyrir gefið fall $f(n)$ þannig að $f(n) = \Theta(g(n))$. Leitist við að g sé á sem einföldustu formi, t.d. innihaldi bara einn lið.

Rifjið upp reiknireglur fyrir logra og veldisvísisföll. Notfærið ykkur ennfremur þá staðreyndir að i) veldisvísisföll a^n vaxa alltaf hraðar en margliðuföll[1] n^b og ii) föll á forminu $(\log n)^k$ vaxa alltaf hægar en margliðuföll[2].

a) $f(n) = n + \sqrt{n}$

Lausn: Neðri mörk eru $n \leq n + \sqrt{n}$. Með $c = 1$ og $n = 1$ fæst að $f(n) = \Omega(n)$. Efri mörk eru $n + \sqrt{n} \leq n + n = 2n$. Með $c = 2$ og $n_0 = 1$ fæst að $f(n) = O(n)$. Þetta sýnir að $f(n) = \Theta(n)$.

b) $f(n) = (\sqrt{2})^{\log_2 n}$

Lausn: Athuga að $(\sqrt{2})^{\log_2 n} = (2^{1/2})^{\log_2 n} = (2^{\log_2 n})^{1/2} = n^{1/2} = \sqrt{n}$ þannig að $f(n) = \Theta(\sqrt{n})$, fáum það með að velja t.d. $c_1 = c_2 = n_0 = 1$ í skilgreiningunni á Θ.

c) $f(n) = \binom{n}{2024}$

Lausn: Athuga að

$$\binom{n}{2024} = \frac{n!}{(n-2024)!\ 2024!} = \frac{n(n-1)\ldots(n-2024+1)}{2024!} \leq \frac{n^{2024}}{2024!}$$

þannig að $\binom{n}{2024} = O(n^{2024})$. Eins fæst að $\binom{n}{2024} = \Omega(n^{2024})$ með því að velja nógu lítið gildi á c og nógu stórt gildi á n_0. Þar með er $\binom{n}{2024} = \Theta(n^{2024})$.

d) $f(n) = \log_{2024}((\log(n^{\sqrt{n}}))^2)$

Lausn: Notum reiknireglur fyrir logra og veldísvísisföll til að umrita $f(n)$. Lograreglan $\log(a^b) = b \log a$ gefur

$$\log(n^{\sqrt{n}}) = \log(n^{n^{1/2}}) = \sqrt{n} \log n.$$

[1] Um allar rauntölur b og $a > 1$ gildir að $\lim_{n\to\infty} \frac{n^b}{a^n} = 0$ (má t.d. sanna með þrepun á b).

[2] Fæst með því að setja $\log n$ í stað n og 2^a í stað a í markgildinu hér að ofan.

Þá fæst

$$\log_{2024}((\log(n^{\sqrt{n}}))^2) = \frac{1}{2024} 2\log(\sqrt{n}\log n) = \frac{2}{2024}(\log\sqrt{n} + \log\log n)$$

þar sem $\log_a x = \frac{\log_b x}{\log_b a}$ og $\log(ab) = \log a + \log b$. Þar sem $\log n$ vex hægar en $\sqrt{n}$ (eiginleiki ii) hér að ofan) fæst að $\log\sqrt{n} \le \log\sqrt{n} + \log\log n \le 2\log\sqrt{n}$ og þá $f(n) = \Theta(\log\sqrt{n}) = \Theta(\frac{1}{2}\log n) = \Theta(\log n)$.

2. Tímaflækja ($\star$)

Í þessu dæmi á að ákvarða tímaflækju nokkurra einfaldra reiknirita. Finnið neðri-efri aðfellumörk Θ fyrir tímaflækjuna $T(n)$, þar sem n er stærð inntaks.

Gerið ráð fyrir að grunnaðgerðir, $+, -, *, /$ og stakasamanburður ($a < b$ og $a = b$) taki fastan tíma og sama gildi um föllin $\max(a, b)$, $\min(a, b)$, $\text{sqrt}(n)$, $\log(n)$, $\text{mod}(a, b)$ og $\text{floor}(x)$. Skýrið svör ykkar stuttlega.

a)
```
def marglida(A[0...n], x)
    p = A[0]
    xpos = 1
    for i = 1 to n
        xpow = x * xpow
        p = p + A[i] * xpow
    return p
```

(Fallið reiknar gildi margliðunnar $p(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$).

Lausn: For-lykkjan er framkvæmd n sinnum. Aðgerðir inni í for-lykjunni taka tíma $\Theta(1)$ og tímaflækjan verður þá $\Theta(n)$.

b)
```
def process(A[1...n])
    for i = 1 to n - 1
        k = i
        for j = i + 1 to n
            if A[j] < A[k]
                k = j
        s = A[i]
        A[i] = A[k]
        A[k] = s
    return
```

Lausn: Hér er hægt að telja fjölda reikniaðgerða nákvæmlega en þar sem við erum að leita eftir aðfellumörkum þarf þess ekki. Athugum að ytri lykkjan er framkvæmd $n - 1 = \Theta(n)$ sinnum og innri lykkjan $(n - i - 1) = \Theta(n)$ sinnum. Tímaflækjan er því $\Theta(n \cdot n) = \Theta(n^2)$ sem er dæmigert fyrir tvær hreiðraðar for-lykkjur á þessu formi.

Athugasemd: Þetta reiknirit kallast valröðun (e. selection sort) og var það tekið fyrir í Tölvunarfræði 2.

c)
```
def process(A[1...n], B[1...(n-2)])
    k = 3
    W = [-1, 2, -1]
    for i = 1 to (n - k + 1)
        B[i] = 0
        for j = 1 to k
            B[i] += A[i + j] * W[j]
    return
```

Lausn: Hér eru tvær hreiðraðar for-lykkjur eins og í b) lið. Það sem er öðruvísi hér er að innri for-lykkjan er alltaf framkvæmd 3 sinnum, óháð gildinu á n og tímaflækjan er því $\Theta(n)$.

d) $\star\star$ (Próf í Stærðfræðimynstrum 2022, breytt)

```
def process(n):
    i = 0
    j = 0
    while i <= n:
        i += j
        j += sqrt(n)
    return i
```

Lausn: Athugum hvenær gildið á i verður hærra en n, þ.e.

$$\sqrt{n} + 2\sqrt{n} + 3\sqrt{n} + \ldots + k\sqrt{n} > n$$

þar sem k er fjöldi ítrana (mínus 1). Leysum

$$\begin{aligned}\sqrt{n}(1 + 2 + \ldots + k) &= n \\ 1 + 2 + \ldots + k &= \sqrt{n} \\ k(k+1)/2 &= \sqrt{n}\end{aligned}$$

Við fáum annars stigs ójöfnu í k sem hefur lausn $k = \frac{-1+\sqrt{1+8\sqrt{n}}}{2}$. Athugum að

$$\sqrt{8}\sqrt{\sqrt{n}} = \sqrt{8\sqrt{n}} < \sqrt{1+8\sqrt{n}} < \sqrt{8\sqrt{n}+8\sqrt{n}} = 4\sqrt{\sqrt{n}}$$

sem sýnir að keyrslutíminn er $\Theta(n^{1/4})$.

3. Tímaflækja ($\star\star$)

Í þessu dæmi skoðum við einfalt reiknirit sem byggir á gagnagrind sem kallast `StatísktFylki` en hún geymir raðað safn hluta af fastri stærð (t.d. bendla, heiltölur, strengi af fastri lengd o.s.frv.). Lengd fylkisins (fjöldi hluta) er föst. Eftirfarandi aðgerðir eru studdar:

- `StatísktFylki(n)`: Úthlutar minni fyrir fylki af stærð n. Öll stök í fylkinu fá gildið núll. Tímaflækja: $\Theta(n)$.
- `StatísktFylki.get(i)`: skilar gildi í sæti i. Tímaflækja: $\Theta(1)$.
- `StatísktFylki.set(i, x)`: skrifar gildi x í sæti i. Tímaflækja: $\Theta(1)$.

Athugum einfalt verkefni sem felst í að ákvarða hvort einhverjir tveir nemendur í hópi nemenda eigi sama afmælisdag. Eftirfarandi reiknirit tekur inn lista af nemendum á forminu (nafn, kennitala). Ef nemendur N1 og N2 eiga sama afmælisdag þá skilar fallið nöfnum þeirra en ef engir slíkir eru til staðar þá skilar fallið NULL.

```
def finna_afmaelisdag(nemendur):
   n = len(nemendur)                          O(1)
   F = StatísktFylki(n)                       ___
   for k = 1 to n                             ___
      (nafn1, afmdagur1) = nemendur[k]        O(1)
      for i = 1 to k                          ___
         (nafn2, afmdagur2) = F.get(i)        ___
         if afmdagur1 == afmdagur2            ___
            return (nafn1, nafn2)             O(1)
      F.set(k, (nafn1, afmdagur1))            ___
   return NULL                                O(1)
```

Ákvarðið tímaflækju einstakra skrefa í reikniritinu (fyllið í ___) og finnið svo efri mörk á keyrslutíma reikniritsins sem fall af fjölda nemenda. Þið getið gert ráð fyrir að hægt sé að vinna með nemendaupplýsingar í föstum tíma.

Athugið að þetta er langt frá því að vera hagkvæmt reiknirit. Hvernig má bæta keyrslutímann með einföldum hætti (svarið með 1 – 2 setningum)?

Lausn: Notum tímaflækju einstakra aðgerða í gagnagrindinni til að fylla í eyðurnar.

```
def finna_afmaelisdag(nemendur):
   n = len(nemendur)                          O(1)
   F = StatísktFylki(n)                       O(n)
   for k = 1 to n                             n
      (nafn1, afmdagur1) = nemendur[k]        O(1)
      for i = 1 to k                          k
         (nafn2, afmdagur2) = F.get(i)        O(1)
         if afmdagur1 == afmdagur2            O(1)
            return (nafn1, nafn2)             O(1)
      F.set(k, (nafn1, afmdagur1))            O(1)
   return NULL                                O(1)
```

Við höfum tvær hreiðraðar for-lykkjur, í ytri lykkjunni er $k = 1, \ldots, n$ og í innri lykkjunni er $i = 1, \ldots, k$ þ.a. heildartíminn verður þá í versta falli

$$T(n) = O(n) + \sum_{k=1}^{n} (O(1) + k \cdot O(1))$$

(ef við höfum fastan fjölda aðgerða þar sem hverja aðgerð má framkvæma í föstum tíma þá verður heildartíminn $O(1)$).

Í formúlum sem innihalda liði á borð við $O(n)$ og $O(1)$ getum við sett fasta úr skilgreiningunni inn í staðinn (fastarnir eru ekki endilega þeir sömu fyrir alla liðina). Þá fæst

$$\begin{aligned} T(n) &= c_1 n + \sum_{k=1}^{n} (c_2 + k \cdot c_3) \\ &= c_1 n + c_2 n + c_3 \sum_{k=1}^{n} k \\ &= (c_1 + c_2) n + c_3 \frac{n(n+1)}{2} \\ &= O(n) + O(n^2) = O(n^2). \end{aligned}$$

Með því að nota hagkvæmari gagnagrind, t.d. hakkatöflu má bæta tímaflækjuna verulega (meira síðar).

4. Bylt pör ($\star\star$)

Meðmælakerfi (e. *recommender system*) eru notuð til að koma með uppástungur að vörum, fréttum, kvikmyndum ofl. Með því að finna notendur sem hafa svipaðan smekk og þú, er hægt að stinga upp á hlutum sem þú gætir haft áhuga á.

Til að þetta sé hægt þarf tölulegan mælikvarða á hversu líkan smekk notendur hafa. Eitt slíkt mat byggir á því hvernig tveir einstaklingar raða sömu hlutum. Athugum tvo áhugamenn um ofurhetjur, köllum þá Jay og Bob. Þeir eru látnir raða n ofurhetjum sem þeir þekkja báðir. Síðan er útbúið fylki A með n stökum sem er þannig að $A[i] = j$ ef hetjan sem Jay setti í sæti i lenti í sæti j hjá Bob. Ef t.d. uppáhaldshetja Jay væri Deadshot en Bob setti hana í 9. sæti þá væri $A[1] = 9$.

Við segjum að par (i, j) sé *bylt* ef $A[i] > A[j]$ þegar $i < j$. Þeim mun fleiri bylt pör sem eru í fylkinu, þeim mun meira eru Jay og Bob ósammála. Ef Jay og Bob raða hetjunum nákvæmlega eins þá eru engin bylt pör (A er raðað).

a) $\star$ Setjið fram reiknirit sem telur fjölda byltra para í fylki A með n stökum, sem keyrir í $\Theta(n^2)$.

b) $\star\star$ Setjið fram reiknirit sem telur fjölda byltra para í tíma $O(n \log n)$. *Ábending*: Flétturöðun.

Lausn (drög):

a)
```python
def telja_byltur(A):
    count = 0
    n = len(A)
    for i in range(n):
        for j in range(i + 1, n):
            if A[i] > A[j]:
                count += 1
    return count
```

Reikniritið ítrar yfir öll pör (i, j), $i < j$ og telur hversu mörg uppfylla $A[i] > A[j]$. Látum keyrslutíma svara til fjölda stakasamanburða. Fjöldi stakasamanburða er $(n-1) + (n-2) + \ldots + 2 = n(n+1)/2 - 1$. Neðri mörk á keyrslutímann eru því $\Omega(n) = n^2$ og efri mörk $O(n) = n^2$ þ.a. keyrslutíminn er $\Theta(n^2)$.

b) Við fylgjum ábendingunni og aðlögum flétturöðun að verkefninu, nánar tiltekið útfærslunni í grein 1.4 í Ericson. Í hvert sinn sem víxlað er á stökum fækkar byltum pörum í fylkinu. Í flétturöðun er einungis víxlað á stökum í fléttu-skrefinu (MERGE) þar sem tveimur röðuðum hlutfylkjum er fléttað saman, $L = A[1 \ldots m]$ og $R = A[m + 1 \ldots n]$. Setjum $n\_bylt = 0$ í upphafi flétturöðunar. Innan L eru engin bylt pör og sama gildir innan R (hvort hlutfylki um sig er raðað). Einu byltu stökin sem koma til greina svara til para (i, j) þar sem i er í L og j er í R. Í hvert sinn sem $A[i] > A[j]$, þá endar $A[j]$ fyrir framan $A[i]$ í fléttaða fylkinu B. Það eru $m - i$ stök fyrir aftan $A[i]$ í L þannig að þessi stakasamanburður leiðir til þess að það fækkar um $(m - i + 1)$bylt pör. Breytum síðustu línum í MERGE þannig:

else if $A[i] < A[j]$
    $B[k] \leftarrow A[i]; i \leftarrow i + 1$
else
    $n\_bylt \leftarrow n\_bylt + (m - i + 1)$ // Lína sem bætist við
    $B[k] \leftarrow A[j]; j \leftarrow j + 1$

Keyrslutími flétturöðunar er $O(n \log n)$, kóðinn sem bættist við framkvæmir 4 reikniaðgerðir og tekur því $O(1)$ tíma. Keyrslutíminn verður því áfram $O(n \log n)$.

5. Leit að næst-stærsta staki í fylki ($\star\star\star$)

Til að ákvarða stærsta stak í fylki A með n stökum þarf $n-1$ stakasamanburði[3]. Næst-stærsta stakið í fylkinu má ákvarða með því að finna fyrst stærsta stakið í $A[1\ldots n]$, víxla á því og fyrsta stakinu í fylkinu, $A[1]$, og finna síðan stærsta stakið í $A[2,\ldots n]$. Fjöldi stakasamanburða verður þá $n-1+n-2=2n-3$.

Setjið fram reiknirit til að finna næst-stærsta stak í fylki með n stökum sem notar í mesta lagi $n+\lceil log_2 n\rceil-2$ stakasamanburði. Ekki nota Quickselect! *Ábending*: Útsláttakeppni.

Lausn: Gerum til einföldunar ráð fyrir að n sé veldi af 2. Lítum á leitina að stærsta og næst stærsta stakinu sem keppni milli staka í fylkinu. Útbúum tvíundatré þar sem laufin svara til staka fylkisins og innri hnútar til staka sem "vinna" keppni milli 2ja staka. Stærsta stakið vinnur keppnina ($n-1$ stakasamanburðir) og lendir í rótinni eftir $\log_2 n$ umferðir[4]. Næst stærsta stakið er stærsta stakið af þeim $\log_2 n$ sem borin voru saman við stærsta stakið, finnum það með $\log_2 n-1$ stakasamanburðum. Alls verður fjöldi stakasamanburða þá $n+\log_2 n-2$.

[3] Hvers vegna er ekki hægt að gera betur?

[4] Tvíundartréið er hrúga (e. heap).