# PERANGKAT LUNAK PENDUKUNG PEMBELAJARAN ALGORITMA *HEAPSORT*

Cici Al Akhyati[1], Asahar Johar[2], Boko Susilo[3]

[1,2,3]Program Studi Teknik Infomatika, Fakultas Teknik, Universitas Bengkulu.
Jl. WR. Supratman Kandang Limun Bengkulu 38371A INDONESIA
(telp: 0736-341022; fax: 0736-341022)

[2]asahar.johar@yahoo.com

*Abstrak:* Tujuan dari penelitian ini adalah menghasilkan perangkat lunak yang mampu mendukung proses pembelajaran algoritma *heapsort*, khususnya perangkat lunak yang dapat menjelaskan dan menampilkan simulasi dari prosedur kerja algoritma *heapsort*. Algoritma heapsort adalah salah satu algoritma pengurutan yang dipelajari di Teknik Informatika untuk memecahkan masalah pengurutan. Dalam proses pembelajaran Algoritma *heapsort* dosen dan mahasiswa Teknik Informatika dituntut untuk menggambar prosedur kerja algoritma *heapsort* yang sulit. Permasalahan yang muncul adalah bagaimana membuat perangkat lunak yang mampu mendukung pembelajaran algoritma heapsort. Perangkat lunak ini menggunakan Visual Basic 6.0 sebagai bahasa pemrograman. Perangkat lunak dikembangkan menggunakan metode *waterfall* yang melibatkan proses analisis dan desain berorientasi objek, proses pengkodean dan proses pengujian. Proses pengujian dilakukan dengan mencocokkan hasil pengurutan yang dilakukan secara manual dengan hasil yang diperoleh perangkat lunak ini. Hasil eksekusi dari proses pengurutan dapat dicetak berupa *print out* dan dapat disimpan dalam bentuk *text file* (*.txt)

*Kata kunci:* Perangkat lunak, *Heapsort*, *Visual Basic* 6.0, *Waterfall*, Metode Berorientasi Objek.

***Abstract:*** **The purpose of this research is to produce software which is able to support the learning of heapsort algorithm, especially the software which can explain and show the procedure of work of the algorithm heapsort. Heapsort algorithm is one of sorting algorithm which is learn in "Teknik Informatika" to solve the problem sorting. In the process of heapsort algorithm learning, lecture and student of "Teknik Informatika" is demanded to describe the difficult procedure of work of heapsort algorithm. The problem is how to make software which is able to support the learning of heapsort algorithm. This software uses visual basic 6.0 as a program language. The software is constructed with using waterfall method which involves the process of object oriented analysis design, code process, testing process. The testing process is done with adjusting the result of manual sorting with that is produced by the software. The result of execution of the sorting process can be printed and saved in text file.**

**Keywords: Software, Heapsort, Visual Basic 6.0, Waterfall, Object Oriented Method**

## I. PENDAHULUAN

Komputer dibuat sebagai alat bantu untuk menyelesaikan masalah. Permasalahan apapun

dapat diselesaikan oleh komputer asalkan langkah-langkah penyelesaiannya disediakan oleh manusia. Dalam hal ini, komputer hanya bertindak untuk menjalankan perintah-perintah yang tertulis didalam program. Urutan langkah-langkah penyelesaian masalah yang terdapat dalam program disebut algoritma. Algoritma merupakan kunci utama kehidupan semua program meski untuk aplikasi sederhana sekalipun. Algorima disusun oleh langkah-langkah berhingga, yang masing-masing langkahnya memerlukan satu operasi atau lebih.

Referensi [1] mengatakan sebenarnya manusia mampu melaksanakan perintah-perintah yang tertulis didalam program, tetapi komputer memiliki kelebihan dibandingkan manusia. Komputer adalah benda mati, jadi tidak mengnal lelah dan bosan. Komputer mampu mengerjakan perintah yang banyak sekalipun, selain itu komputer juga mampu mengerjakan perintah yang sama berulang kali, 100 kali, sejuta kali, atau berapa kalipun yang manusia perintahkan. Manusia sering lupa, sedangkan komputer tidak. Komputer memiliki memori yang besar sehingga mampu menyimpan data dan informasi dalam jumlah yang banyak.

Dalam kehidupan sehari-hari komputer sering digunakan untuk mengurutkan sekumpulan nilai. Permasalah pengurutan (sorting) banyak ditemukan dalam aplikasi yang berhubungan dengan data pengurutan Nomor Induk Mahasiswa (NIM), ilai, Nomor ID sebuah barang inventaris, daftar kata, daftar nama, dan sebagainya. Lebih mudah membaca data terurut daripada data yang tersusun acak karena data yang terurut memudahkan proses pencarian data.

Didalam bidang Teknik Informatika banyak terdapat jenis-jenis algoritma pengurutan yang dapat digunakan untuk memecahkan masalah pengurutan, di antaranya adalah *bubble sort*, *merge sort*, *insertion sort*, *quick sort*, dan *selection sort*, serta masih banyak lagi jenis algoritma pengurutan lainnya. Oleh karena itu, teknik dan kejelian untuk memilih algoritma pengurutan yang tepat dan sesuai dengan permasalahan pengurutan yang dihadapi sangat diperlukan karena masing-masing algoritma pengurutan tersebut memiliki karakteristik yang berbeda-beda.

Referensi [2] menyebutkan algoritma *heapsort* adalah algoritma pengurutan yang memiliki kompleksitas waktu terbaik. Selain itu juga, *heapsort* menerapkan teknik yang unik di dalam memecahkan masalah pengurutan, yaitu dengan menggunakan *heaptree*"

Algoritma *heapsort* memiliki kelebihan ketika menangani data dalam skala yang besar atau *massive*. Pengurutan bilangan dapat lebih efisien bila menggunakan algoritma *heapsort*, karena algoritma ini memiliki kelebihan yaitu tidak menggunakan banyak tabel, tetapi hanya satu tabel yang dipakai untuk menyimpan hasil dari pengurutan tersebut.

Manfaat besar yang diberikan oleh pengurutan bilangan menggunakan algoritma *heapsort* tentu harus diikuti dengan kemampuan mahasiswa Teknik Informatika dalam memahami dan mengerti prosedur kerja dari algoritma *heapsort*. Pada kenyataanya sistem pembelajaran yang sedang berjalan saat ini belum sepenuhnya mendukung. Hal ini disebabkan sarana yang digunakan dalam mendistribusikan pengetahuan dari dosen kepada mahasiswa masih sangat minim.

Dalam proses pembelajaran yang dilakukan selama ini, untuk memahami materi algoritma *heapsort*, mahasiswa dituntut untuk menggambarkan proses atau prosedur kerja dari algoritma *heapsort*. Proses penggambaran ini akan

terus berlangsung pada setiap semester mata kuliah tersebut diatas yang terdapat materi mengenai algoritma *heapsort*nya. Kondisi seperti ini tentu saja akan menguras pikiran dan tenaga dosen jika harus memberikan penjelasan berulang karena perbedaan daya tangkap mahasiswa yang tidak sama. Pembelajaran akan semakin sulit dengan adanya langkah-langkah atau prosedur kerja yang terdapat pada algoritma *heapsort*, karena mahasiswa dituntut untuk mengerjakan suatu perintah ang sama berulang kali.

Peranan komputer tidak terlalu diperlukan apabila data yang diurutkan hanya sedikit (misalkan hanya 10 buah data), karena mahasiswa mampu mengerjakan secara manual. Bagaimana jika data yang diurutkan jumlahnya sangat banyak (misalnya 50 sampai ribuan data)? Tentu mahasiswa juga mampu melakukannya, tetapi mungkin dibutuhkan waktu yang cukup lama, karena memerlukan ketelitian dan kesabaran untuk mengerjakannya. Disinilah computer berperan dalam mengerjakan perintah yang diberikan tanpa lelah dan bosan namun tetap memberikan hasil pekerjaan yang teliti dalam waktu yang singkat.

Dengan memanfaatkan peranan komputer dalam membantu melakukan proses pengurutan, penulis bermaksud untuk merancang dan mengembangkan sebuah perangkat lunak yang mampu untuk menjelaskan prosedur kerja dari algoritma *heapsort* dalam menampilkan data secara terurut. Oleh karena itu, penulis mengambil tugas akhir dengan judul "**Perangkat Lunak Pendukung Pembelajaran Algoritma *Heapsort***".

## II. LANDASAN TEORI

### A. *Algoritma Heapsort*

*Sorting* atau pengurutan merupakan pekerjaan yang sering dilakukan dalam kehidupan sehari-hari. Tujuan dari pengurutan adalah menyusun sejumlah data sedemikian sehingga diperoleh suatu keteraturan apakah terurut menaik atau terurut menurun. Dengan adanya data terurut, disamping enak dipandang maka pencarian dapat dilakukan dengan cepat baik dengan algoritma biner maupun dengan rumus interpolasi.

Algoritma *Heapsor*t menggunakan struktur pohon *heap* (*heaptree*) untuk melakukan pengurutan. Prosedur *heapsort* mengurutkan sekumpulan data pada sebuah *array* atau *heaptree*. Cara kerjanya adalah, *heapsort* akan mengambil data pada *node* akar (*index array* = 1) dan menggantinya (*exchange*) dengan data pada *node* paling akhir (*index array* = *index* paling maksimum dari pohon *heap*). Setelah itu, *node* terakhir dihapus dan *heapsort* memanggil prosedur *heapify*

dengan tujuan agar setelah proses penggantian data, pohon masih memenuhi property *heap*. Data-data yang dikeluarkan merupakan data yang terurut, baik menaik (*ascending*) maupun menurun (*descending*).

### B. *Heaptree (Pohon Heap)*

*Heaptree* [3] adalah pohon biner yang memiliki beberapa persyaratan berikut:

1) *Struktur pohon:* harus lengkap atau sempurna, kecuali untuk *level* yang terbawah (dengan syarat: semua *node* pada level terbawah harus berada di sebelah kiri).

2) *Struktur pohon:* memenuhi properti *heap*, yaitu:

a. *Node* akar (*root node*) memiliki data terbesar atau terkecil yang terdapat pada pohon.
b. *Subtree* di sebelah kiri dan sebelah kanan *node* akar memenuhi persyaratan berikut: *node* induk (*parent*) memiliki data yang paling besar atau

paling kecil dibandingkan dengan data pada kedua anaknya (*child node* sebelah kiri atau sebelah kanan).

### C. *Representasi Pohon Biner dengan Array*

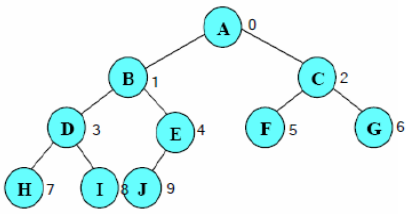
*Array* atau larik adalah struktur data static yang menyimpan sekumpulan elemen yang bertipe sama dimana setiap elemen diakses langsung melalui indeksnya. Indeks *array* harus tipe data yang menyatakan keterurutan misalnya integer atau karakter [4].

Pohon biner dapat direpresentasikan dengan *array* dengan ketentuan sebagai berikut dengan i adalah indeks:

*a.* Akar (r*oot)* ditempatkan pada posisi 1 dalam *array*

b. *Left subtree* data yang ada pada posisi ke-i diletakkan pada posisi 2i

c. *Right subtree* - nya diletakkan pada posisi 2i + 1




Gambar 1. Pohon Biner Lengkap [5]

Sesuai dengan Gambar 1, Jumlah data maksimum jika jumlah *level* = 4 adalah 15. Dengan demikian pohon di atas akan disimpan dalam *array* seperti Gambar 2 di bawah ini :

| A | B | C | D | E | F | G | H | I | J | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |

Gambar 2 Representasi Pohon Biner dengan *Array* [5]

## III. METODOLOGI

### A. *Tempat Penelitian*

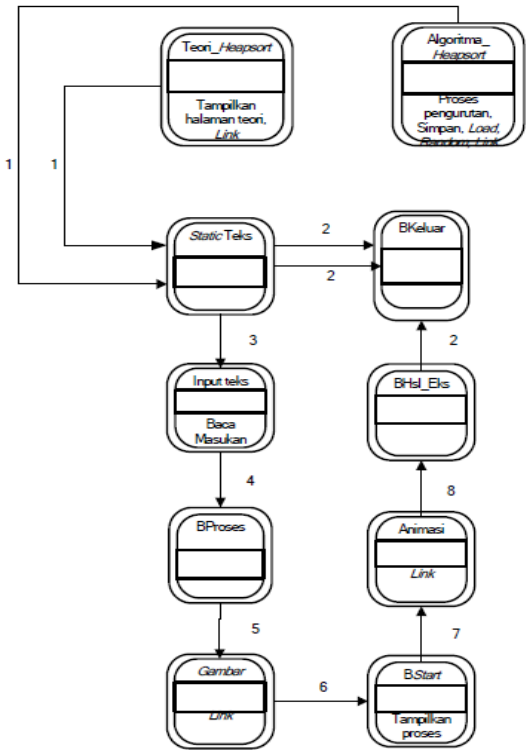
Penelitian dilakukan pada laboratorium komputer (Research Laboratory). Dengan cara melakukan pengolahan terhadap data yang telah dikumpulkan. Pada metode ini penulis melakukan penelitian dibantu oleh fasilitas laboratorium komputer, dengan menggunakan personal komputer (PC).

### B. *Teknik Pengumpulan Data*

Penelitian dilakukan dengan menggunakan metode penelitian kepustakaan (*Library Research*) dimana pengumpulan data dilakukan dengan membaca bukubuku literatur, diktat kuliah, buku-buku yang berkaitan dengan yang berhubungan dengan struktur data, desain dan analisis algoritma, dan prosedur kerja algoritma *heapsort*, dan literatur lain yang berasal dari kliping majalah, dan artikel-artikel dari internet.

## IV. ANALISIS DAN PERANCANGAN SISTEM

Lapisan pesan merupakan bagian dari penggambaran OOA. Hubungan pesan adalah model hubungan yang menggambarkan suatu objek mengirimkan pesan kepada objek lain untuk melakukan pemrosesan. Keterangan lebih lanjut dapat dilihat pada Gambar 3.




*Gambar 3 Lapisan Pesan OOA*

Berdasarkan gambar 3, penomoran dari angka 1 sampai 10 dapat dijelaskan sebagai berikut:

1. Setelah *user* memilih bagian Pendukung Pembelajaran yang diinginkan yang terdiri dari subjek Algoritma *Heapsort* dan Teori *Heapsort*, maka akan tampil sebuah *Static*_Teks yang menginformasikan mengenai algoritma *heapsort.*
2. Dari *form* terpilih baik *form* algoritma *heapsort* maupun teori *heapsort* akan muncul *button* keluar yang berarti mengakhiri program secara keseluruhan.
3. Pada *form* algoritma *heapsort*, *user* akan diminta memasukkan sembarang data berupa angka dengan syarat-syarat yang telah ditentukan.
4. *Button* proses ketika diklik akan menampilkan *form* baru yaitu *form* proses pengurutan.
5. Ketika *button* proses diklik, form proses pengurutan akan menggambarkan representasi data *input* kedalam pohon biner *heap*.
6. *User* diminta untuk mengklik *button start* untuk menerima *service* berupa animasi proses pengurutan bilangan.
7. Menampilkan proses pengurutan bilangan baik secara menaik (*ascending*) maupun secara menurun (*descending*).
8. *Button* hasil eksekusi yang diklik *user* akan menampilakan *form* baru yang berisi hasil eksekusi dari program pengurutan.

## V. HASIL DAN PEMBAHASAN

Berdasarkan proses analisis dan desain berorientasi objek yang telah dilakukan sebelumnya, didapat 7 (tujuh) *form* yang membentuk perangkat lunak pendukung pembelajaran algoriitma *heapsort*. Berikut ini adalah *form- form* yang terdapat pada perangkat lunak pendukung pembelajaran algoritma heapsort:

### *A. Form Algoritma Heapsort*

Pada *form* algoritma *heapsort* terdapat fasilitas untuk menginput data dengan syarat-syarat yang telah ditentukan sebelumnya. Fasilitas *input* data tersedia pada *button load*, yaitu fasilitas memasukkan angka dari *file input*. *Button random* berfungsi untuk menghasilkan barisan data secara acak. Pada *form* algoritma *heapsort* ini, *user* diminta untuk memilih hasil pengurutan apakah berupa data pengurutan secara menaik (*ascending*) atau berupa data pengurutan secara menurun (*descending*). User juga harus memilih sifat *heap* yaitu apakah data pada *node parent* lebih besar atau samadengan data pada *node* anak sebelah kiri dan anak sebelah kanan, atau sifat heap yang *node parent*nya lebih kecil atau sama dengan data pada anak sebelah kiri dan anak sebelah kanan. *Form* algoritma *heapsort* digambarkan pada gambar 4 sebagai berikut:


*Gambar 4 form Algoritma Heapsort*

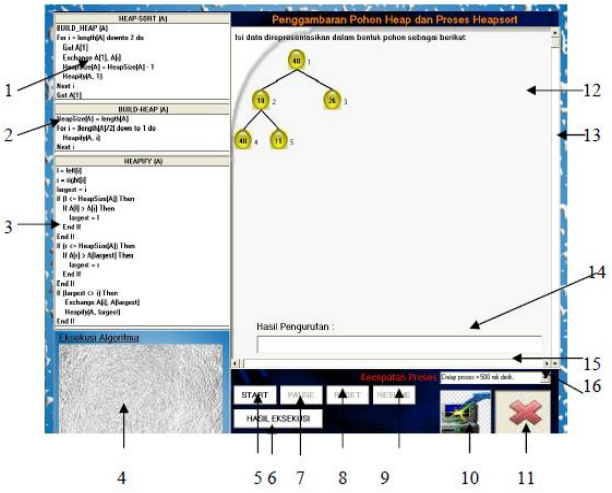
Keterangan Gambar 4 *Form* Algoritma *Heapsort*:

1. *Button close*, ketika diklik berfungsi untuk menutup *form* Algoritma *Heapsort.*

2. *Textbox*, berfungsi untuk memasukkan barisan angka yang dipisahkan dengan huruf koma satu dengan yang lain.
3. *Button save*, ketika diklik berfungsi untuk menyimpan *input* kedalam bentuk *file*.
4. *Button Load*, ketika diklik berfungsi membuka *file input*.
5. *Button random*, ketika diklik berfungsi untuk menghasilkan barisan angka secara acak.
6. *Optionbutton*, ketika diklik berfungsi untuk memilih hasil pengurutan terurut menaik (*ascending*)
7. *Optionbutton*, ketika diklik berfungsi untuk memilih hasil pengurutan terurut menurun (*descending*).
8. *Optionbutton*, ketika diklik berfungsi untuk memilih properti *heap* sehingga data pada *node parent* adalah data yang terbesar.
9. *Optionbutton*, ketika diklik berfungsi untik memilih properti *heap* sehingga data pada *node parent* adalah data yang terkecil.
10. *Button* proses pengurutan, ketika diklik berfungsi untuk menampilkan *form* berikutnya yaitu *form* proses pengurutan
11. *Button home*, ketika diklik berfungsi untuk menampilkan *form home*.
12. *Button keluar*, ketika diklik berfungsi untuk menutup *form* Algoritma *Heapsort*.

*B. Form Proses Pengurutan*

Pada Gambar 5.3 diatas, terdapat *button* proses pengurutan. Ketika *button* tersebut diklik, akan muncul form proses pengurutan. Pada *form* ini, data yang i*input* telah digambarkan dalam bentuk pohon *heap*. Pada bagian kiri *form* terdapat tiga buah tabel yang masing-masing mendeskripsikan algoritma *heapsort*, *builheap*, dan *heapify*. Ketika user mengklik button start, maka proses pengurutan akan dimulai dengan menggunakan prosedur-prosedur *heapsort* yang telah tersedia. Selain itu, terdapat fasilitas *combobox* yang digunakan untuk menentukan kecepatan proses pengurutan. Fasilitas pendukung lain yang digunakan *form* proses pengurutan ini adalah button start, button resume dan *button reset*. *Form* proses pengurutan dijelaskan pada Gambar 5.




*Gambar 5 Form Proses Pengurutan*

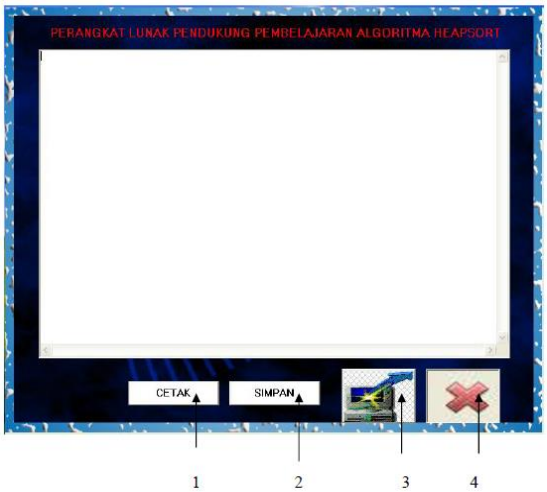
Keterangan Gambar 5 *Form* Proses Pengurutan:

1. Label algoritma prosedur *heapsort*
2. Label algoritma prosedur *buildheap*
3. Label algoritma prosedur *heapify*
4. Label untuk menampilkan eksekusi algoritma.
5. *Button start*, ketika diklik berfungsi untuk memulai proses pengurutan
6. *Button* hasil eksekusi, ketika diklik berfungsi untuk menampilkan *form* hasil eksekusi.
7. *Button pause*, ketika diklik berfungsi untuk menghentikan proses pengurutan untuk sementara.
8. *Button reset*, ketika diklik berfungsi untuk mengulangi proses pengurutan dari awal.
9. *Button resume*, ketika diklik berfungsi untuk melanjutkan proses pengurutan setelah dihentikan untuk sementara (*pause*).

10. *Button home*, ketika diklik berfungsi untuk menampilkan *form home*.
11. *Button* keluar, ketika diklik berfungsi untuk keluar *form* Proses Pengurutan.
12. Daerah penggambaran pohon.
13. *Vscrollbar*, untuk menggulung daerah penggambaran pohon secara vertikal.
14. *Textbox* untuk menampilkan hasil pengurutan
15. *Hscrollbar*, untuk menggulung daerah penggambaran pohon secara horizontal
16. *Combobox*, untuk memilih kecepatan proses

*C. Form Hasil Eksekusi*

Berdasarkan data yang diinput dan proses pengurutan menggunakan algoritma *heapsort*, maka didapat hasil eksekusi algoritma. Pada *form* algoritma *heapsort*, terdapat *button* hasil eksekusi. Apabila *button* tersebut diklik, maka akan muncul *form* hasil eksekusi. *Form* hasil eksekusi ditampilkan pada Gambar 6 berikut:




*Gambar 6 Form Hasil Eksekusi*

Pada *form* ini, terdapat fasilitas simpan *file* dan cetak *file* yang masingmasing tersedian pada *button* cetak dan *button* simpan. Keterangan Gambar 6 *Form* Hasil Eksekusi:

1. *Button* cetak, ketika diklik berfungsi untuk mencetak hasil eksekusi ke *printer*.
2. *Button* simpan, ketika diklik berfungsi untuk menyimpan hasil eksekusi ke *text file* (*.txt).
3. *Button home*, ketika diklik berfungsi untuk menampilkan *form home*.
4. *Button* keluar, ketika diklik berfungsi untuk keluar *form* Hasil Eksekusi.

## VI. KESIMPULAN

1) *Hasil analisis:* untuk merancang perangkat lunak yang mampu menampilkan simulasi dari prosedur kerja algoritma *heapsort* adalah dengan pemahaman pengenai prosedur kerja algoritma *heapsort.* Perangkat lunak ini dibangun menggunakan bahasa pemrograman Visual Basic 6.0.

2) *Pembuatan perangkat lunak:* dilakukan berdasarkan hasil analisis dan desain berorientasi objek:

a. Pengembangan sistemnya menggunakan metode pengembangan *Linear Sequential Model (Waterfall),* yang didalamnya terdapat metode analisis dan desain berorientasi objek.
*b*. Ada dua puluh tujuh kelas dan objek yang diperoleh dari hasil analisis

3) *Struktur hirarki:* menggunakan struktur *gen spec* dan *whole part*

4) *Perangkat lunak yang dihasilkan:* berupa program pendukung pembelajaran algoritma *hepasort*.

5) *Data yang diinput:* divisualisasikan kedalam bentuk pohon heap.

6) *Data akan diurutkan:* sesuai dengan nilai yang diinput dengan tetap memperhatikan prosedur *heapsort*

7) *Hasil eksekusi program:* dapat terdokumentasi dan dapat dicetak.

## REFERENSI

[1] Munir, Rinaldi. 2004. Algoritma dan Pemrograman dalam Bahasa Pascal dan C. Informatika. Bandung.

[2] Chalikdjen, Efendy dkk Heap Tree dan Kegunaannya dalam Heap Sort. Makalah STMIK. Institut Teknologi Bandung. [Online] tersedia: http://MakalahStemik08 .pdf. [15 April 2008]
[3] Hariyanto, Bambang. 2003. Struktur Data. Informatika Bandung.
[4] Jauhari, Jaidan. 2004. Struktur Data. [Online] tersedia: http://Primitif List.pdf. [31 Mei 2008]
[5] Denny. 2000. Struktur Data dan Algoritme. Fakultas Ilmu Komputer Universitas Indonesia. [Online] tersedia: http://pala.acad.cs.ui.ac.id/academic/animation//