Individuell inlämningsuppgift i kursen Algoritmer vt2015

Inledning

Syftet med inlämningsuppgiften är att ge en fördjupning inom området algoritmer. Genom att studenten själv skall implementera och på ett kontrollerat sätt utvärdera en algoritm introduceras ett vetenskapligt arbetssätt. Uppgiften tränar dessutom skriftlig och muntlig framställning genom att resultatet kommuniceras i form av en laborationsrapport och en kort muntligt presentation vid ett seminarium.

Uppgiften

Varje student skall välja en algoritm från listan nedan. Val sker via Doodle (länk på PingPong). Notera att varje algoritm kan väljas av maximalt 2 studenter. Uppgiften består av att först läsa in sig på algoritmen och sedan implementera samt testköra den på ett kontrollerat sätt i syfte att studera implementationens komplexitet. Arbetet redovisas genom en skriftlig rapport (cirka 4-6 sidor) och en muntlig presentation om 10 minuter.

Detaljer

Vid implementering är det mycket viktigt att lämpliga datastrukturer väljs. Många algoritmers komplexitet är helt beroende av vald representation. Avseende kodkvalitet, så måste implementationen av algoritmens centrala idéer dels följa den pseudokod man utgått ifrån och dels vara tydligt och korrekt genomförd, så att resonemang om komplexitet etc. går att utföra. Implementering sker i språket C och ni skall kunna lämna in en körbar fil där man kan variera input till algoritmen. All kod i denna uppgift skall skrivas egenhändigt; det är alltså inte tillåtet att använda sig av kodsnuttar från externa källor.

Huvudsyftet med laborationen är att undersöka algoritmernas komplexitet. Detta ställer naturligtvis krav på att man har möjlighet att variera input. I vissa fall, när det också finns en *brute-force* algoritm som löser samma problem, skall även denna implementeras och jämförelser göras. För vilka algoritmer detta gäller, se nästa sida.

Rapporten kan utformas på något olika sätt, men bör följa nedanstående mall där varje punkt motsvarar ett avsnitt:

1. **Inledning**. En kort beskrivning av studiens övergripande syfte, inklusive kort presentation av det algoritmiska problemet som studeras.

2. **Algoritmen eller algoritmerna i detalj**. Kopiera inte bara från källorna utan se till att er beskrivning här blir så bra och detaljerad som möjligt. Ni kan använda både beskrivningar i pseudokod och exempel på hur algoritmen arbetar. Se dock till att beskrivningen innehåller en generell förklaring, dvs. inte baserad endast på en exempelexekvering.

3. **Designval**. Vilka konkreta val gjordes vid implementeringen; t.ex. avseende datastrukturer och de delar av algoritmer som inte specificerats detaljerat i pseudokoden? Motivera val och koppla till teori, t.ex. avseende hur val av representation påverkar tids- och/eller minneskomplexitet.

4. **Experimentplanering**. Vilken input användes? Vilka körningar genomfördes? Motivera samtliga val utförligt, exempelvis genom att förklara hur input designats för att testa *best*/*average*/*worst case* till de olika algoritmerna.

5. **Resultat**. Uppmätta tider eller andra kvantitativa resultat, exempel antal exekveringar av den kritiska operationen. Fundera noga igenom vilka inputstorlekar det är vettigt att jämföra och hur resultaten bör presenteras. I de fall ni har icke-deterministiska element i era experiment, t.ex. slumpmässig input, kör upprepade exekveringar med samma inputstorlek och rapportera medelvärden, samt eventuellt standardavvikelser.

6. **Analys**. Resultaten kopplade till och belysta utifrån teorin. Här bör man resonera om de resultat som körningarna gav i relation till algoritmens teoretiska komplexitet, samt reflektera lite över sitt experimentupplägg.

7. **Slutsatser** och **diskussion**.

Appendix: Programkod och körningsexempel.

Examination

Den skriftliga rapporten, implementeringen och presentationen bedöms tillsammans med betyget U eller G. Som vanligt finns 3 examinationstillfällen, vilka ligger i kursens schema. Om någon missar presentationstillfället kommer det att ordnas ett kompletterande tillfälle under omtentaperioden i augusti, vilket då kommer att infalla i samband med examination 3.

Muntlig presentation av rapporten och demonstration av ert program sker i sal E604 fredag 5/6 kl. 8.30 – 12. Även studenter som inte presenterar förväntas närvara.

Det är möjligt att få bonuspoäng både för en mycket välskriven rapport och för en väl genomförd presentation. Om en av delarna är mycket bra, får man 1 bonuspoäng och om båda delarna är mycket bra, så får man 1,5 bonuspoäng.

Organisation

Uppgiften genomförs individuellt. Handledning sker vid två tillfällen (bokning i Doodle, länkar på PingPong) och omfattar följande moment:
H1: 2012-05-12 resp. 2012-05-20. Val av datastruktur och experimentplanering. Här är det lämpligt att de som väljer en algoritm från kap 3-8 kommer på första handledningen och de som väljer en av de senare algoritmerna kommer på den andra handledningen.
H2: 2012-05-26. Rapport. Struktur och innehåll.

Observera alltså att handledning inte ges på kodnivå. Det är ert eget ansvar att få till en fungerande implementation, vilket innebär att man bör välja en uppgift som man känner att man reder ut kodmässigt.

Algoritmer med sidhänvisning

1. Johnson-Trotters algoritm för att generera permutationer (s. 170 – 172)
2. Binärsökning och interpolationssökning (s. 177 – 178, 187 – 189)
3. Quickselect med Lomutopartition (s. 184 – 186) *
4. Närmsta paret med divide-and-conquer (s. 218 – 221) *
5. Konvexa höljet med divide-and-conquer (s. 221 – 223) *
6. Quicksort med olika strategier för val av pivotelement (202 – 207)
7. Horspools algoritm för strängmatchning (s. 285 – 288)*
8. Boyer-Moores algoritm för strängmatchning (s. 289 – 292)*
9. Öppen och sluten hashing (s. 295 – 300)
10. B-träd med insättning och uppslag (s. 302 – 305)
11. Optimala binära sökträd (s. 323 – 328)
12. Warshalls algoritm (s. 330 – 334)
13. Floyds algoritm (s. 334 – 337)
14. Prims algoritm (s. 344 – 348)
15. Kruskals algoritm (s. 351 – 357)
16. Dijkstras algoritm (s. 359 – 363)
17. Huffmans algoritm. (s. 364 – 368)
18. Backtracking. Valfri tillämpning som inte finns i boken. (s. 450)
19. Branch-and-bound. Gärna ett exempel som inte finns i boken. (s.458)
20. TSP. Avancerade lösningsstrategier/algoritmer som inte finns i boken.*
21. SAT. Någon avancerad lösningsstrategi som inte finns i boken.*

För algoritmer markerade med * ingår implementation av *brute-force* algoritm med samma representation.