Høgskolen i Gjøvik
Avdeling for teknologi

Kontinuasjonseksamen

FAGNAVN: **Algoritmiske metoder**

FAGNUMMER: **IMT2021**

EKSAMENSDATO: **9. august 2005**

KLASSE(R): **03HBINDA / 03HBINFA / 03HBMETEA / div. andre**

TID: **09.00-14.00**

FAGLÆRER: **Frode Haug**

ANTALL SIDER UTLEVERT: **4 (inkludert denne forside)**

TILLATTE HJELPEMIDLER: **Alle trykte og skrevne.**

- Kontroller at alle oppgavearkene er tilstede.
- Innføring med penn, eventuelt trykkblyant som gir gjennomslag. Pass på så du ikke skriver på mer enn ett innføringsark om gangen (da det blir uleselige gjennomslag når flere ark ligger oppå hverandre).
- Ved innlevering skilles hvit og gul besvarelse, som legges i hvert sitt omslag.
- Oppgavetekst, kladd og blå kopi beholder kandidaten til klagefristen er over.
- Ikke skriv noe av din besvarelse på oppgavearkene. Men, i oppgavetekst der du skal fylle ut svar i tegning/tabell/kurve, skal selvsagt dette innleveres sammen med hvit besvarelse.
- Husk kandidatnummer på alle ark. Ta vare på dette nummeret til sensuren faller.

Oppgave 1 (teori, 25%)

Denne oppgaven inneholder tre uavhengige oppgaver fra kap. 9, 11 og 15 i læreboka.

a) Du skal utføre Quicksort på teksten ”F R I M E R K E N E” (blanke regnes ikke med). **Lag en figur tilsvarende fig. 9.3 side 119 i læreboka**, der du for hver rekursive sortering skriver de involverte bokstavene og markerer/uthever hva som er partisjonselementet.

b) Følgende prioritetskø, organisert som en heap, er gitt:

74 58 59 52 57 56 58 47 48 49 50 51

Utfør etter tur følgende operasjoner på denne heap: insert(57), insert(75), remove(), remove() og replace(50). **Skriv opp heapen etter at hver av operasjonene er utført.**

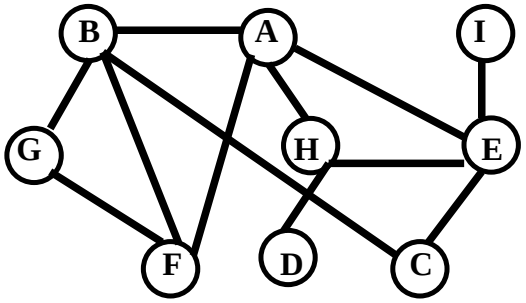
NB: For hver av operasjonene skal du operere videre på den heapen som ble resultatet av den forrige operasjonen.

c) Legg key’ene ”FRIMERKESLIKKER” (i denne rekkefølge fra venstre til høyre) inn i et 2-3-4 tre. **Tegn opp treet etterhvert som bokstavene legges inn.**
Gjør også om sluttresultatet til et Red-Black tre.

Oppgave 2 (teori, 25%)

Denne oppgaven inneholder tre uavhengige oppgaver fra kap. 29 & 31 i læreboka og om FSM.

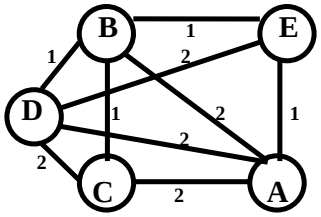
a) Vi har grafen :



a1) **Skriv opp nabomatrisen** for dette tilfellet.

a2) **Tegn dybde-først søketreet** for dette tilfellet (dvs. ved bruk av nabomatrise), når ”Search” (s.424) og ”Visit” (s.427) fungerer som angitt i læreboka.

b) Følgende vektede (ikke-rettede) graf er gitt:



Hver nodes naboer er representert i en naboliste. Alle listene er *sortert alfabetisk*.
Tegn opp minimums spenntreet for denne grafen, etter at koden s.455 i læreboka er utført (der "priority" er lik "t->w").

Tegn også opp innholdet i prioritetskøen etterhvert som konstruksjonen av minimums spenntreet pågår (jfr. fig.31.4 i læreboka). **NB:** Husk at ved lik prioritet så vil noden sist oppdatert (nyinnlagt eller endret) havne først i køen.

c) Følgende deterministiske endelige tilstandsmaskin (FSM) er gitt:

$M = (\{q_0, q_1, q_2, q_3\}, \{0, 1\}, \delta, q_0, q_3)$

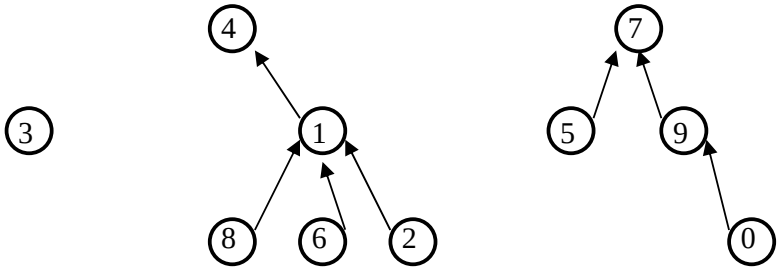
Transisjonstabellen:

$\delta(q_0, 0) = q_1$	$\delta(q_1, 0) = q_1$	$\delta(q_2, 0) = q_1$	$\delta(q_3, 0) = q_1$
$\delta(q_0, 1) = q_0$	$\delta(q_1, 1) = q_2$	$\delta(q_2, 1) = q_3$	$\delta(q_3, 1) = q_0$

Tegn tilstandsmaskinen.
Hva slags setninger/språk godtar den ?

Oppgave 3 (koding, 20%)

Vi har en mengde med N+1 noder. Disse er identifisert ved tallene 0, 1, 2, …... , N. Nodene er strukturert i en skog av rotrettede trær (hver node *er* med i *ett* tre, også trær bestående av bare rot-noden). Ett eksempel for ti noder (N=9):



a) **Beskriv hvordan nodestrukturen kan representeres** vha. *kun* heltallsarrayen:
`int foreldre[N+1]` **Angi innholdet i denne ut fra eksemplet ovenfor.**

b) **Lag den *ikke-rekursive* funksjonen** `void vei_mot_rot(int node)`

Denne funksjonen skriver ut veien fra noden gitt ved parameteren 'node' opp til roten av det rotrettede treet den tilhører. Følgende utskrift *skal* foregå om funksjonen ble kjørt på eksemplet ovenfor med utgangspunkt i node nr.2 (kallet ved: **`vei_mot_rot(2)`**):

```
Fra node: 2  til node: 1  til node: 4  som er rot.
```

c) **Lag den *rekursive* funksjonen** **`void vei_fra_rot(int node)`**

Denne funksjonen skriver ut veien fra roten og ned til 'node'. Følgende utskrift *skal* foregå om funksjonen ble kjørt på eksemplet ovenfor og målnoden var nr.0 (**`vei_fra_rot(0)`**):

```
Fra rot: 7  til node: 9  til node: 0
```

NB: Det er *ikke* lov å direkte lete seg baklengs fra 'node' og opp til rota (som i oppgave b). Rekursjonen selv skal ivareta denne "baklengse letingen".

Oppgave 4 (koding, 30%)

En populær hjernetrimoppgave er for tiden "SUDOKU". Utgangspunkt for en slik *kan* f.eks. være:

	5	3	2		8	7		
4			7				8	
1						4		
	2		3					
7	3			5			2	8
					2		5	
		9					3	6
	8				9			5
		1	4		5	2	9	

Oppgaven går kort og godt ut på å fylle de tomme feltene slik at *alle* rader/linjer, kolonner og 3x3 bokser (de ni rutene omsluttet av tykkere streker) hver inneholder *alle* tallene fra 1-9 *en* gang.

Lag funksjonen **`bool finn_losning()`** som (om mulig) finner *en* løsning på en slik oppgave. Til dette *skal* det brukes arrayen: `int sudoku[9][9];`

Du kan forutsette at når funksjonen starter (kalles fra 'main'), så ligger de initielle verdiene for rutenettet (lest inn fra fil) allerede i denne arrayen. Du trenger *ikke* å skrive ut løsningen, det holder at funksjonen fyller arrayen med lovlige verdier i alle sine 81 "skuffer"/ruter, og returnerer 'true'.

NB1: Løsningen trenger (og bør) *ikke* være rekursiv. Dette løses nok best vha. iterasjon (løkker).

NB2: Slike SUDOKUer er av ulike vanskelighetsgrader. Vårt program skal løse brett av den enklere typen. Dvs. til enhver tid *bør* det være *minst en* rute som bare kan inneholde *en* entydig verdi. Denne forenklingen medfører at funksjonen *ikke* skal prøve å sette inn flere ulike verdier i en og samme rute (backtracking).

Hint1: Altså: i en gitt rute kan det stå en verdi som *ikke* er brukt av noen av de andre rutene på samme rad/linje, kolonne eller i samme 3x3 område som den selv.

Hint2: Det er nok lurt å gå gjennom hele rutenettet flere ganger (inntil alle tallene er på plass), og for hver runde sette på plass så mange nye tall som mulig. Om en slik gjennomgang ikke får på plass
minst ett nytt tall, så er oppgaven uløselig med vår algoritme, og 'false' returneres.

Lykke til !

frode@haugianerne.no