Seksjon 1

1 OPPGAVE

1 (25%).

Skriv (i Java plus standard library, eller i pseudokode) en metode

int threeSumPos(int[] a)

som returnerer antallet tripler *i,j,k* med *0 <= i < j < k < a.length* slik at *a[i] + a[j] + a[k] > 0*. Du kan anta at alle elementer i tabellen *a* er forskjellige. For å oppnå full skåre bør din metode kjøre i *O(N^2 log N)* tid, der *N = a.length*.

Fill in your answer here

2 OPPGAVE

2 (25%).

Anta at følgende metode er gitt:
// precondition: 0<=lo<=mid<hi<=a.length, and a[lo..mid] and a[mid+1..hi] are sorted
void merge(Comparable[] a, int lo, int mid, int hi)
// postcondition: a[lo..hi] is sorted
Skriv en metode (i Java eller pseudokode, uten å bruke metoder fra biblioteket)

void sort(Comparable[] a, int lo, int hi*)*

som sorterer *a[lo..hi]*. For å oppnå full skåre bør din metode kjøre i tid *O(N log N)*, der *N = a.length*. Du kan anta at *merge* kjører i tid *~6(hi-lo)*. Bonusspørsmål (10% ekstra): Gi et argument at din metode faktisk kjører i tid *O(N log N)*. For å forenkle argumentet kan du anta at N er en potens av 2.

Fill in your answer here

3 OPPGAVE

3 (25%).

En heltalls heap er et binært tre der alle noder, unntatt maks en, enten har to barn, eller ikke barn i det hele tatt (et løv). Unntaket er at (maks) en node kan ha et løv som enebarn. Dessuten tilfredsstiller en heltalls heap følgende datainvariant: heltallsverdien i enhver node er større enn eller lik de verdiene i barna, eller den verdien i barnet, til denne noden (hvis noden har barn i det hele tatt). Du kan anta at heap-en aldri inneholder flere enn *N* elementer. Beskriv en egnet datastruktur for å representere en heltalls heap (10%). Skriv (i Java eller i pseudokode, uten å bruke metoder fra biblioteket) metoder for å legge til elementer (5%) og for å fjerne et maximalt element (10%).

Fill in your answer here

4 OPPGAVE

4 (25%).

Gitt en rettet graf *G* med noder *0,...,V-1,* representert vha nabolister av ut-piler. Mer presist, for enhver node *v*, *adj[v]* er en kjedede liste som inneholder alle noder *w* slik at *G* har en pil fra *v* til *w*. Skriv (i Java eller i pseudokode) en metode

boolean acyclic()

som returnerer en boolsk verdi *true* hvis og bare hvis *G* er asyklisk.

(Hint: ingen informasjon om selve syklusen (hvis den finnes) trengs å bli returnert; det holder med å bruke boolske tabeller for å markere noder som har blitt besøkt tidligere og noder som befinner seg på den aktuelle søkestien.)