

Eksperimentel undersøgelse af hobe

Dat2A K-opgave

Martin Zachariasen

6. februar 2002

1 Formalia

Dette er den karaktergivende rapportopgave på kurset Dat2A, 2001–2002. Opgaven skal løses i grupper på indtil 3 personer. Opgaven bliver stillet mandag d. 18/2-2002 kl. 9.00 og skal afleveres i førstedelsadministrationen i 2 eksemplarer inden **tirsdag d. 5/3-2002 kl. 11.30**.

2 Formål

Formålet med opgaven er at

- få praktisk erfaring med brug af datastrukturer, eksemplificeret ved *hobe* og deres anvendelser
- få et indblik i datastrukturers og algoritmers implementation i objekt-orienterede programmeringssprog
- at blive bedre i stand til at relatere teoretiske køretidsanalyser til reelle køretider på virkelige maskiner

En selvvalgt hob skal beskrives og implementeres. Denne hob skal sammenlignes med en allerede implementeret udgave af Fibonacci hoben i LEDA. Sammenligningen udføres ved at køre en sekvens af algoritmer, der benytter hoben (et såkaldt anvendelsesprogram), og måle køretiden.

3 Om opgaven

3.1 LEDA

Alle algoritmer og datastrukturer skal implementeres i C++ og så vidt muligt ved anvendelse af klassebiblioteket LEDA. Anvendelse af LEDA på DIKU's maskiner er beskrevet på:

<http://www.diku.dk/research-groups/algorithms/libraries.html#LEDA>

3.2 Selvvalgt hob

Den selvvalgte hob skal have samme interface som LEDA klassen `p_queue`. Hoben skal være parameteriseret, dvs. nøgle-typen og data-typen skal angives som (template) parametre til hoben. Det anbefales, at man benytter den udleverede¹ skabelon `~dat2a/K/my_heap.h` som udgangspunkt for implementationen af hoben. (Bemærk at det ikke er nødvendigt at understøtte UNION operationer.)

Man *kan* implementere en *binær hob* (CLRS [1] kap. 6), en *binomial hob* (CLRS kap. 19) eller varianter heraf, men dette er ikke et krav. Den selvvalgte hob må ikke være en Fibonacci hob.

3.3 Anvendelsesprogram

Anvendelsesprogrammet er tilgængeligt fra `~dat2a/K/run_heaps.cc`. Dette program består af to forskellige algoritmer, der anvender hobe: Konstruktion af Huffman koder (CLRS kap. 16.3) samt konstruktion af mindste udspændende træer ved anvendelse af Prim's algoritme (CLRS kap. 23.2). Huffman konstruktion foretages på et alfabet, hvor bogstavernes frekvenser er tilfældigt uniformt fordelte. Prim's algoritme bliver benyttet på komplette grafer og gitter-grafer, der hhv. har mange kanter og få kanter i forhold til antal knuder.

¹Alle udleverede filer er også tilgængelige via kurssets hjemmeside.

4 Krav til besvarelsen

Rapporten skal være på max. 20 sider, excl. programudskrifter og udskrifter fra kørsler (der også vedlægges). Desuden skal alle kildetekstfiler — herunder en README-fil der beskriver hvordan filerne oversættes til et færdigt program — gøres tilgængelige på en 1. delskonto (di-konto) ejet af en af deltagerne; filerne lægges i et katalog med navnet K2A og må ikke rettes efter rapportens aflevering.

Den eksperimentelle opførsel af hoben sammenlignes med Fibonacci hobens resultater ved kørsel af det udleverede anvendelsesprogram. Resultaterne skal kommenteres detaljeret.

Herudover skal rapporten på en naturlig måde behandle følgende punkter (dvs. flet besvarelsen af disse ind i rapportens struktur):

- De teoretiske køretidsegenskaber for den selvvalgte hob skal opsummeres. Hvis der ikke anvendes en hob fra CLRS, skal de teoretiske køretidsegenskaber uddybes, og alle operationer kort præsenteres som pseudo-kode.
- Den selvvalgte hobs eksperimentelle egenskaber skal diskuteres og sammenlignes med Fibonacci hobens egenskaber. Specielt skal den observerede køretid sammenlignes med værste-tids køretiderne for hoben. Prøv iøvrigt at fjerne argumentet der angiver optimering til oversætteren (argumentet `-O`), og kommenter resultatet kort.
- Det skal på overbevisende måde sandsynliggøres, at implementationen af jeres hob fungerer korrekt. Hobens korrekthed skal verificeres ved at køre jeres egen hob parallelt med LEDA-hoben: Lav en gennemtænkt sekvens af operationer, og sammenlign jeres egen hobs resultater med Fibonacci hobens resultater. Vær især opmærksom på at teste funktionaliteten af operationer, som enten slet ikke benyttes i anvendelsesprogrammet eller kun i specielle situationer (herunder `del_min` og `del_item`).
- Antag at den selvvalgte hob skulle understøtte UNION operationen. Angiv nedre og øvre grænser for køretiden af denne operation.
- Redegør for, hvad det udleverede anvendelsesprogram laver, og kommenter dets anvendelse af hobe. Prim's algoritme er implementeret lidt anderledes end i CLRS. Påpeg forskelle.

- Det bemærkes at Prim's algoritme kun foretager `decrease_p` operationer for en brøkdel af de komplette grafers kanter. Hvorfor? Følgende programstump konstruerer en kant-vægtet graf med N knuder, hvor Prim's algoritme bliver tvunget til at anvende `decrease_p` i forbindelse med undersøgelse af *alle* kanter — undtagen dem som medfører en indsættelse i hoben:

```

GRAPH<int,int> G;
for (int i = 1; i <= N; i++) G.new_node();
int c = (int) N*(N-1)/2;
for (u = G.first_node(); u; u = G.succ_node(u)) {
    for (v = G.last_node(); v != u; v = G.pred_node(v))
        G.new_edge(u,v,c--);
}
edge_array<int> cost(G);
forall_edges(e, G) cost[e] = G[e];

```

Forklar hvordan Prim's algoritme vil opføre sig på denne graf (lav f.eks. en tegning af grafen for $N=5$). Hvad sker der i hoben hver gang en knude får formindsket sin nøgle-værdi? Foretag eksperimenter med denne graf som input til Prim's algoritme og kommenter resultatet.

5 Gode råd

Spørgetime om opgaven annonceres på kursets hjemmeside. Desuden svarer instruktorerne på spørgsmål i forbindelse med de sædvanlige øvelsestimer. Nyhedsgruppen `diku.dat2a` er beregnet som elektronisk diskussionsforum for problemstillinger knyttet til kurset. Instruktorer og lærere læser indlæg i denne gruppe regelmæssigt, og bidrager gerne med opklaring af spørgsmål m.v. Brug nyhedsgruppen.

Litteratur

- [1] T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein: Introduction to Algorithms (2nd ed.), MIT Press, 2001.