09.71.23 Tölvunarfræði IIa/II

Upptökupróf **19. ágúst, 1999**
kl. 9^{00} - 12^{00}/13^{00}

Fyrri hluta prófsins, sem samanstendur af 6 spurningum þar sem 5 bestu svörin munu gilda, eiga allir að leysa. Seinni hluti prófsins, Dæmi 7 og 8, er aðeins fyrir nemendur í Tölvunarfræði II. ***Öll skrifleg hjálpargögn og reiknivél leyfð***.

Fyrri hluti:

1. Við höfum eintengdan lista þar sem hver hnútur hefur sviðin `Lykill` og `Gogn`, auk sviðsins `Naesti`, sem bendir á næsta hnút í listanum. Skilgreining hnútarins er hér að neðan.

```
struct Hnutur {
    Ltag Lykill;
    Gtag Gogn;
    Hnutur* Naesti;
}
```

Listinn er óraðaður, en við viljum stundum finna í honum gögn sem samsvara tilteknum lykli. Til að flýta fyrir leitinni notum við aðferð sem kallast *færa fremst* (e. move-to-front): Í hvert sinn sem leitað er að lykli *X* í listanum, og hann finnst, er hnútur hans færður fremst í listann. Útfærið aðgerðirnar `Innsetning(X, G)`, sem setur hnút með lykil `X` og gögn `G` fremst í listann, `Eyðing(X)`, sem eyðir hnúti með lykli `X` úr listanum, og `Finna(X)`, sem skilar gögnum þess hnútar sem hefur lykil `X`.

Hver er aðalástæða þess að fundið stak er fært fremst? Er tímaflækja þessarar útgáfu af `Finna` betri en ef ekki væri fært fremst?

2. Útfærið í C++ aðgerðirnar `Split` og `Merge` á tengda-lista útgáfuna af biðröð (`Queue`) í kafla 15.2.2 í kennslubókinni. Aðgerðin `Split` klýfur biðröðina upp í tvennt, þannig að annað hvert stak í henni er áfram í gömlu biðröðinni (fyrsta, þriðja, fimmta, ... stak), en síðan er skilað annari biðröð sem inniheldur hin stökin (annað, fjórða, sjötta, ...) í þeirri röð sem þau voru upphaflega. Aðgerðin `Merge` er eins konar andhverfa `Split`, því hún tekur inn biðröð og fléttar hana inní biðröðina sem aðgerðin er skilgreind á. Fyrsta stakið kemur þannig úr upphaflegu biðröðinni, annað stakið úr inntaksbiðröðinni, það þriðja úr upphaflegu, o.s.frv.

3. Við viljum breyta gagnagrindinni tvíleitartré (e. binary search tree) þannig að það noti *tafða eyðingu* (e. lazy deletion). Þá hefur hver hnútur heiltölusviðið `InUse`, auk sviðanna `Element`, `Left` og `Right`. Við innsetningu hnútar er gildi `InUse`-sviðs hans sett 1 (satt), en síðan við eyðingu hans er gildi sviðsins sett 0 (ósatt), í stað þess að taka hnútinn sjálfann út úr trénu. Tvíleitartréð inniheldur því á hverjum tíma aðeins þá hnúta sem hafa `InUse` == 1 þó að gagnagrindin geti innihaldið fleiri hnúta, sem búið er að eyða.

Miðið við útfærsluna í kafla 18.1 í kennslubókinni og sýnið nýja útgáfur af föllunum `Remove`, `Find` og `FindMax` í C++, sem nota tafða eyðingu eins og lýst er að ofan.

4. Hægt er að skilgreina forgangsbiðröð þar sem bæði `DeleteMin` og `DeleteMax` taka $O(\log n)$ tíma. Þessi gagnagrind kallast *min-max heap* og hún er alveg eins og venjuleg hrúga, nema að hrúguskilyrðið er nú þannig að fyrir hnút X á jöfnu dýpi (0, 2, 4, ...) er X minna en foreldri sitt, en stærra en afi sinn, og fyrir hnút Y á oddatöludýpi (1, 3, 5, ...) er Y stærra en foreldri sitt, en minna en afi.

a) Sýnið dæmi um slíka hrúgu með a.m.k. 12 stökum.
b) Hvernig eru stærstu og minnstu stökin fundin?
c) Lýsið (í orðum) aðferð til að setja inn stak í min-max hrúgu.

5. Vektor inniheldur N röðuð stök, en á eftir þeim í vektornum koma $f(N)$ óröðuð stök. Hvaða röðunaraðferð væri best að nota ef

a) $f(N) = O(1)$ (þ.e. fastur fjöldi óraðaðra staka, óháð fjölda röðuðu stakanna)
b) $f(N) = O(\log N)$
c) $f(N) = O(\sqrt{N})$.

Rökstyðjið hvern lið fyrir sig með skýrskotun til tímaflækju röðunaraðferðanna og hegðun þeirra á vektor af þessari gerð.

6. Í hakkatöflunni í kafla 19 í kennslubókinni er aðgerðin `IsEmpty` ekki útfærð. Væri hægt að útfæra hana þannig að láta hana skila röksegðinni `(CurrentSize==0)`, ef breytan `CurrentSize` væri jafnframt lækkuð um 1 í eyðingaraðgerðinni (og hækkuð í innsetningar-aðgerðinni, eins og gert er)? Rökstyðjið svar ykkar.

Seinni hluti:

Eftirfarandi dæmi eru aðeins fyrir nemendur í Tölvunarfræði II.

7. *a)* Í forgangstöflu (`PrecTable`) reiknivélarinnar (bls. 370 í kennslubókinni) eru nokkur gildi sem aldrei eru notuð. Hver eru þau og hvers vegna eru þau ekki notuð?

b) Gögn sem kóðuð eru með Huffman aðferð eru viðkvæm fyrir því að bitar tapast í geymslu eða sendingu. Lýsið því hvað gerist við afkóðun á afgangnum af gögnunum og lýsið gróflega hvað hægt er að gera við þessu.

8. Eins og komið hefur fram í námskeiðinu lendir aðferð Dijkstra í vandræðum ef net hefur neikvæðar þyngdir á stikum. Væri þá ekki hægt að leggja fasta C við allar þyngdirnar þannig að þær yrðu allar jákvæðar og finna stystu leiðir í því neti? Rökstyðjið svar ykkar.