Heimadæmi 8, TÖL203. Elfar Oliver Sigurðarson, [eos35@hi.is](mailto:eos35@hi.is)

Æfingadæmi  
  
Þurfum alltaf að sækja eftir stærðarröð frá eldra fylki þegar við tvöföldum.

Mod reikningur með ASCII kóða. Modið er 5. ASCII kóðinn er gefinn.

1. Sjálfstæð keðjun. E69, A65, S83, Y89, Q81, U85, T84, gerir mod listann að 4,0,3,4,1,0,4  
     
   Mynd fylgir seinna
2. Línuleg könnun. Núna er modið 4.   
     
   E = 1, E fær pláss 1   
   A = 1, A fær pláss 2.   
     
   Þegar fylkið er helmingsfullt viljum við tvöfalda fylkið. Núna er modið orðið 8.   
     
   A mod 8 = 1, A fær sæti 1.   
   Y mod 8 = 1, Y fær sæti 2.  
   E mod 8 = 5, E fær sæti 5.   
   S mod 8 = 3, S fær sæti 3.   
     
   Núna er fylkið helmingsfullt og þurfum að stækka uppí 16.   
     
   E mod 16 = 5, E fær sæti 5.   
   A mod 16 = 1, A fær sæti 1.   
   S mod 16 = 3, S fær sæti 3.  
   Y mod 16 = 9, Y fær sæti 9.  
   Q mod 16 = 1, Q fær sæti 2.  
   U mod 16 = 5, U fær sæti 6.  
   T mod 16 = 4, T fær sæti 4.  
     
   Allt búið
3. [Ýmsar gagnagrindur] [Próf 2019] Teiknið eftirfarandi gagnagrindur þegar búið er að  
   setja stökin 5, 3, 8, 1, 2, 6, 4, 9, 7 inn í þessari röð.  
   a. Hrúgu (max heap). Hér þarf bara að teikna lokaniðurstöðu. Teiknið bæði tréð og  
   samsvarandi fylki.  
   b. Tvíleitartré (binary search tree). Hér þarf bara að teikna lokaniðurstöðu.  
   c. Vinstri-hallandi rauð-svart tré (left-leaning red-black tree). Þetta er sú útgáfa sem  
   við fórum yfir í námskeiðinu. Teiknið gagnagrindina eins og hún lítur út eftir hverja  
   innsetningu, táknið rauða leggi með r. Til einföldunar eru fyrstu fjögur trén gefin.  
   d. Hakkatafla af stærð 12 með hakkafallinu ℎ(𝑘) = 3𝑘 og árekstrar eru leystir með  
   Linear Probing. Hér þarf bara að teikna lokaniðurstöðu.
4. Taka út size() og isEmpty()
5. Dfg
6. Vinna okkur afturábak úr æfingadæmunum. K = talan sjálf. Ekki nákvæmlega eins og æfingadæmið þar sem í æfingadæminu fékk útkoman eftir modreikning sætið, en hérna er það K(fyrir útreikning) sem fær sætið. K = K, ekki K % n = K.  
     
   Hérna þarf að skoða H(k) og sæti jafnóðum fyrir hverja tölu. Svipað og í Sudoku, þá eru öll K og sæti tengd, nema í þessu getur staðsetning K haft áhrif á K+1. Best er að skoða K sem hafa sömu tölu bæði í H(k) og Sæti. Ef talan er sú sama, þá uppfyllir það skilyrðið að það K gæti hafa verið fyrsta talan. Hver sem er af þeim þ.e.a.s.. Sem þýðir 3, 7 og 8 eru svo sannarlega ekki seinust, og mögulega fyrst.  
     
   Eftir að hafa ákveðið það, þá er best að skoða hvaða tölur eiga sameiginlegt H(k).   
     
   K8 kom t.d. á undan K1, því þau deila H(k) en K1 er í aftara sæti. Sama á við um að skoða H(k) og Sæti. K1 er með Sæti 6, K5 er með H(k) 6 en er með sæti 0. Það þýðir K5 kom á eftir 1.   
     
   Sama með K2 og K5. K2 á að vera í Sæti 0, en það var upptekið og fékk þar með sæti 1, svo K2 kom á eftir K5. Af öllum þeim K(8, 1, 5, og 2) sem ég minntist á, gæti einungis K2 hafa komið seinast.  
     
   Núna kemur þægilegra samband milli talna þar sem 3 og 7 eru í sínu sæti, og við sjáum á H(k) og Sæti K10 að 10 kom á eftir þeim báðum. 10 hefði getað fengið sæti 3, en 7 var nú þegar komið og 10 fékk ekki sæti 2 þar sem K3 var nú þegar kominn.  
     
   Lokasvar: K2 og K10

|  |  |  |
| --- | --- | --- |
| K | H(k) | Sæti |
| 5 | 6 | 0 |
| 2 | 0 | 1 |
| 3 | 2 | 2 |
| 7 | 3 | 3 |
| 10 | 2 | 4 |
| 8 | 5 | 5 |
| 1 | 5 | 6 |

1. Keyrum GenAttack einu sinni og svo þurfum við að gera okkar eigin skrá sem tengist GenAttack *EKKI*.  
   Pípun = ör og nafnið á skránni. > fileName.java  
   javac GenAttack.java  
   java GenAttack 16 > attack.txt