i

Informasjon om eksamen

Velkommen til eksamen i IN2010/INF2220 - Algoritmer og datastrukturer.

Tid: Fredag 29. november 2019 kl. 14.30–18.30 (4 timer).

Tillatte hjelpemidler: ingen.

Det anbefales å lese raskt gjennom eksamen før du setter i gang, for å få ett overblikk og for å notere eventuelle spørsmål til faglærer.

Oppgavesettet består av tre seksjoner som er vektet omtrent slik:

- **30%** for seksjon 1 (korte svar, automatisk rettet)
- **45%** for seksjon 2 (spilldesign)
- **25%** for seksjon 3 (prioritetskøer og heap)

For hver oppgave er maksimal poengsum oppgitt. Dette indikerer vanskelighetsgraden/viktigheten av oppgaven. Ikke bruk alt for mye tid på en oppgave med 1 poeng. Kun oppgave **1d** og **1e** gir minuspoeng for feil svar.

I oppgave **2b** skal du svare med digital håndtegning. Du skal bruke skisseark som du får utdelt.
Det er **ikke** anledning til å bruke bruke digital håndtegning på andre oppgaver enn **2b**.
Det blir **ikke** gitt ekstratid for å fylle ut informasjonsboksene på skisseark (engangskoder, kand.nr. o.l.).

For grafoppgavene kan du anta at grafen ikke inneholder

- enkle løkker (en kant fra en node til seg selv)
- parallelle kanter (mer enn en kant mellom et par av noder)

For oppgaver om P og NP skal du anta at $P \neq NP$.

For oppgaver som ber om kode (**3g** og **3h**), kan du svare med pseudokode/naturlig språk, men jo næremer en fullt detaljert algoritme svaret ditt er, jo bedre. I **2f** er valg av beskrivelse helt opp til deg, selv om du er gitt en kode-editor som svarfelt.

1(a) Kanter i et spenntre

Anta at G er en sammenhengende graf med 12 noder og 20 kanter og at S er et spenntre for G.
Hvor mange kanter har S?

Svar her: 11

Maks poeng: 1

1(b) Kanter i en spennskog

Grafen G har 6 noder og 6 kanter.
Vi bruker Kruskals algoritme til å finne F (spennskogen til G).
Avhengig av hvordan G ser ut, hva er det minste antall kanter F kan inneholde?

Svar her: 5

Maks poeng: 3

1(c) Minimale spenntrær

Kryss av for den, eller de algoritmene, som gitt en sammenhengende vektet graf, finner et **minimalt spenntre** for grafen.

Velg ett eller flere alternativer:

- Bredde-først-søk (BFS)
- Kruskal
- Dybde-først-søk (DFS)
- Prim
- Borůvka

Maks poeng: 5

1(d) FEM

Gitt følgende problem, kalt FEM:
INSTANS: et heltall n
SPØRSMÅL: er $n = 5$?

Er FEM i NP?

Velg ett alternativ:

- Ja, fordi FEM er i P
- Nei, fordi FEM er i P
- Ja, fordi FEM ikke er i P
- Nei, fordi FEM ikke er i P

Merk: Du får 1 poeng for riktig svar, 0 poeng for ubesvart og -1 poeng for feil svar.
Det innebærer at du er forventet å få færre poeng for ren tipping enn ved å la være å svare.
Kun ett alternativ gir 1 poeng.

Maks poeng: 1

1(e) SUDOKU-SJEKKER

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

Vi sier at et sudokubrett har orden n, hvis boksene, kolonnene og radene har størrelse n^2.
Det vil si at man skal fylle inn tallene fra 1 til n^2.
For eksempel vil et vanlig sudokubrett (1–9) ha orden 3.
I et ferdigløst sudokubrett av orden n inneholder hver boks, kolonne og rad tallene fra 1 til n^2 nøyaktig en gang.
Noen uferdige sudokubrett kan ha flere enn en unik løsning (for eksempel et tomt brett).

Alice har laget et program som heter Aloku.
Aloku er ganske bra på å løse sudokubrett.
Gitt et uløst sudokubrett av orden n gjør programmet følgende:

- Hvis $n > 100$, er brettet for stort og programmet terminerer uten å prøve å løse det
- Hvis $n \leq 100$ og brettet har nøyaktig en unik løsning, vil programmet returnere denne løsningen i endelig tid
- Hvis $n \leq 100$ og brettet har flere enn en unik løsning, vil programmet aldri terminere (kjøre for alltid)

SUDOKU-SJEKKER:
INSTANS: Et uløst sudokubrett av orden n
SPØRSMÅL: Vil Aloku løse brettet i endelig tid?

I denne oppgaven antar vi at $P \neq NP$.

Velg ett alternativ angående kompleksiteten til problemet SUDOKU-SJEKKER, som gitt et uløst sudokubrett skal avgjøre om Aloku finner en løsning eller ikke.

Velg ett alternativ:

- SUDOKU-SJEKKER er uavgjørbart, siden det er mulig at Aloku kjører for alltid. Vi må derfor løse stoppeproblemet som er uavgjørbart.
- SUDOKU-SJEKKER er NP-komplett, fordi det er kjent at SUDOKU er NP-komplett. Vi trenger ikke å løse stoppeproblemet.
- SUDOKU-SJEKKER kan løses i konstant tid, fordi Aloku kun kan løse endelig mange brett.

Merk: Du får 5 poeng for riktig svar, 0 poeng for ubesvart og -3 poeng for feil svar. Det innebærer at du er forventet å få færre poeng for ren tipping enn ved å la være å svare.
Kun ett alternativ gir 5 poeng.

Maks poeng: 5

1(f) Kodeanalyse

```
boolean loops(int n) {
    boolean output = true;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            for (int k = 0; k < n; k++) {
                for (int l = 0; l < n; l++) {
                    for (int m = 0; m < n; m++) {
                        output = !output;
                    }
                }
            }
        }
    }
    return output;
}
```

o(n^5)

Gi kjøretiden til kodesnutten i O-notasjon:

Maks poeng: 1

1(g) 5-CLIQUE

En k-klikk, eller en klikk av størrelse k, i en graf $G = \langle V, E \rangle$ er en delmengde $C \subseteq V$ av k noder som utgjør en komplett graf. Det vil si at hvis $u, v \in C$ er to forskjellige noder i klikken, så må $\{u, v\} \in E$.
I problemet 5-CLIQUE skal du avgjøre om en graf inneholder en klikk av størrelse 5.

5-CLIQUE
INSTANS: En graf G
SPØRSMÅL: Inneholder G en 5-klikk?

Hint: Se forrige oppgave.

Velg ett alternativ:

- 5-CLIQUE er NP-komplett
- 5-CLIQUE er i P

Maks poeng: 3

1(h) Korteste avstander i grafer

For hver graftype, velg den **raskeste** algoritmen som finner korteste avstander i grafen.

	Bredde-først-søk	Dijkstra	Topologisk Sortering	Bellman-Ford
Ingen negative sykler	○	○	○	○
Ingen negative kanter	○	○	○	○
Vektet DAG	○	○	○	○
Uvektet	○	○	○	○

Forklaring:

- Uvektet - en uvektet graf. Grafen er enten rettet eller urettet.
- Vektet DAG - en vektet, rettet asyklisk graf. Kantene kan ha negativ vekt.
- Ingen negative kanter - grafen er vektet, men ingen kanter har negativ vekt. Grafen er enten rettet eller urettet.
- Ingen negative sykler - grafen er vektet, men inneholder ingen sykler med negativ vekt. Grafen er enten rettet eller urettet.

Maks poeng: 8

1(i) Huffmankode

Gitt huffmankodetabellen:

A	0
C	100
G	101
T	11

Hvilken tekststreng står koden 1010111101000 for?

Svar her: gattaca

Maks poeng: 1

1(j) Lukket hashing

Gitt en hashtabell av lengde 5 som kan lagre heltall.
Vi bruker lukket hashing med lineær prøving og hashfunksjonen $h(k) = k \, mod \, 5$.
Vi legger følgende elementer inn i tabellen i denne rekkefølgen:

$$17, 98, 59, 32, 40.$$

På hvilken indeks havner siste element (40)?

Svar her: 1

Maks poeng: 2

i

Introduksjon

Denne seksjonen er delt inn i ni oppgaver (a-i) som delvis bygger på hverandre.
Hvis du står fast på en oppgave er det mulig å gå videre til neste, men det er sterkt anbefalt å gjøre dem i den rekkefølgen de er gitt.
Under følger viktig informasjon om oppgaven som gjelder gjennom hele seksjonen.

I denne oppgaven skal du hjelpe til å designe verdener i et dataspill.

En (spill)verden består av forskjellige rom. I hvert rom finnes det et panel som viser spilleren hvilke andre rom man kan gå til fra det aktuelle rommet.

Eksempelverden

Rom	A	B	C	D	E	F	G	H
Panel	B,C,F	A,H	A,D,E	G	F,G,H	E,H	H	

Fra et rom velger spilleren et nytt rom fra panelet. Når et rom er valgt, blir spilleren gitt en oppgave som må løses for å komme seg til rommet han valgte. Etter oppgaven er løst blir spilleren transportert til det valgte rommet.

Målet i hver verden er å komme frem til et **skattkammer** på kortest mulig tid.
Et rom k er et **skattkammer** hvis og bare hvis det oppfyller følgende krav:

- Det er ikke mulig å nå noen andre rom fra k.
- Det er mulig å komme seg til k fra alle andre rom i verdenen.

I eksempelverdenen over er H et skattkammer.

Hver verden begynner med at spilleren blir plassert i et vilkårlig **startrom**.
Et rom s er et **startrom** hvis og bare hvis det oppfyller følgende krav:

- Det er mulig å komme seg fra s til alle andre rom i verdenen.

I eksempelverdenen over er det tre startrom, nemlig A,B og C.

2(a)

Representasjon

Vi skal modellere en verden som en rettet graf der nodene representerer rommene i verdenen.

1. Hva utgjør kantene i grafen?
2. Hva bestemmer utgraden til en node?
3. Hva er utgraden til et skattkammer?
4. I eksempelverdenen, hva er utgraden til rom D?

Merk: Svarene dine skal gjelde for en generell verden, ikke bare eksempelverdenen. Dette gjelder ikke for punkt (4).

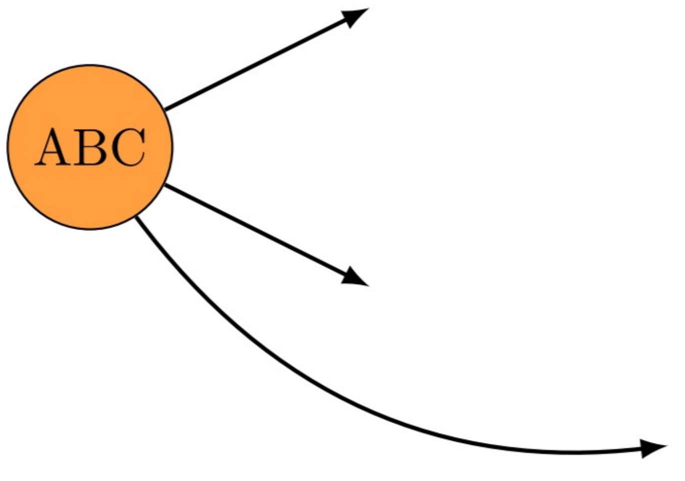
Skriv ditt svar her:

1. kantene i grafen utgjør hvilken rom en kan gå til fra et gitt rom.
2. utgraden til en node bestemmer hvilke rom man kan gå til fra den noden
3. utgraden til et skattekammer er 0
4. utgraden til rom D er 1

Maks poeng: 2

2(b) Komponentgraf

Du blir gitt en verden og representerer den som en rettet graf $G = \langle V, E \rangle$ som i oppgave a.
Vi kan regne ut de sterkt sammenhengende komponentene til G og lage **komponentgrafen** C til G.
Komponentgrafen C er en graf der hver sterkt sammenhengende komponent i G utgjør nodene i C. Det går en rettet kant i C fra en komponent til en annen, hvis det gikk en kant, i G, fra en node i den første komponenten til en node i den andre komponenten.



Du får her en påbegynt tegning av komponentgrafen C til eksempelverdenen.
Legg merke til at {A, B, C} er en node i C, fordi {A, B, C} er en sterkt sammenhengende komponent i G, og at den noden har utgrad 3.
Fullfør tegningen av komponentgrafen til eksempelverdenen.

I denne oppgaven skal du svare med digital håndtegning. Bruk eget skisseark (utdelt). Se instruksjon for utfylling av skisseark i lenken under oppgavelinjen.

Hint: Det kan være lurt (med tanke på denne og de neste oppgavene) å også tegne grafen til eksempelverdenen for din egen del. En slik tegning skal *ikke* leveres inn.

Maks poeng: 3

2(c) Skattkammer I

Du blir gitt en verden og representerer den som en rettet graf $G = \langle V, E \rangle$ som i oppgave a.
La noden $k \in V$ være et skattkammer.
La C_k være den sterkt sammenhengende komponenten til k.
For hvert av spørsmålene under, gi en kort begrunnelse (1-3 setninger).

1. Hvor mange noder inneholder C_k?
2. Hva er utgraden til C_k?

Merk: Svarene dine skal gjelde for en generell verden, ikke bare eksempelverdenen.

Skriv ditt svar her:

1. Ck inneholder en node
2. utgraden til Ck er ukjent

Maks poeng: 2

2(d) Skattkammer II

Forklar kort hvorfor en verden ikke kan inneholde mer enn ett skattkammer.

Merk: Svaret ditt skal gjelde for en generell verden, ikke bare eksempelverdenen.

Skriv ditt svar her:

Den ene regelen for et skattekammer som sier at man skal kunne nå et skattekammer k fra ALLE rom i G gjør det slik at det bare kan være et eller ingen skattekammer. fordi hvis både k1 og k2 er skattekammer sier det at vi skal kunne nå k1 fra k2 og k2 fra k1, men hvis det går an så blir ingen av de et skattekammer fordi det bryter regelen.

Maks poeng: 2

2(e) Startrom

Du blir gitt en verden og representerer den som en rettet graf $G = \langle V, E \rangle$ som i oppgave a.
La noden $s \in V$ være et startrom.
La C_s være den sterkt sammenhengende komponenten til s. Vi skal nå vise at startrommene i verdenen er nøyaktig nodene i C_s.

1. Forklar hvorfor det er slik at hvis $t \in V$ også er i C_s, så må t være et startrom.
2. Vis at alle startrom i verdenen er i C_s.

Merk: Svarene dine skal gjelde for en generell verden, ikke bare eksempelverdenen.

Skriv ditt svar her:

Maks poeng: 4

2(f) Algoritmedesign

En spilldesigner har laget et forslag til en verden og representert den som en rettet graf $G = \langle V, E \rangle$. Du blir bedt om å sjekke om verdenen representert ved G har følgende egenskaper:

- verdenen har nøyaktig ett skattkammer
- verdenen har nøyaktig tre startrom.

Skriv en algoritme som sjekker om en foreslått verden har de to egenskapene over.
For enkelhetsskyld kan du anta at du får inn grafen G som input.
Du kan også anta at du for hver node får en liste over utgående- og innkommende kanter for noden.

Her kan du, om ønskelig, gjenbruke resultater fra de tidligere deloppgavene (selv de du ikke har besvart).
Du kan bruke naturlig språk og/eller pseudokode for å beskrive algoritmen din.
I tillegg er det mulig å gjenbruke algoritmer vi har sett på i IN2010. Hvis du for eksempel ønsker å traversere grafen med dybde-først søk, trenger du ikke å forklare hvordan dybde-først søk fungerer. Hvis du ønsker å benytte en modifisert versjon av en algoritme fra kurset, må modifikasjonene komme tydelig frem.

I IN2010 har vi lagt vekt på effektive algoritmer. I denne oppgaven vil derfor en rask algoritme gi større uttelling enn en mindre rask algoritme.
I neste oppgave blir du bedt om å analysere kjøretiden til algoritmen din.

Merk: Svaret ditt skal gjelde for en generell verden, ikke bare eksempelverdenen.
Skriv algoritmen din her:

Maks poeng: 15

2(g) Analyse

Gi en analyse av kjøretiden til algoritmen din fra deloppgave f ved hjelp av O-notasjon.
Hvis du brukte algoritmer kjent fra kurset trenger du ikke forklare hvorfor de har den kjøretiden de har.

Her vil du få uttelling om analysen din samsvarer med algoritmen du ga, uavhengig av hvor rask algoritmen er.

Merk: Svaret ditt skal gjelde for en generell verden, ikke bare eksempelverdenen.

Skriv ditt svar her:

Maks poeng: 5

2(h) Minimal gjennomføringstid

I denne deloppgaven får du også oppgitt den minimale tiden som trengs for å løse de forskjellige oppgavene.
Panelene består nå av par av rom og positive heltall som gir den minste tiden det er mulig å løse den aktuelle oppgaven på (se PDF til venstre).

Vi ønsker nå å finne den minimale gjennomføringstiden til en verden.
Du skal bruke Dijkstras algoritme til å finne den raskeste mulige måten å komme seg til skattkammeret H, i eksempelverdenen over. Spilleren begynner i rom A.
List opp alle estimatene til skattkammeret underveis i beregningen.
Her skal du altså gi en liste der første element er ∞ og siste element er den korrekte avstanden til skattkammeret.

Skriv ditt svar her:

[«inf», 20, 17, 16]

Maks poeng: 8

2(i) Ukomprimerbare verdener

Dijkstras algoritme bruker, som kjent fra kurset, $O(|E|log(|V|))$ (eventuelt $O(|V|^2)$) tid.
I den siste deloppgaven skal vi se på en spesiell type verden der vi kan regne ut den minimale gjennomføringstiden i lineær tid.
En **ukomprimerbar** verden er en verden slik at hvis vi representerer den som en graf $G = \langle V, E \rangle$, og deretter regner ut komponentgrafen C, får vi at antall noder i G vil være det samme som antall noder i C.
Dette vil være tilfelle hvis alle noder er "alene" i sin sterkt sammenhengende komponent.

Anta at vi blir gitt en ukomprimerbar verden med nøyaktig ett skattkammer og nøyaktig ett startrom og med minimale tider for alle oppgaver.
Forklar kort hvorfor det er mulig å finne den minimale gjennomføringstiden fra startrommet til skattkammeret i lineær ($O(|V| + |E|)$) tid.
Her er det ikke meningen at du skal implementere en algoritme, vi er kun ute etter en kort begrunnelse for hvorfor det er mulig.

Skriv ditt svar her:

Maks poeng: 4

i Introduksjon

En enkel måte å sortere på er å legge elementene som skal sorteres inn i en prioritetskø (PQ) med metoden **add**. Først når alle elementene er lagt inn, tar vi ut ett og ett element ved hjelp av PQs **removeMin**-metode. De vil da komme i stigende orden. I denne seksjonen skal vi se på forskjellige sorteringsstrategier for PQ.

I alle oppgavene legger vi de samme 6 elementene inn i PQ ved hjelp av **add**, slik:
PQ.add(17);
PQ.add(43);
PQ.add(98);
PQ.add(11);
PQ.add(43);
PQ.add(56);

PQ benytter en array som intern struktur. Det du skal finne ut i oppgavene, er hvordan denne arrayen ser ut etter at 56 er satt inn avhengig av hvilken sorteringsmåte oppgaven angir at PQ bruker.

3(a) PQ bruker utplukkssortering

Hvordan vil den indre arrayen se ut etter kallene på **add** hvis PQ benytter utplukkssortering (selection sort) for å finne elementet med høyest prioritet?

Skriv svaret som en kommaseparert liste (som i PDF-en) her:

Maks poeng: 1

3(b) PQ bruker innstikkssortering

Hvordan vil den indre arrayen se ut etter kallene på **add** hvis PQ benytter innstikkssortering for å finne elementet med høyest prioritet?

Skriv svaret som en kommaseparert liste (som i PDF-en) her:

Maks poeng: 1

3(c) PQ bruker BST

PQ benytter nå, istedet for en array, et binært søketre med mindre verdier mot venstre, og større eller like verdier til høyre. Hvordan blir rekkefølgen av tallene hvis vi gjør en BFS-traversering fra rota i dette treet og skriver ut tallene? Vi traverserer venstre barn før høyre barn.

Skriv svaret som en kommaseparert liste (som i PDF-en) her:

Maks poeng: 1

3(d) PQ bruker heapsortering

Hvordan vil den indre arrayen se ut etter kallene på **add** hvis PQ benytter heapsortering (med en *minimumsheap* hvor **add** legger nytt element bakerst og lar dette boble oppover til rett plass) for å finne elementet med høyest prioritet?

Skriv svaret som en kommaseparert liste (som i PDF-en) her:

Maks poeng: 2

3(e) Krav til en heap

Forklar de to kravene som må oppfylles for at et binærtre skal være en minimumsheap.

Skriv ditt svar her:

Maks poeng: 2

3(f)

heapSort-forklar

Starten på algoritmen for heapsortering gjør heltallsarrayen A om til en **maksheap**:

```
public int[] heapSort(int [] A){

    int n = A.length;
    for(int i = n/2; i >= 0; i--){
        downHeap(A,i,n-1);
    }
    ...
    return A;
}
```

Uttrykket $n/2$ er n heltallsidvidert med 2.

Forklar kort hvorfor det er unødvendig å begynne for-løkka på $n-1$, men at det holder å begynne på $n/2$ slik koden gjør.

Skriv ditt svar her:

Maks poeng: 2

3(g) heapSort-kode

Starten på algoritmen for heapsortering gjør heltallsarrayen A om til en **maksheap**:

```
public int[] heapSort(int [] A){

    int n = A.length;
    for(int i = n/2; i >= 0; i--){
        downHeap(A,i,n-1);
    }
    ...
    return A;
}
```

Uttrykket $n/2$ er n heltallsidvidert med 2.

Fullfør koden for heapSort slik at A er sortert i stigende rekkefølge når A returneres.
Her kan du kalle på downHeap, uten å implementere metoden. I neste oppgave blir du bedt om å implementere downHeap slik at heapSort fungerer som den skal.

Javakode:

```
1
```

Maks poeng: 4

3(h)

downHeap-kode

Starten på algoritmen for heapsortering gjør heltallsarrayen A om til en **maksheap**:

```
public int[] heapSort(int [] A){

    int n = A.length;
    for(int i = n/2; i >= 0; i--){
        downHeap(A,i,n-1);
    }
    ...
    return A;
}
```

Uttrykket $n/2$ er n heltallsidvidert med 2.

Skriv javakode for downHeap, slik at heapSort fungerer som den skal.

Javakode:

1

Maks poeng: 12

Document 3
Attached

Vi har en prioritetskø PQ, der den indre datastrukturen er en array. Prioritetskøen har følgende metoder tilgjengelig:

- public void add(E e); // *inserts the specified element into the priority queue*

- public E removeMin(); // *retrieves and removes the head of the queue, or returns null if the queue is empty.*

For enkelhets skyld er elementene representert med et positivt heltall der verdien til tallet er elementets prioritet. Minst verdi gir høyest prioritet i køen.
Vi setter inn 6 verdier i PQ ved å kalle på add i denne rekkefølgen: $17, 43, 98, 11, 43, 56$.

Question 11
Attached

Eksempelverden

Rom	A	B	C	D	E	F	G	H
Panel	B, C, F	A, H	A, D, E	G	F, G, H	E, H	H	

Et rom k er et **skattkammer** hvis og bare hvis det oppfyller følgende krav:

- Det er ikke mulig å nå noen andre rom fra k.
- Det er mulig å komme seg til k fra alle andre rom i verdenen.

Et rom s er et **startrom** hvis og bare hvis det oppfyller følgende krav:

- Det er mulig å komme seg fra s til alle andre rom i verdenen.

Question 12
Attached

Eksempelverden

Rom	A	B	C	D	E	F	G	H
Panel	B, C, F	A, H	A, D, E	G	F, G, H	E, H	H	

Et rom k er et **skattkammer** hvis og bare hvis det oppfyller følgende krav:

- Det er ikke mulig å nå noen andre rom fra k.
- Det er mulig å komme seg til k fra <u>alle andre</u> rom i verdenen.

Et rom s er et **startrom** hvis og bare hvis det oppfyller følgende krav:

- Det er mulig å komme seg fra s til <u>alle andre</u> rom i verdenen.

Question 13
Attached

Eksempelverden

Rom	A	B	C	D	E	F	G	H
Panel	B, C, F	A, H	A, D, E	G	F, G, H	E, H	H	

Et rom k er et **skattkammer** hvis og bare hvis det oppfyller følgende krav:

- Det er ikke mulig å nå noen andre rom fra k.
- Det er mulig å komme seg til k fra alle andre rom i verdenen.

Et rom s er et **startrom** hvis og bare hvis det oppfyller følgende krav:

- Det er mulig å komme seg fra s til alle andre rom i verdenen.

Question 14

Attached

Eksempelverden

Rom	A	B	C	D	E	F	G	H
Panel	B, C, F	A, H	A, D, E	G	F, G, H	E, H	H	

Et rom k er et **skattkammer** hvis og bare hvis det oppfyller følgende krav:

- Det er ikke mulig å nå noen andre rom fra k.
- Det er mulig å komme seg til k fra alle andre rom i verdenen.

Et rom s er et **startrom** hvis og bare hvis det oppfyller følgende krav:

- Det er mulig å komme seg fra s til alle andre rom i verdenen.

Question 15

Attached

Eksempelverden

Rom	A	B	C	D	E	F	G	H
Panel	B, C, F	A, H	A, D, E	G	F, G, H	E, H	H	

Et rom k er et **skattkammer** hvis og bare hvis det oppfyller følgende krav:

- Det er ikke mulig å nå noen andre rom fra k.
- Det er mulig å komme seg til k fra <u>alle andre</u> rom i verdenen.

Et rom s er et **startrom** hvis og bare hvis det oppfyller følgende krav:

- Det er mulig å komme seg fra s til <u>alle andre</u> rom i verdenen.

Question 16
Attached

Eksempelverden

Rom	A	B	C	D	E	F	G	H
Panel	B, C, F	A, H	A, D, E	G	F, G, H	E, H	H	

Et rom k er et **skattkammer** hvis og bare hvis det oppfyller følgende krav:

- Det er ikke mulig å nå noen andre rom fra k.
- Det er mulig å komme seg til k fra alle andre rom i verdenen.

Et rom s er et **startrom** hvis og bare hvis det oppfyller følgende krav:

- Det er mulig å komme seg fra s til alle andre rom i verdenen.

Question 17
Attached

Eksempelverden

Rom	A	B	C	D	E	F	G	H
Panel	B, C, F	A, H	A, D, E	G	F, G, H	E, H	H	

Et rom k er et **skattkammer** hvis og bare hvis det oppfyller følgende krav:

- Det er ikke mulig å nå noen andre rom fra k.
- Det er mulig å komme seg til k fra alle andre rom i verdenen.

Et rom s er et **startrom** hvis og bare hvis det oppfyller følgende krav:

- Det er mulig å komme seg fra s til alle andre rom i verdenen.

Question 18

Attached

Eksempelverden med minimale tider

Rom	A	B	C	D	E	F	G	H
Panel	(B;1), (C;3), (F;5)	(A;1), (H;19)	(A;2), (D;1), (E;16)	(G;14)	(F;1), (G;1), (H;11)	(E;2), (H;12)	(H;8)	

Et rom k er et **skattkammer** hvis og bare hvis det oppfyller følgende krav:

- Det er ikke mulig å nå noen andre rom fra k.
- Det er mulig å komme seg til k fra alle andre rom i verdenen.

Et rom s er et **startrom** hvis og bare hvis det oppfyller følgende krav:

- Det er mulig å komme seg fra s til alle andre rom i verdenen.

Question 19
Attached

Eksempelverden med minimale tider

Rom	A	B	C	D	E	F	G	H
Panel	(B;1), (C;3), (F;5)	(A;1), (H;19)	(A;2), (D;1), (E;16)	(G;14)	(F;1), (G;1), (H;11)	(E;2), (H;12)	(H;8)	

Et rom k er et **skattkammer** hvis og bare hvis det oppfyller følgende krav:

- Det er ikke mulig å nå noen andre rom fra k.
- Det er mulig å komme seg til k fra alle andre rom i verdenen.

Et rom s er et **startrom** hvis og bare hvis det oppfyller følgende krav:

- Det er mulig å komme seg fra s til alle andre rom i verdenen.

Question 20
Attached

Vi har en prioritetskø PQ, der den indre datastrukturen er en array. Prioritetskøen har følgende metoder tilgjengelig:

- public void add(E e); // *inserts the specified element into the priority queue*
- public E removeMin(); // *retrieves and removes the head of the queue, or returns null if the queue is empty.*

For enkelhets skyld er elementene representert med et positivt heltall der verdien til tallet er elementets prioritet. Minst verdi gir høyest prioritet i køen.
Vi setter inn 6 verdier i PQ ved å kalle på add i denne rekkefølgen: $17, 43, 98, 11, 43, 56$.

Question 21
Attached

Vi har en prioritetskø PQ, der den indre datastrukturen er en array. Prioritetskøen har følgende metoder tilgjengelig:

- public void add(E e); // *inserts the specified element into the priority queue*
- public E removeMin(); // *retrieves and removes the head of the queue, or returns null if the queue is empty.*

For enkelhets skyld er elementene representert med et positivt heltall der verdien til tallet er elementets prioritet. Minst verdi gir høyest prioritet i køen.
Vi setter inn 6 verdier i PQ ved å kalle på add i denne rekkefølgen: $17, 43, 98, 11, 43, 56$.

Question 22

Attached

Vi har en prioritetskø PQ, der den indre datastrukturen er en array. Prioritetskøen har følgende metoder tilgjengelig:

- public void add(E e); // *inserts the specified element into the priority queue*
- public E removeMin(); // *retrieves and removes the head of the queue, or returns null if the queue is empty.*

For enkelhets skyld er elementene representert med et positivt heltall der verdien til tallet er elementets prioritet. Minst verdi gir høyest prioritet i køen.
Vi setter inn 6 verdier i PQ ved å kalle på add i denne rekkefølgen: $17, 43, 98, 11, 43, 56$.

Question 23

Attached

Vi har en prioritetskø PQ, der den indre datastrukturen er en array.
Prioritetskøen har følgende metoder tilgjengelig:

- public void add(E e); // *inserts the specified element into the priority queue*
- public E removeMin(); // *retrieves and removes the head of the queue, or returns null if the queue is empty.*

For enkelhets skyld er elementene representert med et positivt heltall der verdien til tallet er elementets prioritet. Minst verdi gir høyest prioritet i køen.
Vi setter inn 6 verdier i PQ ved å kalle på add i denne rekkefølgen: $17, 43, 98, 11, 43, 56$.