i

Info

Eksamen IN3130 og INF4130 høsten 2019

Tid

25. november kl. 14:30-18:30
Faglærerne vil gå en runde fra ca kl 15.30.

Oppgavene

Til alle oppgavene kan du supplere svaret med en tegning. Du skal tegne på spesialark som du får utdelt. Les informasjonen om håndtegning i lenken under oppgavelinjen. Fyll ut informasjonen på arket med én gang -- du vil ikke få ekstratid til dette når eksamen er over.

Tillatte hjelpemidler

Alle trykte og skrevne hjelpemidler er tillatt.

1(a)

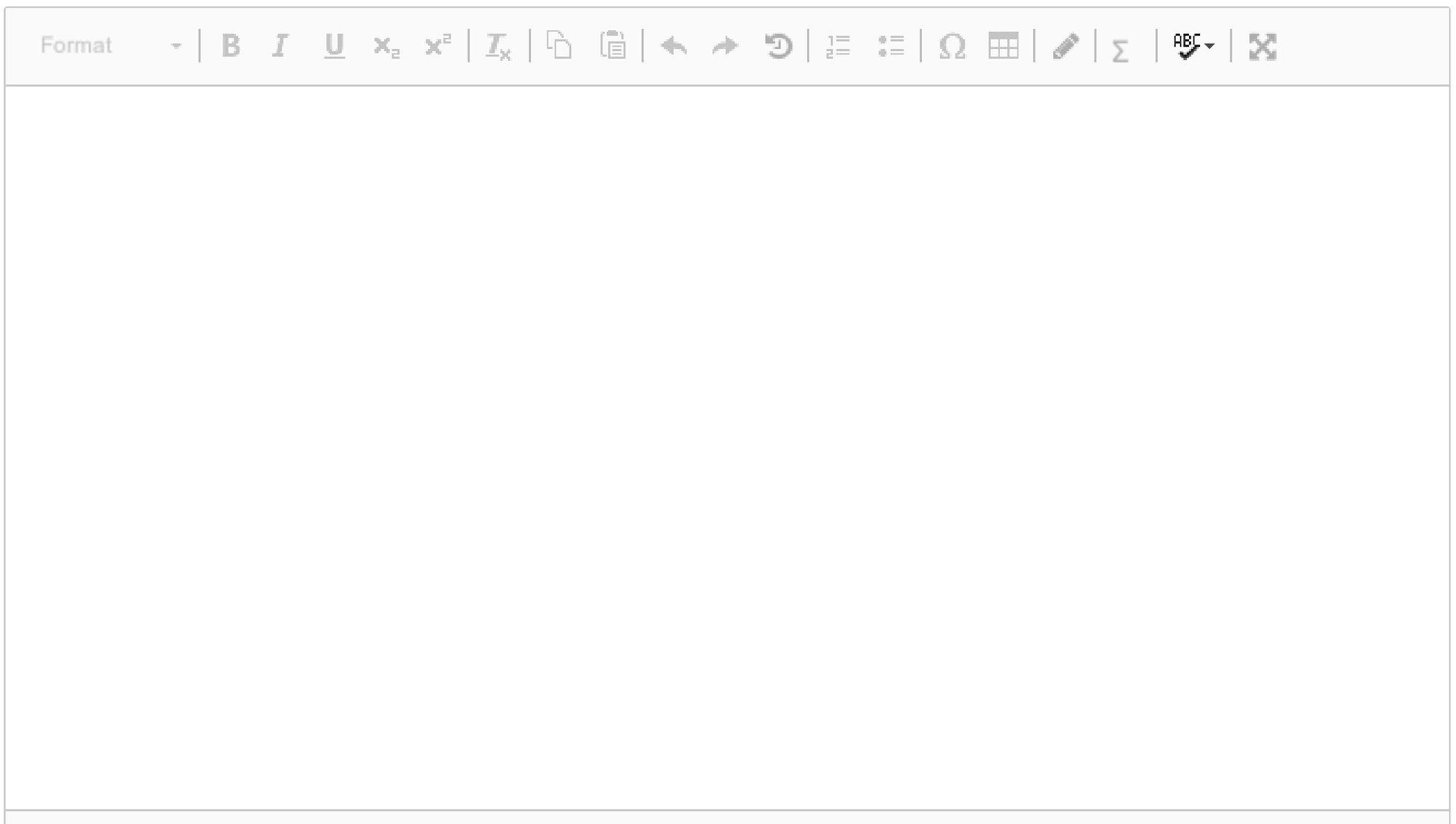
Flyt i nettverk (a)

Vi har en rettet graf *G*, og *A* og *B* er to (forskjellige) noder i *G*. Vi sier at et sett (en mengde) av rettede veier fra *A* til *B* i *G* er *kant-uavhengig* dersom to veier i settet aldri bruker samme kant (men de kan godt gå gjennom samme node).

Du skal beskrive hvordan man kan finne det maksimale antall veier man kan ha i et kant-uavhengig sett av veier fra A til B. Du skal altså ikke finne selve veiene, men bare antallet. Når du beskriver din løsningsmetode kan du henvise til algoritmer som er kjent fra pensum, og bare angi hvilke data algoritmen(e) skal arbeide på. Forklar.

Hint: Overskriften på oppgaven er *Flyt i nettverk*

Skriv ditt svar her:



Maks poeng: 6

1(b) Flyt i nettverk (b)

(fortsatt fra (a))
Vi sier nå at et sett med rettede veier fra *A* til *B* er *node-uavhengig* dersom to veier i settet aldri går gjennom samme node (bortsett fra at alle starter i *A* og ender i *B*)

Du skal beskrive hvordan man kan finne det maksimale antall veier man kan ha i et node-uavhengig sett av veier fra *A* til *B*. Du skal altså ikke finne selve veiene, men bare antallet. Når du beskriver din løsningsmetode kan du henvise til algoritmer som er kjent fra pensum, og bare angi hvilke data algoritmen(e) skal arbeide på. Forklar kort.

Skriv ditt svar her:

Format | B I U x_2 x^2 T_x | Ω | ∑ | ABC

Words: 0

Maks poeng: 6

2(a) Uavgjørbarhet

Er følgende språk uavgjørbart? Gi et bevis.

L = { M | M er koden til en Turingmaskin som avgjør HAMILTONICITY }

Skriv svaret her:

Format | B I U x_2 x^2 | I_x | | | | | Ω | | | ∑ | ABC | |

Words: 0

Maks poeng: 7

2(b) NP-kompletthet

På forelesningene så vi at problemet PARTITION (gitt en mengde *A* og et heltall "størrelse" *s*(*a*) for hvert element *a* i *A*; avgjør om *A* kan deles i to delmengder slik at summen av størrelsene for hver delmengde er lik) er NP-komplett. Det er imidlertid ikke noe spesielt med tallet 2 her.

(1) Vis at 3-PARTITION (oppdeling i tre delmengder med lik sum) er NP-komplett.

(2) Generaliser beviset til å gjelde for *k*-PARTITION for enhver konstant *k*.

Vi kan imidlertid ikke generalisere uendelig. Det er opplagt at *n*-PARTITION, hvor *n* er antall elementer i *A*, er avgjørbart i polynomisk tid (alle elementene må opplagt ha samme størrelse).

(3) Hva med (*n*-1)-PARTITION? Eller (*n*-*k*)-PARTITION, hvor *k* er en konstant. Bevis svarene dine.

Skriv svaret her:

Format

Words: 0

Maks poeng: 8

2(c) Kjernekonsepter og forståelse

For hver påstand, indiker om den er korrekt med «JA» eller «NEI», og gi en kort forklaring.

1. En Turingmaskin kan beregne enhver funksjon som kan beregnes på en moderne PC.
2. Noen uavgjørbare problemer kan løses ved å bruke en parallelldatamaskin.
3. Vi modellerer problemer som formelle språk for å kunne dele dem inn i klasser.
4. Hvert problem, eller formelle språk, har et "komplement". For eksempel er NON-HAMILTONICITY problemet å avgjøre om en gitt graf ikke er hamiltonsk. Hvis et problem er NP-komplett, så er komplementet også NP-komplett.
5. Noen NP-komplette problemer har polynomisk gjennomsnittstidskompleksitet.

Skriv svaret her:

Format

Words: 0

Maks poeng: 7

3(a)

Dyn. progr. (a)

Vi har gitt et alfabet *A* (f.eks. alle norske bokstaver), samt en mengde *M* med ikketomme "ord" over dette alfabetet. Vi får så gitt en "lang" streng *S* over det samme alfabetet, og problemstillingen er å avgjøre om S kan deles opp i mindre biter slik at hver bit er et ord i mengden *M*. Symbolene (eller bokstavene) i S er nummerert fra 1 til n.

Vi skal løse dette med dynamisk programmering, og må derfor tenke på hva slags tabell som er egnet til å lagre svar på delproblemer. Foreslå en tabell for dette, og angi hva betydningen av dataene i denne skal være i den delen som er fylt ut.

Hint: Dette er nokså rett fram. Les resten av oppgaven før du svarer.

Skriv ditt svar her:

Format

Words: 0

Maks poeng: 5

3(b) Dyn. Progr. (b)

(fortsatt fra (a))

Angi med en programskisse hvordan du vil fylle ut tabellen, og hva du vil gjøre i hvert skritt under utfyllingen. Bruk navnet *T* på tabellen i programskissen. Vi antar at mengden av ord i *M* er vesentlig større enn lengden av *S*, og at vi har et tabellverk over ordene i mengden *M*. For oppslag i denne finnes en metode

boolean isInM(String R) { ... }

som besvarer om *R* er med i mengden *M* eller ikke.

En sammenhengende substreng (en "bit") av *S* fra og med indeks *i* til og med indeks *j* kan du angi som S[i:j]. Husk å angi hvordan man finner svaret på problemet etter at tabellen er ferdig utfylt! Forklar.

Vurder også om man kan initialisere tabellen på passelig måte for å få færrest mulig tester inne i løkker i programmet.

Skriv ditt svar her:

Format

Words: 0

Maks poeng: 6

3(c) # Dyn. Progr. (c)

(fortsatt fra (b))
Vi antar at et oppslag i *M*-tabellen tar tid $O(|R|)$, der $|R|$ er antall bokstaver i strengen *R*. Av hvilken orden vil tidsbruken til algoritmen fra (b) være, uttrykt i *n*? Forklar.

Skriv ditt svar her:

Format

Words: 0

Maks poeng: 5

3(d) # Dyn. Progr. (d)

(fortsatt fra (c))
Vi stiller nå det ekstra kravet at oppdelingen av *S* i biter skal ha så få biter som mulig. Svaret skal rett og slett være det minste antall biter man kan klare seg med når hver bit skal være i *M*. Om *S* ikke kan deles opp i biter der hver bit er i *M,* så skal svaret være 0. Angi hva slags tabell du nå vil bruke, og angi med en programskisse hvordan du vil fylle den ut for å finne dette antallet. Forklar.

Skriv ditt svar her:

Format

Words: 0

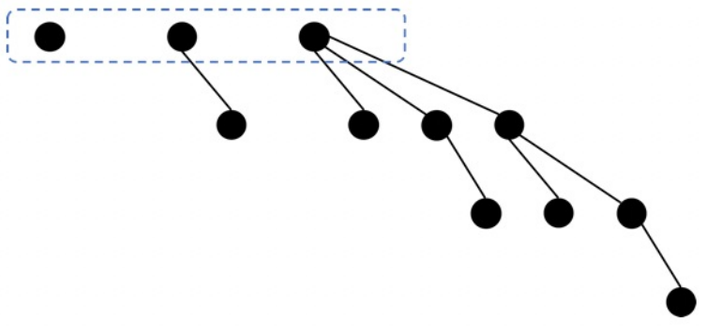
Maks poeng: 6

4(a) # Prioritestkøer (a)

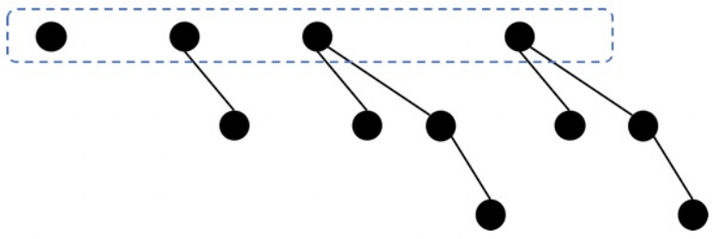
I deloppgavene her skal du identifisere hvilke av et antall skoger som er lovlige og hvilke som er ulovlige binomial-heaper (deloppgave (a)) og Fibonacci-heaper (deloppgave (b)). Flere valg kan være lovlige i hver deloppgave.

Hvilke av de fire skogene i figuren under er lovlige binomial-heaper og hvilke ulovlige? Begrunn kort.

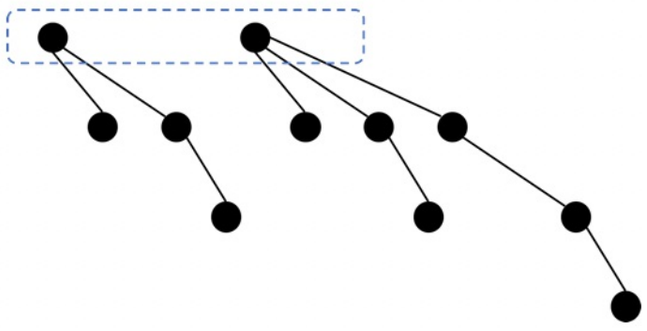
i.



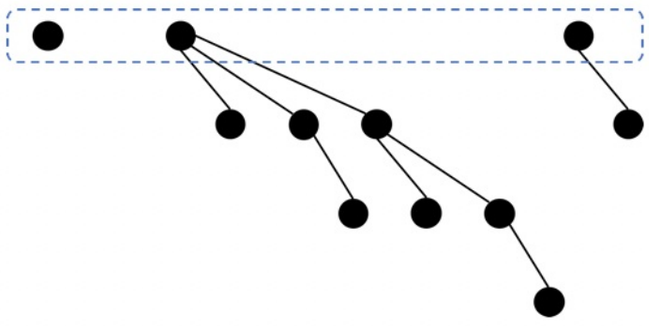
ii.



iii.



iv.



Skriv svaret her:

Format | B I U x_2 x^2 T_x | ABC | Ω | Σ

Words: 0

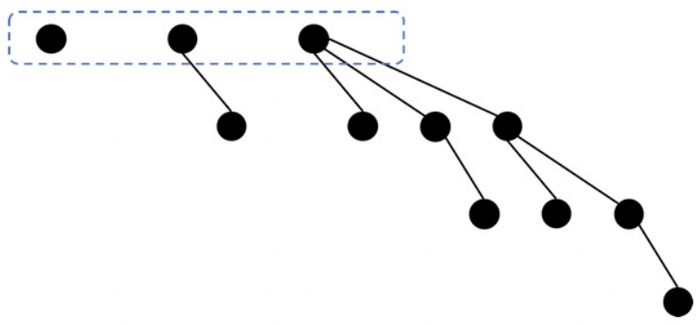
Maks poeng: 7

4(b) Prioritestkøer (b)

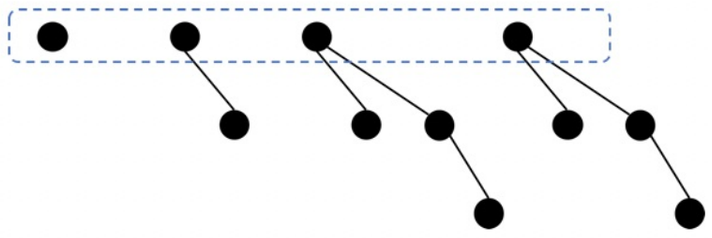
(fortsatt fra (a))

Hvilke av de fire skogene i figuren under er lovlige Fibonacci-heaper og hvilke ulovlige? Begrunn kort.

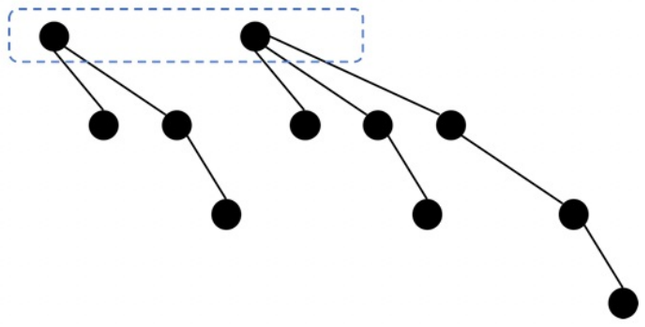
i.



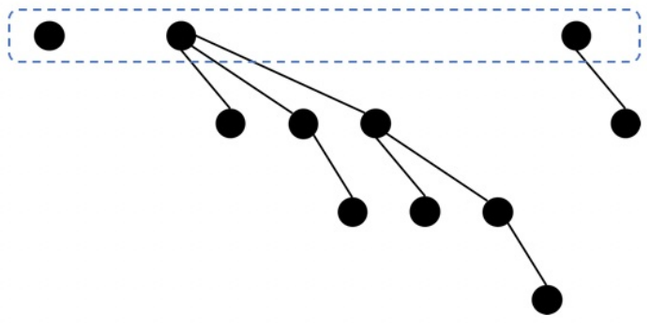
ii.



iii.



iv.



Skriv svaret her:

Format

Words: 0

Maks poeng: 7

5(a)

A* og heuristikker (a)

Besvar følgende, kort:

i. Hva vil det si at en heuristikk er monoton?
ii. Hva er det vi oppnår/unngår med en monoton heuristikk i A*-algoritmen?

Skriv svaret her:

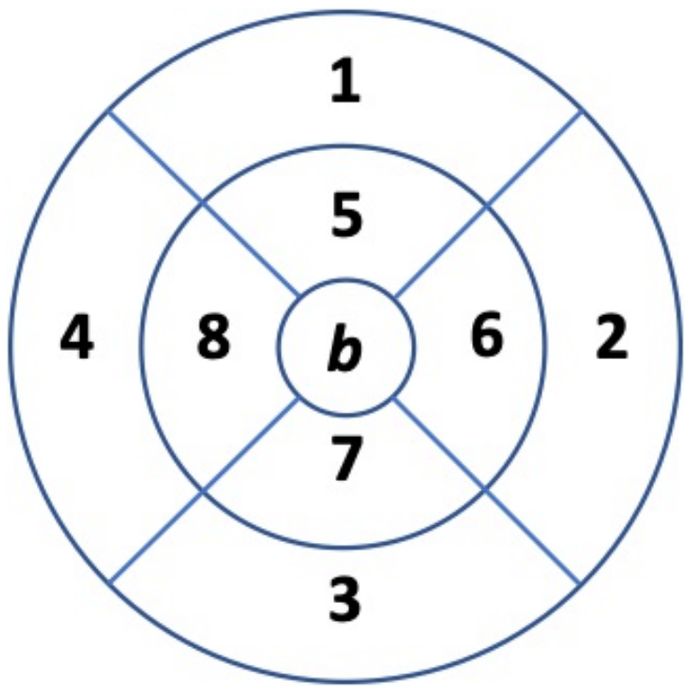
Format | B I U x_2 x^2 I_x | ABC | Ω | ∑

Words: 0

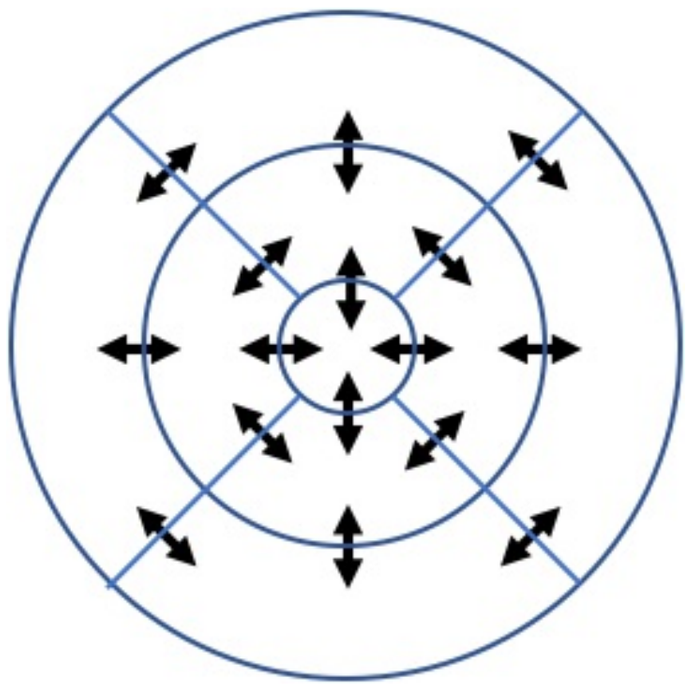
Maks poeng: 6

5(b) A* og heuristikker (b)

Vi er gitt en rund variant av 8-spillet. Figur 1 viser den ønskede slutt-tilstanden for spillet, med det tomme feltet *b* («hullet») i midten.



Figur 1. Slutt-tilstanden for det runde 8-spillet.



Figur 2. Mulige flytt av det tomme feltet («hullet») i det runde 8-spillet.

Som i den obligatoriske oppgaven, er det enklest å modellere de mulige trekkene vi kan gjøre ved å se på hvordan det tomme feltet («hullet») flyttes. Figur 2 viser de mulige måtene å flytte det tomme feltet på for alle rutene på brettet. (Legg merke til at vi kun flytter en og en brikke, vi vrir ikke hele ringer i ett flytt.)

Besvar følgende: Er heuristikken *h* nedenfor monoton? Begrunn svaret kort.

h = antall feilplasserte brikker; vi teller ikke med «hullet»

Skriv svaret her:

Format

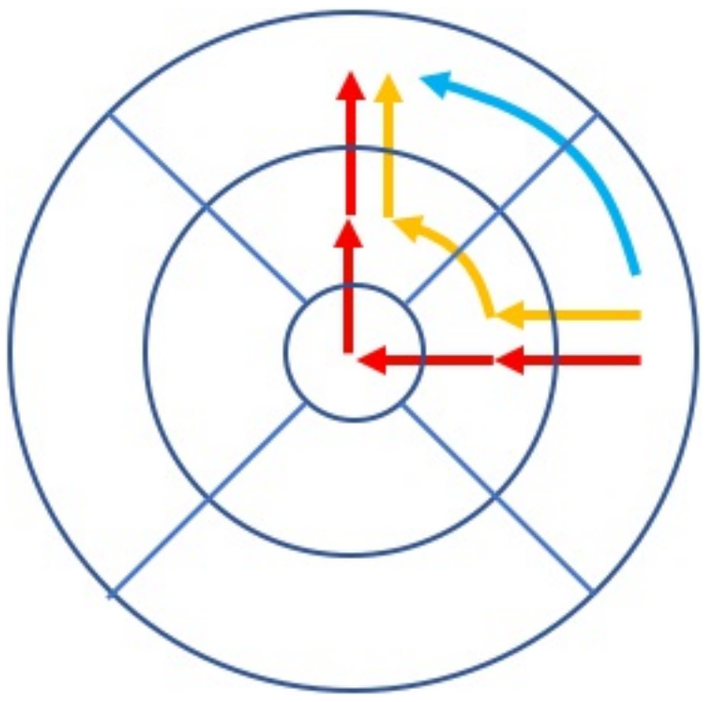
Words: 0

Maks poeng: 6

5(c) A* og heuristikker (c)

(fortsatt fra (b))

I den runde varianten av 8-spillet er ikke avstanden mellom to ruter helt entydig. Figur 3 viser tre mulige stier man kan flytte hullet langs, alle med forskjellig lengde. (Den røde stien tilsvarer å flytte langs en Manhattan-sti. Den blå stien vil på mange måter tilsvare et diagonalt flytt i det tradisjonelle, firkantede 8-spillet.)



Figur 3. Tre mulige mål på avstanden mellom to ruter på brettet.

I det tradisjonelle, firkantede 8-spillet kan vi bruke summen av Manhattan-avstandene mellom brikkenes nåværende posisjon og riktige posisjon som heuristikk.

Besvar følgende:

i. Beskriv (med ord) et egnet mål på et avstandsmål mellom to ruter på det runde brettet, som lar deg bruke summen av disse avstandene mellom brikkenes nåværende posisjon og riktige posisjon som heuristikk for A*-algoritmen på det runde 8-spillet.
ii. Begrunn hvorfor ditt valgte avstandsmål vil gi en monoton heuristikk.

Skriv svaret her:

Format

Words: 0

Maks poeng: 6

6(a)

Strengsøk (a)

Vi søker etter patternet **ananas** med ulike strengsøke-algoritmer. Anta at alfabetet vårt består av de vanlige norske små bokstavene.

For hver mulige indeks for mismatch i patternet vis hvor langt shift vi får med KnuthMorrisPratt-algoritmen. Indiker overlappende prefikser og suffikser.

Skriv svaret her:

Format

Words: 0

Maks poeng: 6

6(b)

Strengsøk (b)

Vi søker etter patternet **ananas** med ulike strengsøke-algoritmer. Anta at alfabetet vårt består av de vanlige norske små bokstavene.

Beregn shift-verdiene vi får med patternet i Hoorspool-algoritmen som vi så på i kurset (den forenklede BoyerMoore-algoritmen).

Skriv svaret her:

Format

Words: 0

Maks poeng: 6