DT046G

Mittuniversitetet
MID SWEDEN UNIVERSITY

Tentamen

dt046g/dt064g Datastrukturer och algoritmer

Martin Kjellqvist*

2017-08-31

Instruktioner

Läs igenom frågorna noggrant innan du börja besvara dem. Du har begränsat med tid, planera hur du ska besvara frågorna. Besvara endast det som efterfrågas. Skriv inte om saker som inte berörs av frågan.

Skriv svaren på erhållna svarspapper, inte på tentan. Varje ny fråga besvaras på ett nytt svarspapper. Skriv bara på en sida på svarspapperet.

Skriv tydligt. Om svaret är oläsligt får du 0 poäng - även om svaret är korrekt. Frågorna är *inte* ordnade efter svårighetsgrad.

Tid 5 timmar.

Hjälpmedel Inga.

Max poäng 50

Antal frågor 8

Preliminära gränser

E $\geq$ 40%, D $\geq$ 50%, C $\geq$ 60%, B $\geq$ 75%, A $\geq$ 90%.

Frågor

(6p) 1. $O()$-notationen kan vid en första anblick verka ge ett grovt och oprecist intryck. Vi har under kursens gång sett flera prov på dess användbarhet.

-Beskriv $O()$-notationens svagheter att förutsäga en implementations prestanda körandes på riktig hårdvara.

-Vi har under kursen flera gånger konstaterat att om en algoritm tillhör $O(N)$ tillhör den också $O(N^2)$. Hur kan detta vara sant?

(9p) 2. Följande problem och situationer har en tidskomplexitetskaraktär.

Din uppgift är att identifiera komplexiteten i situationen. Du behöver inte nödvändigtvis lösa problemen, de finns där för att illustrera en situation.

I en tabell anger du komplexiteten med $O()$ notation för varje situation, beskriv ditt resonemang.

Ordna tabellen så att den minst komplexa står först i tabellen, den mest komplexa sist.

*martin.kjellqvist@miun.se

A För att uppmuntra sin son att lära sig multiplikationstabellen har Pappa placerat sonens födelsedagspresent i trädet på baksidan av huset.

Presenten är lagd i en kamoflagefärgad påse så den är väldigt svår att hitta.

Trädet är stort och vackert, och i varje förgrening har pappan skrivit en fråga på en liten lapp. Alla lapparna har frågor på formen *"Om 5x8 är 34 tar du högra grenen, annars tar du den vänstra."*.

Sonen är en utmärkt klättrare, men tyvärr har sonen inte lärt sig multiplikationstabellen särskilt bra så han måste gissa på nästan hälften av lapparna. Sonen är dessutom lite tankspridd, så han kommer inte ihåg alla vägar han tagit.

Hittar sonen sin födelsedagspresent?

B Lilla syster är ganska duktig på multiplikationstabellen och om hon tror att hon gjort fel på en fråga kommer hon att räkna rätt på andra försöket.

Hittar lilla syster sin brors födelsedagspresent?

C I ett **jättestort** grannskap har alla småbarnsföräldrar satt upp en telefonkedja där varje förälder har tre nödtelefonnummer. Om någon förälder blir uppringd **av någon på listan**, ringer densamme upp de övriga två på listan. De enskilda nödsamtalen tar ungefär en minut.

```
1 Anders har en lista med
2 - Bertil
3 - Cecilia
4 - Danielle
5
6 Då Cecilia ringer, ringer Anders upp Bertil och sedan Danielle.
```

Listan är så finurligt uppställd så ingen kommer att ringa i cirklar.

Mamma står och lagar mat och är precis på väg att ställa in lasagnen i ugnen då hon märker att lille Bastian inte är hemma. Han sade att han skulle till någon i grannskapet men Mamma kommer inte ihåg vem?

Hinner lille Bastian hem innan maten är kall?

3. Felsökning: båda delarna ger väldigt konstiga resulat. Resultaten är korrekta men verkar ta osedvanligt lång tid.

(4p) (a) I en tidsmätning att prova olika sorteringsalgoritmer har du följande snutt.

```
1  #define MIN_TIME 0.01;
2  template<typename sort_function>
3  double time_sort(size_t size, sort_function sort){
4    timer t;
5    std::vector<double> v(size);
6    std::generate(v.begin(), v.end(), std::rand );
7
8    int count = 0;
9    t.start();
10   while( t.get_laptime() < MIN_TIME ){
11     count++;
12     sort( v.begin(), v.end() );
13   }
14   return t.get_laptime()/count;
15 }
```

Sorteringen verkar ta alldeles för kort tid.

Varför?

Vilken är sorteringalgoritmen?

(4p) (b) Test-teamet har kommit fram till att din range klass verkar vara horribelt dålig och verkar addera komplexiteten $O(N)$ till varje operation.

Du har följande klass.

```
1  struct out_of_range{};
2
3  struct range{
4
5      double low;
6      double high;
7
8      bool in_range( double value ){
9        return value>=low&&value<=high;
10
11     int get_element_in_range(vector<double> v, int index){
12         if ( !in_range(v[index]))
13         throw out_of_range();
14     return v[index];
15   }
16 };
```

Varför lägger din range-klass till $O(N)$?

(3p) 4. (a) I vilka fall degenererar vanlig quicksort till $O(N^2)$?

(3p) (b) Vilken sorteringsmetod imiterar quicksort då den degenererar till $O(N^2)$

(3p) 5. (a) Ange vilka egenskaper som måste vara uppfyllda för att en container ska vara en *heap*.

(3p) (b) Hur gör man för använda en arraystruktur som lagringyta för en heap? (Istället för en traditionell trädstruktur).

(4p) 6. Vilka blir delresultaten av att tillämpa en heapsort på tecknen

examquestion

Förklara delstegen. Din förklaring måste vara välstrukturerad och lättläst. varje upptänklig situation, fokusera på principerna och hur noderna arrangeras.

(8p) 7. Implementera en komplett klass i c++ kod som representerar en hashtabell.

Tabellen ska innehålla operationerna insert(e), find(e):bool, med vanlig semantik. Returvärdet för find är true om elementet finns.

Tabellens lastfaktor ska aldrig överstiga 0.3.

Beskriv vilken komplexitet de båda operationerna har i din kod.

Din enda restriktion är att du inte får använda någon av klasserna `unordered_set`, `unordered_multiset`, `unordered_map`, `unordered_multimap`.

(3p) 8. Beskriv med ord eller algoritm hur `remove` kan implementeras i din hashtabell ovan.

Lycka till,
Martin och Daniel