



UNIVERSITETET I BERGEN

KANDIDAT

430

PRØVE

INF102 0 Algoritmer, datastrukturer og programmering

Emnekode	INF102
Vurderingsform	Skriftlig eksamen
Starttid	20.11.2024 09:00
Sluttid	20.11.2024 12:00
Sensurfrist	--
PDF opprettet	04.01.2025 16:00

Flervalgsoppgaver				
Oppgave	Tittel	Status	Poeng	Oppgavetype
i	Egenerklæring			Informasjon eller ressurser
i	Info om eksamen H23			Informasjon eller ressurser
1	kjøretid generateEven	Riktig	3/3	Flervalg
2	kjøretid generateHigh	Riktig	3/3	Flervalg
3	kjøretid quadruple	Feil	0/3	Flervalg
4	Gjett tallet	Riktig	3/3	Fyll inn tall
5	Velg datastruktur for største element	Riktig	3/3	Flervalg
6	Korteste sti	Riktig	3/3	Flervalg
7	Brøyting	Riktig	3/3	Flervalg
8	Velg datastruktur for iterate sorted	Riktig	3/3	Flervalg
9	Doubling ratio	Riktig	3/3	Flervalg
10	Flyrute	Riktig	3/3	Flervalg
Langsvarsoppgaver				
Oppgave	Tittel	Status	Poeng	Oppgavetype
11	Save Rudolph 2.0	Besvart	12/15	Langsvar
12	Unique Sequence	Besvart	20/20	Programmering

Seksjon 3

Oppgave	Tittel	Status	Poeng	Oppgavetype
13	Poeng fra semesteroppgaver H24	Besvart	34/35	Tekstfelt

1 kjøretid generateEven

Hva er kjøretiden til denne metoden?

```
public static ArrayList<Integer> generateEven(int n){
    ArrayList<Integer> numbers = new ArrayList<>();
    for(int i=0; i<n; i++) {
        if(i%2==0) {
            numbers.add(i);
        }
    }
    return numbers;
}
```

Velg ett alternativ:

- ☐ $O(1)$
- ☐ $O(\log(n))$
- ☒ $O(n)$
- ☐ $O(n \log n)$
- ☐ $O(n^2)$
- ☐ $O(n^2 \log n)$
- ☐ $O(n^3)$



Maks poeng: 3

2 kjøretid generateHigh

Hva er kjøretiden til denne metoden?

```
public static LinkedList<Integer> generateHigh(int n){  
    LinkedList<Integer> numbers = new LinkedList<>();  
    for(int i=n; i<n+10; i++) {  
        numbers.add(i);  
    }  
    return numbers;  
}
```

Velg ett alternativ:

- ☒ $O(1)$
- ☐ $O(\log(n))$
- ☐ $O(n)$
- ☐ $O(n \log n)$
- ☐ $O(n^2)$
- ☐ $O(n^2 \log n)$
- ☐ $O(n^3)$



Maks poeng: 3

3 kjøretid quadruple

Hva er kjøretiden til denne metoden?

n er antall tall i numbers.

```
public static boolean quadruple(ArrayList<Integer> numbers, long target) {
    ArrayList<Long> products = new ArrayList<>();
    for(long i : numbers) {
        for(long j : numbers) {
            products.add(i*j);
        }
    }
    Collections.sort(products);
    int low=0;
    int hi=products.size()-1;
    while(low<=hi) {
        long num = products.get(low)+products.get(hi);
        if(num==target) {
            System.out.println(products.get(low)+" + "+products.get(hi));
            return true;
        }
        if(num>target) {
            hi--;
        }else {
            low++;
        }
    }
    return false;
}
```

Velg ett alternativ:

- ☐ $O(1)$
- ☐ $O(\log(n))$
- ☐ $O(n)$
- ☐ $O(n \log n)$
- ☒ $O(n^2)$
- ☐ $O(n^2 \log n)$
- ☐ $O(n^3)$



Maks poeng: 3

4 Gjett tallet

På lab 3 var oppgave 3 å gjette et tall mellom 0 og n .

Når vi kjører den optimale løsningen med $n = 1000$ brukte den maks 10 gjett for å finne tallet.

Hvor mange gjett er det meste den optimale løsningen vil bruke når $n=16000$?

Skriv et nummer mellom 0 og 16 000:



Maks poeng: 3

5 Velg datastruktur for største element

Hvilken Datastruktur gir best kjøretid hvis du trenger følgende operasjoner:

- Legg til et element

- finn og fjern største element

Du kan anta at hver operasjon skal gjøres ca. like mange ganger.

Velg ett alternativ:

☒ PriorityQueue

☐ HashSet

☐ ArrayList

☐ Sortert liste

☐ LinkedList

☐ Union-Find



Maks poeng: 3

6 Korteste sti

Du trenger å finne korteste sti i en graf der det er noen negative vekter, men ingen negative sykler. Hvilken algoritme vil gi deg den beste kjøretiden?

Velg ett alternativ:

- ☐ MST (Prim's / Kruskal's)
- ☐ Dijkstra's
- ☐ BFS
- ☐ DFS
- ☒ Bellman-Ford
- ☐ A*
- ☐ LCA (Least common ancestor)




Maks poeng: 3

7 Brøyting

Et land med mye snø skal brøyte veiene sine, de har et stort veinett som kobler sammen byer. For å spare penger har de bestemt å ikke brøyte alle veiene, men velge noen veier slik at inbyggerene kan kjøre fra en hvilken som helst by til en hvilken som helst annen by, kun ved å kjøre på de brøytete veiene. Hvilken algoritme ville du valgt?

Velg ett alternativ:

- ☐ A*
- ☒ MST (Prim's / Kruskal's) 
- ☐ Dijkstra's
- ☐ Bellman-Ford
- ☐ LCA (Least common ancestor)
- ☐ DFS
- ☐ BFS

Maks poeng: 3

8 Velg datastruktur for iterate sorted

Hvilken Datastruktur gir best kjøretid hvis du trenger følgende operasjoner:

- Legg til et element

- fjern et element

- Iterer igjennom i sortert rekkefølge

(Du kan anta at legg til og fjern skjer $O(n)$ ganger mens iterer igjennom skjer $O(\log n)$ ganger)

Velg ett alternativ:

- ☐ LinkedList
- ☐ ArrayList
- ☐ Sortert liste
- ☐ HashSet
- ☐ Heap (PriorityQueue)
- ☐ Union-Find
- ☒ Binært søketree (TreeSet)



Maks poeng: 3

9 Doubling ratio

Du har et program som du testet på input av størrelse n , programmet tok ca. 2 minutter.

Når input størrelsen økte til $3n$ så tok programmet ca. 1 time.

Hva er mest sannsynlig at kjøretiden til koden er?

Velg ett alternativ:

☐ $O(1)$

☐ $O(n^5)$

☐ $O(n)$

☐ $O(n^4)$

☐ $O(n^2)$

☐ $O(n^6)$

☒ $O(n^3)$



Maks poeng: 3

10 Flyrute

Du skal lage en app for flybiletter og trenger å finne korteste flyreise fra et sted til et annet. Hvilken algoritme er best å bruke?

Velg ett alternativ:

- ☐ Bellman-Ford
- ☐ BFS
- ☐ Dijkstra's
- ☐ DFS
- ☐ LCA (Least common ancestor)
- ☒ A*
- ☐ MST (Prim's / Kruskal's)



Maks poeng: 3

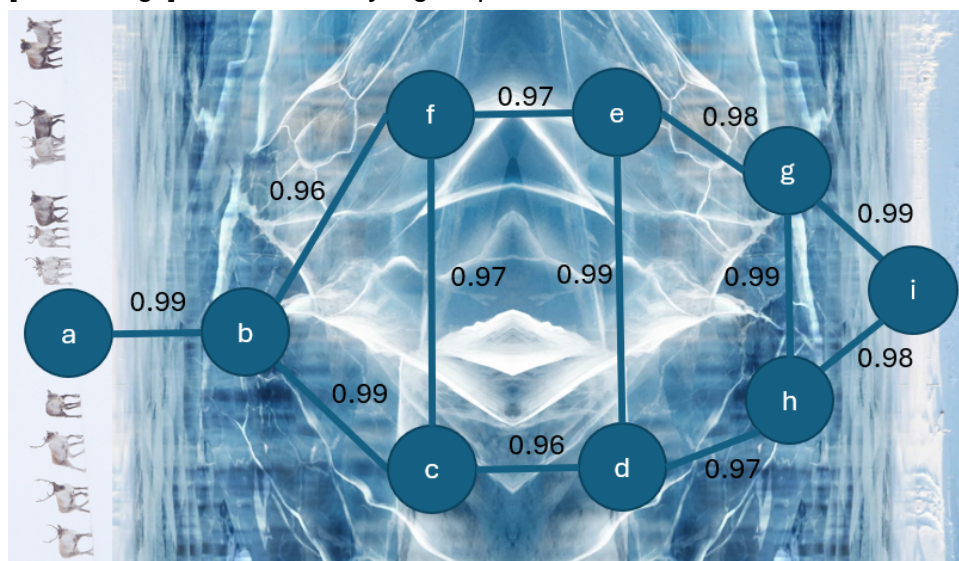
11 Save Rudolph 2.0

En gruppe samer i nordlige russland trenger å lede reinsdyrene sine trygt over isen. På grunn av klimaendringer er isen ekstra tynn i år.

De har modelert isen som en graf, og for hver kant har de beregnet sannsynligheten for at alle reinsdyrene kommer seg trygt over.

Du skal beskrive en algoritme som finner en sti fra en start node **a** til en sluttnode **b** slik at reinsdyrene er tryggest mulig (størst mulig sannsynlighet for at alle kommer seg over isen). Når man regner sannsynligheten for at alle reinsdyrene kommer frem ganger man sammen sannsynlighetene.

I eksempelet under blir tryggeste sti fra a til i:
[a,b,c,d,e,g,i] med en sannsynlighet på ca. 0.904



Skriv ditt svar her

Her tenker jeg å bruke en justert versjon av Dijkstras algoritme, hvor vi finner den tyngste ruten mulig da dette vil være den tryggeste veien.

Dette blir en kjøretid på $O(E \log V)$, E = edge, V = vertex.

I vår Dijkstras vil vi bruke en Max-Heap som Priority Que (pq) og $graph[v]$ er sjansen for å komme til den vertexen fra start, dette kan vi gjøre med å reversere Comaratoren som er Default en Min-heap. For å holde kontroll på stien kan må vi ta vare på parent vertexene til en hver vertex. Dette kan vi gjøre med et hashMap som vi kaller fromV.

Algoritmen vår vil fungerer med å sette $graph[start]$ til 1 for å indikerer 100% sjanse for å komme dit og legge start til i vår pq. Så for alle andre

vertices v setter vi $graph[v]$ til 0 for å vise en 0 prosent sjanse (så langt). Dijkstra vil nå poll neste vertex fra pq og sjekke om sjansen for å komme til den vertex v er bedre enn det vi allerede har. Om den er det så oppdaterer vi $graph[v]$ til den nye sjansen og $fromV$ til den nye parent. Så legger den til i pq sjansene for å komme til naboene. Når vi legger til nabo i pq må vi holde kontroll på at sjansen blir sjanse for å komme til naboen blir sjansen til å komme til før naboen \times edge weight til naboen, for å få sjansen fra start vertex. Sånn fortsetter vi til PQ er tom, og da vet vi at vi har beste vei til en hver node. Da kan vi bruke $fromV$ til å gå bakover helt til vi kommer til at parent er Null, siden det vil indikerer start vertexen.

Ord: 290

Maks poeng: 15

12 Unique Sequence

På denne oppgaven skal du skrive et program som tar som input en liste med tall. Du skal finne det lengste intervallet i listen der alle tallene er forskjellige.

For eksempel:

Input: [1, 4, 7, 3, 4, 5, 2, 1, 7]

Output: 6

Intervallet er fra index 2 til 7 inneholder [7, 3, 4, 5, 2, 1] og alle disse er forskjellige. Dette er det lengste mulige intervallet og inneholder 6 elementer, derfor blir svaret 6.

Vi har laget en main fil der du kan teste løsningen din og en ferdig implementasjon av en brute force løsning. Du skal prøve å gjøre koden raskere og vil få poeng etter hvor rask løsningen din er. Både effektiv algoritme og god Java kode kreves for full score.

Du finner koden her: [SequenceOfUniqueElements.java](#)

Husk å kopiere koden din inn i vinduet under.

Skriv ditt svar her

```
1 klarer ikke copy paste så må skrive inn, kan være små syntax/skrive feil derfor,
2 Tanken var å bruke et hashMap for å holde kontrol på siste index til et hvert tall
  . Vi vet da om den indexen er bak starten til den nåværende sekvensen så skal
  vi se bort ifra den og oppdatere hashMapet. Siden alle hashMap operasjonene
  er  $O(1)$  så blir runtime på  $O(n)$ . Jeg har også en  $O(n^2)$  løsning hvis denne
  ikke fungerer skikkelig.
3
4
5 //O(n)
6 public static int findLongestUnique(ArrayList<Integer> numbers){
7     Integer best = 0;
8     Integer start = 0;
9     Integer end = 1;
10    HashMap<Integer,Integer> valueIndex = new HashMap<>();
11
12    //setter inn startverdien
13    valueIndex.put(numbers.get(start),start);
14
15    //looper gjennom hvert element i listen
16    while(end < numbers.size()){
17        //henter ut det elementet vi skal se på
18        Integer current = numbers.get(end);
19
20        //ser om valueIndex har denne nøkkelen
21        if(valueIndex.containsKey(current)){
22
23            //hvis verdien (indexen) er før start så er ikke den del av denne
24            //sekvensen og vi oppdaterer valueIndex
25            if(valueIndex.get(current) < start){
26                valueIndex.put(current,end);
27            }else{
```

Maks poeng: 20

13 Poeng fra semesteroppgaver H24

På denne oppgaven trenger dere ikke gjøre noe, her får dere poeng for insatsen dere har gjort på semesteroppgaver og ukesoppgaver.

Opp til 5 poeng for ukesoppgaver

Opp til 15 poeng for semesteroppgave 1

Opp til 15 poeng for semesteroppgave 2

Det var også mulig å få 2 bonuspoeng på semesteroppgave 1 som også vil telle med her.

Du kan maks få 35 poeng.

God jul.

Skriv en hyggelig melding til foreleser!

Takk for morsomt semester.

./Thomas

Maks poeng: 35