Institut for Matematik og Datalogi
Syddansk Universitet, Odense

1. februar 2015
RF

DM507 – Opgaver uge 6

I denne uge er timerne med opgaveregning (TE-timer/eksaminatorier) undtagelsesvis uden forberedelse, dvs. du arbejder med opgaverne i klassen. Resten af året vil TE-timerne være *med* forberedelse, dvs. der gennemgås opgaver du har løst (eller forsøgt løst) *før* timerne.

Det er vigtigt for dit udbytte af kurset at du resten af året reelt forsøger at løse opgaverne før TE-timerne – alene og/eller sammen med andre (f.eks. i studiegrupper). Afsæt et antal minutter til at tænke over hver opgave. Forhåbentligt når du så langt, at du får skrevet en løsningen ned til hovedparten af dem. Lidt svære opgaver vil være mærket med $(*)$, og mere svære opgaver vil være mærket med $(**)$

Som man kan se af skemaet, har man i kurset hver uge tre dobbelttimer, og man veksler hver uge mellem 2 dobbelttimer forelæsninger (I-timer) plus én dobbelttime opgaveregning (TE-timer) og det omvendte. For holdet T1 er der flyttet en TE-dobbelttime fra allerførst i semesteret til allersidst, for ikke at have TE-timer før den første I-time. Dette hold er derfor en "halv uge bagud"—mere præcist skal de læse overskriften "Opgaver uge 6" som "opgaver til første TE-dobbelttime i uge 7", læse overskriften "Opgaver uge 7" som "opgaver til anden TE-dobbelttime i uge 7 samt TE-dobbelttimen i uge 8", og så videre.

I denne uge vil det være en fordel at medbringe laptop (med Java installeret) til TE-timerne.

Eksaminatorier

1. Cormen et al. problem 1.1 (side 14). Erstat dog "microseconds" med "nanoseconds" (dvs. 10^{-9} sekunder), da dette ca. er hvad en CPU-cykel tager på en moderne processor. Der er nok at udfylde søjlerne *second*, *hour*, *month*, *century*. For nogle af indgangene kan man finde svaret ved matematisk udregning, for andre må man prøve sig frem ved at indsætte forskellige værdier af n.

2. Brodal noter om puslespil, opgave 1.

3. Brodal noter om puslespil, opgave 2. "Optimal følge af ombytninger" betyder et antal ombytninger som angivet af sætning 1 i noterne. Opgaven viser at andre algoritmer end "grådige algoritmer" (dvs. algoritmer som altid bringer mindst eet element på plads) kan være optimale for dette puslespilsproblem.

4. Lav et Java-program (eller Python-program) som genererer en tilfældig permutation af tallene 1 til n (for et n som er en input parameter). F.eks. kan man i Java bruge typen `ArrayList`, samt metoden `shuffle` fra `Collections` utility klassen. Udskriv tallene i din permutation.

5. ($*$) I en permutation i et array (som i opgaven ovenfor) kan man definere cykler på samme måde som for puslespillet fra første forelæsning: et tal x, som står på plads y i arrayet, giver en pil fra plads y til plads x (dvs. hvis tallet 2 står på plads 5, er der en pil fra plads 5 til plads 2), og en samling pile, der hænger sammen i en cyklisk kæde, kaldes en cykel (eller end kreds).

   Lav en algoritme (og en implementation heraf) som tæller antal cykler i permutationen. Hvad er køretiden for dit program som funktion af n? Brug (mange) gentagne kørsler af dit program til at give et bud på sandsynligheden for at der i en tilfældig permutation med $n = 12$ er k cykler, for $k = 1, 2, \ldots, 12$ (jf. figur 3 i noterne, hvor n dog er 64). Find også det gennemsnitlige antal cykler i dine eksperimenter. Passer dit tal med (er tæt på) formlen på side 4 i noterne?