09.71.23 Tölvunarfræði IIa/II

Lokapróf **14. maí, 1998**

kl. $14^{00} - 17^{00}/18^{00}$

Fyrri hluti 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. Upphafsstilling n staka vektors tekur $\Theta(n)$ tíma. Ef aðferðin sem notar vektorinn tekur mun minni tíma, þá er upphafsstillingin ráðandi liður í tímaflækju forritsins. Hægt er að losna við að eyða tíma í upphafsstillingu með svokallaðri *sýndarupphafstillingu* (e. virtual initialization), en hún notar að vísu meira minnispláss.

Ef n staka vektorinn sem á að upphafstilla heitir T, þá skilgreinum við tvo aðra n staka vektora a og b, ásamt heiltölubreytunni fj. Teljarinn fj segir hversu mörg stök í T hafa verið upphafsstillt. Gildin $a[0]$ til $a[fj]$ segja okkur hvað stök það eru, þannig að $a[0]$ inniheldur vísi þess staks T sem var upphafsstillt fyrst, $a[1]$ vísinn á öðru stakinu sem var upphafsstillt, o.s.frv. Síðan, ef $T[i]$ var k-ta stakið sem var upphafsstillt þá er $b[i] = k$.

Til að athuga hvort að hólfið $T[i]$ hafi fengið eitthvert upphafsgildi, þá athugum við fyrst hvort $0 \le b[i] < fj$. Ef svo er þá athugum við hvort $a[b[i]]$ sé $= i$. Ef það gildir þá hefur $T[i]$ verið upphafsstillt.

a) Teiknið upp mynd af vektorunum þremur eftir að gildin 10, 50 og 20 hafa verið sett inn í hólf númer 0, 3 og 6 í 8 staka vektornum T.

b) Útskýrið hvernig og hvers vegna aðferðin, sem lýst er hér að ofan til að ákvaða hvort hólf í T hafi verið upphafsstillt, virkar.

c) Skrifið föll í C++ fyrir sýndarupphafsstilltan vektor. Eitt fall sem setur gildi v í hólf i og annað sem skilar gildinu í hólfi i, ef það hefur verið upphafsstillt, annars villu (t.d. exception).

2. Lýsið aðferð til að finna k minnstu stökin í vektor með n stökum. Hver er tímaflækja aðferðarinnar (sem fall af n **og** k)? Fyrir hvaða gildi á k verður hagkvæmara að raða vektornum?

3. Við viljum bæta aðgerðinni `FindMin` við gagnagrindina `Stack`. Aðgerðin skilar minnsta gildinu sem er í hlaðanum á hverjum tíma (en eyðir því ekki). Hægt er útfæra aðgerðina hraðvirkt með því að hafa tvo hlaða, annan undir gildin sjálf, en hinn sem heldur utanum hvert er minnsta stakið á hverjum tíma.

Útfærið þessa nýju gagnagrind, köllum hana `MinStack`, í C++, með því að nota ykkur gagnagrindina `Stack`.

4. Er eyðingaraðgerðin (`Remove`) í útfærslunni á tvíleitartré í kennslubókinni víxlin? Það er, ef við eyðum hnútunum u og v út úr trénu T, fæst þá sama tré og ef við hefðum eytt þeim í röðinni *v* og *u*? Myndi það breyta einhverju ef við settum alltaf í staðinn fyrir eydda hnútinn næsta hnút á eftir í "inorder"-röðinni?

5. Hægt er að skilgreina *3hrúgu* (e. 3heap), þar sem hver hnútur hefur 3 börn í stað tveggja í venjulegri hrúgu.

a) Lýsið uppsetningu á 3hrúgu í vektor (svipað og venjuleg hrúga).

b) Skrifið hraðvirka útfærslu á `DeleteMin` fyrir 3hrúgu í C++. Hver er tímaflækjan?

c) Almenna útgáfan væri *k*hrúga. Hvað hluti `DeleteMin` verður ódýrari eftir því sem *k* stækkar, og hvaða hluti dýrari í keyrslutíma?

6. Gerum ráð fyrir að raða eigi milljón staka vektor sem er þannig að fyrstu 999.990 stökin eru röðuð, en öftustu 10 stökin eru handahófsstök sem bætt hefur verið aftan við. Hvaða aðferð af eftirtöldum, eins og þeim er lýst í kennslubókinni, hentar best til að raða slíkum vektor: Innsetningarröðun, Shell-röðun, Quicksort. Rökstyðjið svar ykkar með útreikningum.

Seinni hluti:

Eftirfarandi dæmi eru aðeins fyrir nemendur í Tölvunarfræði II.

7. Í kennslubókinni (bls. 361-2) er sýnt hvernig á að nota hlaða (e. stack) til að reikna út gildi reiknisegðar á "postfix"-formi. Lýsið samskonar aðferð fyrir segðir á "prefix"-formi. Sýnið aðferðina á segðinni `1+2*3^4` (þið þurfið fyrst að breyta segðinni yfir á "prefix"-form).

8. Lát $G=(V,E)$ vera stefnunet (e. digraph), þar sem hver stika (u, v) hefur gildi $r(u, v)$, með $0 \leq r(u, v) \leq 1$, sem táknar áreiðanleika samskipta frá hnúti *u* til hnútar *v*. Til dæmis ef $r(u, v) = 0.98$ þá eru 98% líkur á því að skeyti komist óskaddað frá *u* til *v*. Þannig má líta á netið G sem lýsingu á tölvuneti og $r(u, v)$ sem líkurnar á því að tengingin frá tölvu *u* til tölvu *v* virki.

Lýsið hagkvæmri aðferð til að finna áreiðanlegustu leiðinna frá einum gefnum hnúti til annars í *G*. Hver er tímaflækja aðferðarinnar?