ume IV 2009

ISSN: 1907-3526

SRITI

Seminar Nasional Riset Teknologi Informasi

Proceeding

Seminar Nasional Riset Teknologi Informasi 2009

"Ubiquitos Computing"

Yogyakarta, 08 Agustus 2009

Komputasi

Kecerdasan Buatan

Teknologi Basis Data

(Data Meaning, Data Warehouse)

Pemodelan dan Aplikasi Sistem Informasi

Komunikasi Data dan Jaringan Komputer

Signal Processing

Sistem Kendali Robotika

Pengolahan Citra

Multimedia dan Grafika

Games

Diselenggarakan Oleh :

YAYASAN PENDIDIKAN WIDYA BAKTI

STMIK AKAKOM YOGYAKARTA

Yang Pertama dan Utama

SRITI
Seminar Nasional Riset Teknologi Informasi

Proceeding

Seminar Nasional
Riset Teknologi Informasi 2009

"Ubiquitos Computing"

Yogyakarta, 08 Agustus 2009

Diselenggarakan Oleh :

YAYASAN PENDIDIKAN WIDYA BAKTI
STMIK
AKAKOM
YOGYAKARTA
Yang Pertama dan Utama

STEERING COMMITTEE

Prof. H. Adhi Susanto, M.Sc., Ph.D (UGM)
Prof. Drs. Suryo Guritno, M.Stat., Ph.D (UGM)
Prof. Dr. Ir. Achmad Djunaedi, MUP (UGM)
Prof. Dr. Ir. Prayoto., M.Sc (STMIK AKAKOM)
Prof. Drs. Setiadji, S.U. (STMIK AKAKOM)
Dr. Ir. Inggriani Liem (ITB)
Dr. Ir. Titon Dutomo, M.Eng (PENS-ITS)
Dr. Ir. Sasongko Pramono Hadi, DEA (Dir. ST Multimedia MMTC)
Ir. Lukito Edi Nugroho, M.Sc., Ph.D (UGM)
Drs. Retantyo Wardoyo, M.Sc., Ph.D (UGM)

PELAKSANA

Pelindung
Ketua STMIK AKAKOM

Penanggung Jawab
Kepala Puslitbang dan PPM

Tim Pengarah
Drs. Berta Bednar, M.T.
Ir. M. Guntara, M.T.
Heru Agus Triyanto, S.E., M.M.
Drs. Tri Prabawa, M.Kom.

Ketua 1
Ir. Totok Suprawoto, M.M., M.T.

Ketua 2
Dra. Syamsu Windarti, M.T. , Apt.

Bendahara
Pulut Suryati, S.Kom.
Dra. Torsinawati

Kesekretariatan/Komunikasi

Deborah Kurniawati, S.Kom.
Cosmas Haryawan, S.Tp., S.Kom.
Sri Redjeki, S.Si., M.Kom.
Sigit Anggoro, S.T., M.T.
H. Sri Widodo
F. Prihantini
Nailus Sa'adah
Rita Darundia
Theo A. Richie Y.

Materi/Acara
L.N. Harnaningrum, S.Si., M.T.
Enny Itje Sela, S.Si., M.Kom.
Agung Budi Prasetyo, S.Kom., M.Kom.
Aries Damayanti, S.Kom.
Al. Agus Subagyo, S.E., M.Si.
Thomas Edison Tarigan, S.Kom.

Perlengkapan\Dokumentasi\Dekorasi\Akomodasi\Konsumsi
Indra Yatini Buryadi, S.Kom., M.Kom.
Ir. Mashudi
F.X. Henry Nugroho, S.T.
Ary Adjidharma A.W., S.Kom., MMSI
Dra. M. Titik Maryanti
Dwi Suwarsono
Teki Astuti
Kuindra Iriyanto

DAFTAR ISI

Organisasi Berkas dengan Menggunakan 3 Varian Metode Hash (*Coalesced Hashing, Prograssive Overflow, Buckets*)

Pulut Suryati

STMIK AKAKOM Yogyakarta

Abstrak

Sistem pengelolaan basis data memerlukan dua layanan yaitu definisi basis data dan manipulasi basis data. Ujuk Kerja dipengaruhi oleh metode penyimpanan dan metode akses. Organisasi berkas dapat disamakan dengan istilah organisasi basis data, yaitu media penyimpanan dan metode akses yang digunakan pada media penyimpanan yang digunakan terhadap berkas/basis data. Salah satu metode dalam organisasi berkas dengan metode Hashing. Dengan mengetahui metode tersebut dapat diketahui metode terbaik atau efiesien untuk mengatasi benturan / coalision dalam proses pengorganisasian berkas. Untuk menangani masalah kasus coalising atau tumbukan terdapat beberapa metode diantara adalah *Coalesced Mashing, Prograssive Overflow, Buckets*. Dari metode tersebut disusun bagan alir program berikut metode penyelesaian yaitu *Coalesced Mashing* untuk tipe LISCH dan tipe EISCH , *Prograssive Overflow* dan *Buckets*. Kemudian diimplementasikan dalam suatu aplikasi. Aplikasi yang dihasilkan dapat digunakan sebagai media pembelajaran mahasiswa untuk mengilustrasikan proses pengorganisasian berkas dengan metode hash yaitu *Coalesced Mashing, Prograssive Overflow, Buckets*.

Kata Kunci : hashing, Coalesced Mashing, Prograssive Overflow, Buckets, collision.

PENDAHULUAN

Sistem pengelolaan basis data memerlukan dua layanan yaitu definisi basis data dan manipulasi basis data. Unjuk kerja *Database Manajemen System* (DBMS) dipengaruhi oleh dua faktor utama yaitu: metode penyimpanan dan metode akses. Untuk itu pemilihan DBMS yang sesuai dengan kebutuhan menjadi penting. Dalam konteks ini, istilah berkas diartikan sebagai sekumpulan data sejenis secara relasi (sama dengan istilah basis data). Organisasi berkas dapat disamakan dengan istilah organisasi basis data, yaitu media penyimpanan dan metode akses yang digunakan pada media penyimpanan yang digunakan terhadap suatu berkas/basis data. Sedangkan sistem basis data merupakan berkas dan program pengolah.

Program-program aplikasi sederhana yang memerlukan memori relatif kecil, sangat mungkin untuk menyimpan data-pdata yang diolah/diperlukan dalam memori utama komputer. Umumnya program-program aplikasi tersebut akan memanfaatkan variabel-variabel untuk menampung data input atau hasil olahan.

TINJAUAN PUSTAKA

Pada Penelitian pengukuran kinerja algoritma hashing dalam pencarian kata oleh Chofid Ahmad, Rina Andriani, Rini Dyah Astuti dan Selly Meliana yang dilakukan pada dasarnya adalah untuk mendapatkan kinerja algoritme yang paling optimal dari ketiga struktur data tersebut diatas (antara *Binary Tree, Hash*, dan *Digital Tree*). Namun penelitian hanya difokuskan pada algoritme *Hash*. Penelitian ini akan membandingkan kinerja sebuah algoritme *Hash* yang berbeda cara penyisipan datanya akibat *collision*. Ada dua cara penyisipan yang kami lakukan untuk daerah *overflow* (akibat *collison*). Cara pertama adalah menyisipkan secara *random* data yang mengalami *collision*, sedangkan cara kedua adalah menyisipkan secara terurut berdasarkan frekwensi kemunculan kata. Struktur data *Hash* yang digunakan adalah *Separate Chaining Hash*. Dalam laporan ini kami mengistilahkan model algoritme dengan *random overflow* sebagai *Hash-Random Chain* dan algoritme dengan menggunakan informasi frekwensi sebagai *Hash-Frequent Chain*.

DASAR TEORI

Organisasi Berkas

Untuk membangun suatu sistem perangkat lunak yang penting, diperlukan suatu struktur dan pengolahan informasi yang efektif dan efisien (Alan L.Thrap, 1988).

Terutama struktur penyimpanan harus mendukung efisiensi pengaksesan data untuk sekumpulan program sistem informasi yang meliputi sistem berkas dan cara pengorganisasian file.

- Sistem Berkas atau pengarsipan yaitu
  Suatu sistem untuk mengetahui bagaimana cara menyimpan data dari file tertentu dan mengorganisasi file yang digunakan
- Sistem Akses yaitu
  Cara untuk mengambil informasi dari suatu file

Secara efektif dalam pengorganisasian sekumpulan record yang membutuhkan akses sebuah record dengan cepat adalah organisasi Berkas Relatif. Dalam berkas relatif ada hubungan antara Key yang dipakai untuk mengidentifikasi record dengan lokasi record dalam penyimpan sekunder. Tetapi yang perlu diperhatikan adalah bahwa urutan record secara logik tak ada hubungan dengan urutan secara fisik. Record tidak perlu tersortir secara fisik menurut nilai key.

Bagaimana record yang ke-N dapat ditemukan? Dalam hal ini, perlu kita buat hubungan yang akan menerjemahkan antara nilai Key dan Address. Hubungan ini dinyatakan sebagai fungsi R, yang merupakan fungsi pemetaan

R(NILAI KEY) ⟶ ADDRESS

Dari nilai key ke address dalam penyimpan sekunder. Apda waktus ebuah record ditulis ke dalam berkas relatif, fungsi pemetaan R digunakan untuk menerjemahkan Nilai Key dari Record menjadi Address, dimana record tersebut disimpan.

Salah satu pendekatan yang umum dipakai untuk mengimplementasikan R(NILAI KEY)→ Addres adalah dengan kalkulasi terhadap terhadap nilai ker. Hasilnya adalah alamat relatif. Teknik ini dapat dipakai sendiri atau bersama-sama dengan pencarian tabel. Ide dasar dari kalkulasi alamat adalah mengubah jangkauan niali key yang mungkin, menjadi sejumlah kecil alamat relatif. Salah satu masalah dari teknik ini adalah ditemukannya alamat relatif yang sama untuk nilai key yang berbeda. Keadaan di mana R(K1)=R(K2) dan K1#K2 disebut Benturan (*COLLISION*). Sedangkan nilai key K1 dan K2 disebut Synonim.

Kalkukasi terhadap nilai key untuk mendapatkan sebuah alamat disebut Fungsi Hash. Tujuan utama fungsi hash adalah untuk mengurangi benturan.

Untuk memposisikan rekord yang mengalami tabrakan ke lokasi lain di perlukan metode dimana mekanismenya harus diusahakanagar dilakukan seminim mungkin untuk memperoleh dari home addres. Terdapat tiga penyelesaian terhadap tabrakan yaitu

1. Penyelesaian tabrakan menggunakan link
2. Penyelesaian tabrakan tanpa link
3. Penyelesaian tabrakan menggunakan link Semu.

Fungsi *Hashing*

Hashing merupakan metode pengaksesan data yang dilakukan dengan cara memetakan-/mengkonversikan himpunan nilai kunci aktual pada atribute *primary key* dalam record menjadi himpunan alamat indeks indeks memori (posisi subcript dalam larik), sehingga range address menjadi kecil. Fungsi untuk mengkonversikan nilai aktual kunci menjadi lokasi alamat indeks disebut fungsi hash. Metode pencarian yang memanfaatkan fungsi hash disebut sebagai *hashing* atau *hash addresing*. Tujuan utama fungsi hash adalah agar dua buah nilai kunci (=τ) mempunyai nilai hash yang berbeda. Jika tidak, maka akan terjadi collision.

Fungsi hash yang digunakan untuk memetakan nilai-nilai kunci ke alamat indeks indeks penyimpanan dinotasikan sebagai berikut :

H: K → L

Keterangan :

H : Fungsi hash
K : Himpunan nilai kunci
L : Himpunan alamat indeks indeks dalam memori (posisi subskrip dalam larik)

Setiap fungsi hash tidak bisa terlepas dari *collision*, yaitu terjadinya tabrakan dalam pemetaan data I nilai kunci pada suatu alamat I nomor indeks tabel yang sama. Fungsi hash yang memberikan kemungkinan semakin kecil terjadinya collision berarti fungsi *hash* tersebut semakin baik [Edhy, Sutanta, 2004]. Cara untuk meminimalkan terjadinya *collision* adalah

1. Mengganti fungsi hash
2. Mengganti metode hashing
3. menurunkan *packing factor*

Packing factor = perbandingan jumlah data yang disimpan terhadap kapasitas berkas
Packing_Factor=jumlah_record/jumlah_lokasi_Penyimpanan
Fungsi Hash [Alan L. Tharp, 1988] yang sering digunakan antara lain adalah :

1. Metode K mod N
2. Metode K mod P

3. Metode Trunction
4. Metode Folding
5. Metode Konverensi Radix

Penyelesaian *collision* pada organisasi berkas *hashing*, dapat diklarifikasikan dalam tiga kelompok [Edhy Sutanta, 2004], yaitu :

1. Menggunakan penghubung (*Link*), yang termasuk kelompok ini, adalah :
   a. Metode *coalesced hashing*
      Dalam metode ini, maka record-record hanya memuat field-field data saja.
2. Tanpa menggunakan penghubung (*without link*), yang termasuk kelompok ini adalah
   a. Metode *progressive overflow*
   b. Metode *linear quotient*
   c. Metode *buckets*
   Dalam metode ini, maka record-record memuat field-field data dan *link field*
3. Menggunakan Link semu (*pseudelink*)
   a. Metode computed chaining
   Dalam metode ini, maka record-record memuat field-filed data dan ditambah field untuk menghitung alamat indeks.

Coalesced Hashing

Metode *coalesced hashing* merupakan penyelesaian *collision* dengan menggunakan pointer sebagai penghubung (link) yang menghubungkan alamat asal (home address) ke alamat indeks yang ditunjuk berikutnya [Alan L. Tharp, 1988]. Penggunaan link ini akan membentuk suatu rantai record-record yang mempunyai home addres sama. Metode *coalesced hashing* mempunyai beberapa varian, diantarannya :

1. Tipe LISCH (*Late Insertion Standart Coalesced Hashing*)
2. Tipe EISCH (*Early Insertion Standart Coalesced Hashing*)

Tipe LISCH

Penyelesaian *collision* dengan metode coalesced hashing tipe ini dilakukan dengan cara sebagai berikut :

1. Dilakukan dengan menggunakan link
2. link menunjuk ke alamat kunci yang mengalami collision
3. nilai kunci yang mengalamai collision ditempatkan pada bagain akhir tabel
4. fungsi hash yang digunakan adalah H(K) = K MOD P, dimana
   P = bilangan prima terkecil dari jumlah kunci
   H = fungsi hash
   K = nilai kunci
5. P merupakan ukuran tabel index

Tipe EISCH

Penyelesaian *collision* dengan metode ini dilakukan dengan cara sebagai berikut :

1. Hampir sama dengan tipe LISCH
2. Perbedaannya adalah dalam tipe ini jika terjadi lebih dari 1 kali collision, maka nilai kunci terakhir yang mengalami collision tersebut akan ditunjuk langsung oleh home address

Progressive Overflow

Dalam metode *coalesed hashing* dibutuhkan media penyimpanan tambahan untuk membentuk /menyimpan *link field*. Pada saat media penyimpanan tidak mungkin untuk dilakukan penambahan lagi, maka metode tersebut merupakan pilihan tepat untuk digunakan [Alan L. Tharp, 1988].

Salah satu metode konvensional sederhana untuk mengatasi masalah tersebut adalah diselesaikan dengan menggunakan metode *progressive overflow / liniear probing*. Dalam metode ini, jika suatu kunci mengalami *collision*, maka nilai kunci tersebut akan diletakkan pada lokasi/indeks selanjutnya yang masih kosong. Jika hingga indeks terakhir sudah penuh, maka pencarian lokasi dilakukan mulai dari awal tabel, sehingga membentuk struktur tabel/indeks *circular*.

Algoritma metode *progressive overflow/liniear probling* adalah sebagai berikut :

1. Konversikan nilai kunci (hashing) untuk mendapatkan home address untuk menempatkan kunci tersebut pada tabel /indeks
2. Jika *home address* kosong, maka sisipkan kunci pada lokasi tersebut. Jika tidak kosong, maka lakukan proses berikut
   a. Tentukan nilai penambahan (*increment*) dengan cara mencari sisa bagi kunci dengan ukuran tabel, jika hasilnya 0 (nol) set nilai variabel *increment* dengan 1.
   b. Set nilai pencacah (counter) untuk pencarian lokasi kosong dengan nilai 1
   c. Selama nilai pencacah (*counter*) kurang dari ukuran tabel, maka lakukan proses berikut :
      a. Hitung lokasi selanjutnya dengan cara menambahkan nilai/indeks alamat sekarang dengan nilai *increment* dan kemudian hasilnya di modulo dengan ukuran tabel
      b. Jika alamat kosong, sisipkan kunci pada lokasi tersebut jika tidak kosong, tambahkan nilai pencacah (counter) dengan 1 dan kembali ke langkah -1
   d. Akhiri dengan komentar "Tabel Penuh"

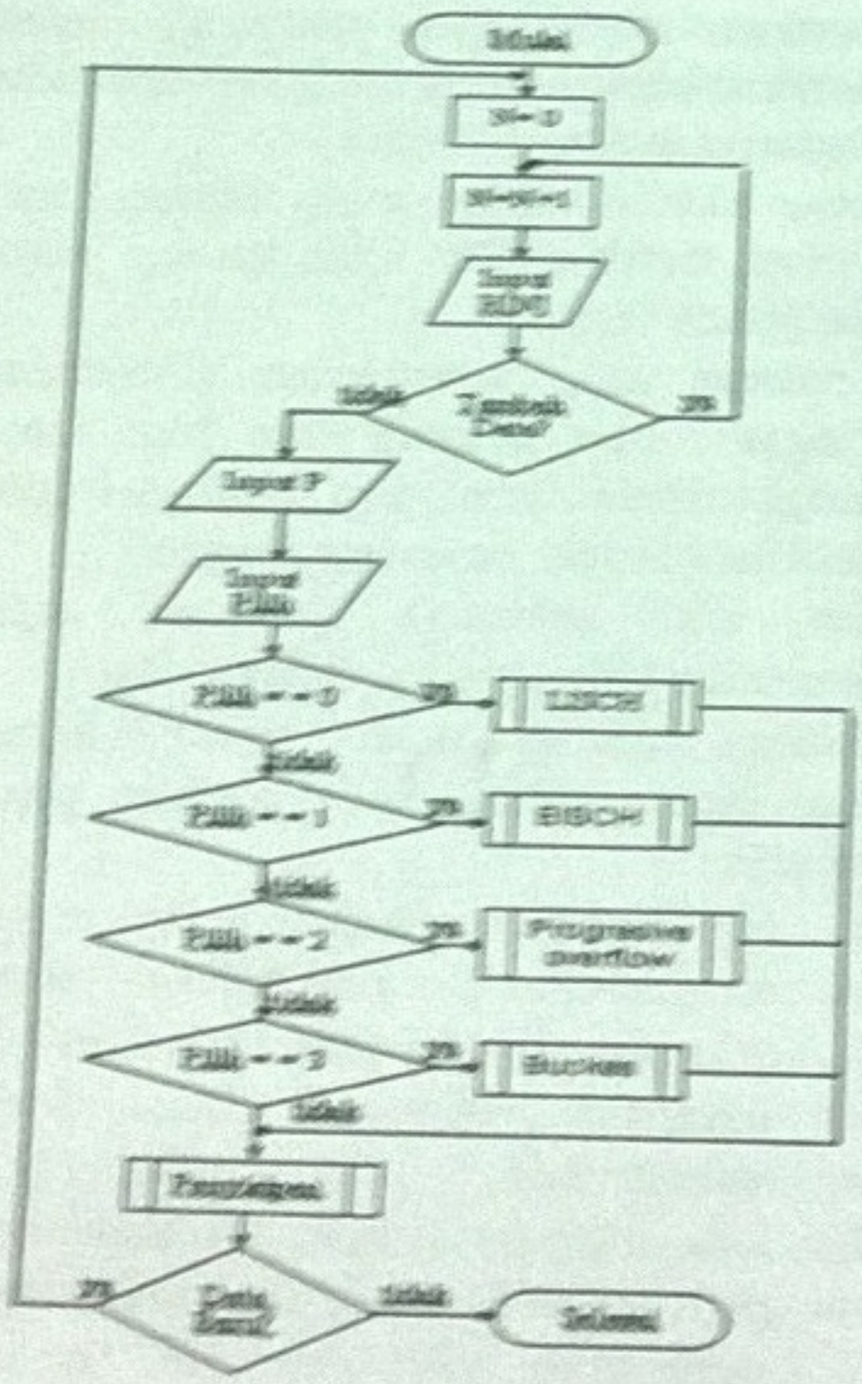
Metode Buckets

Pembicaraan metode *hashing* sebelumnya mengasumsikan bahwa hanya ada sebuah record yang mungkin menempati ada media penyimpan. Banyaknya jumlah pengaksesan ke media penyimpan sekunder dapa diminimalkan dengan teknik penyimpanan multirecord. Jika hal ini dilakukan, maka penanggannanya akan dillakukan oleh blok = halaman = buckets.

Buckets [Edhy Sutanta, 2004] merupakan sebuah unit yang dapat digunakan kenika data/record disimpan ke atau diakses dari media penyimpanan sekunder. *Buckets* memuat lokasi indeks tidak sebenarnya dari record, tetapi sebagai indeks ke dalam suatu array pointer. Setiap bucket terdiri atas satu untai record yang bersama-sama menempati alamat hash yang sama. Jumlah record yang dapat disimpan dalam array pointer / untai indeks disebut sebagai faktor blok (blocking faktor), sama dengan banyaknya bucket. Semakin besar nilai faktor block maka semakin kecil kemungkinan terjadinya *collision*, karena record-record yang mengalami collision dapat disimpan ke dalam satu lokasi /indeks.

Perancangan

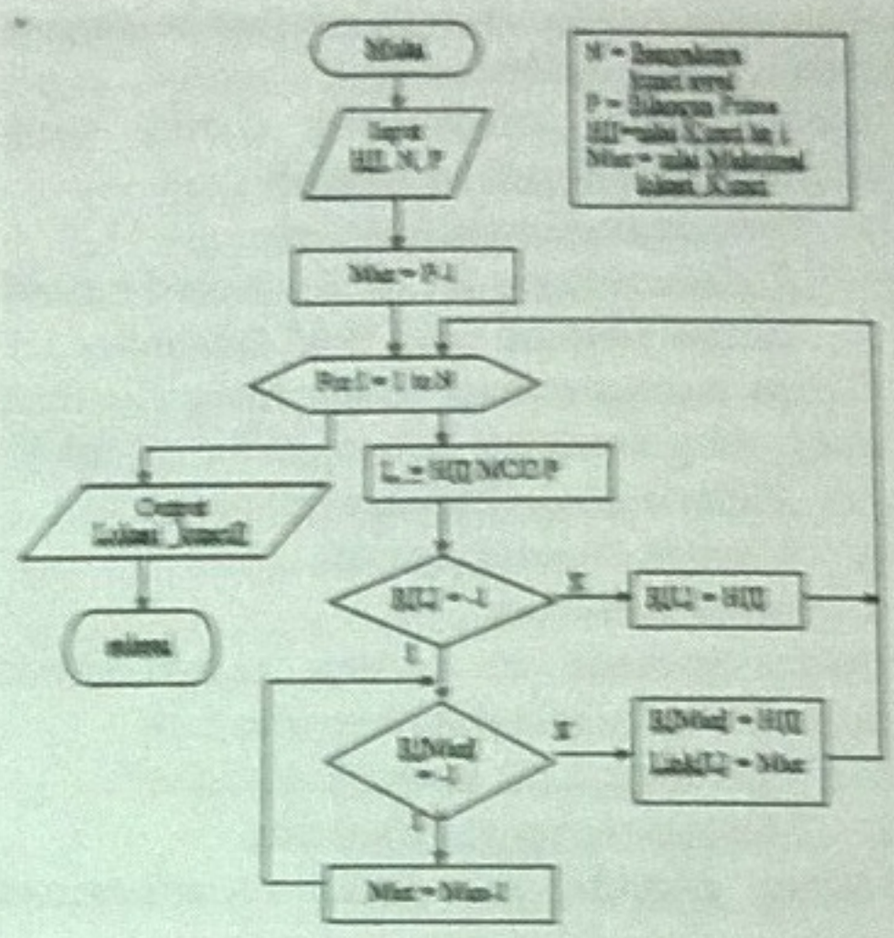
Dari dasar teori yang ada disusun Bagan alir program (*Flowchart*) sebagai berikut



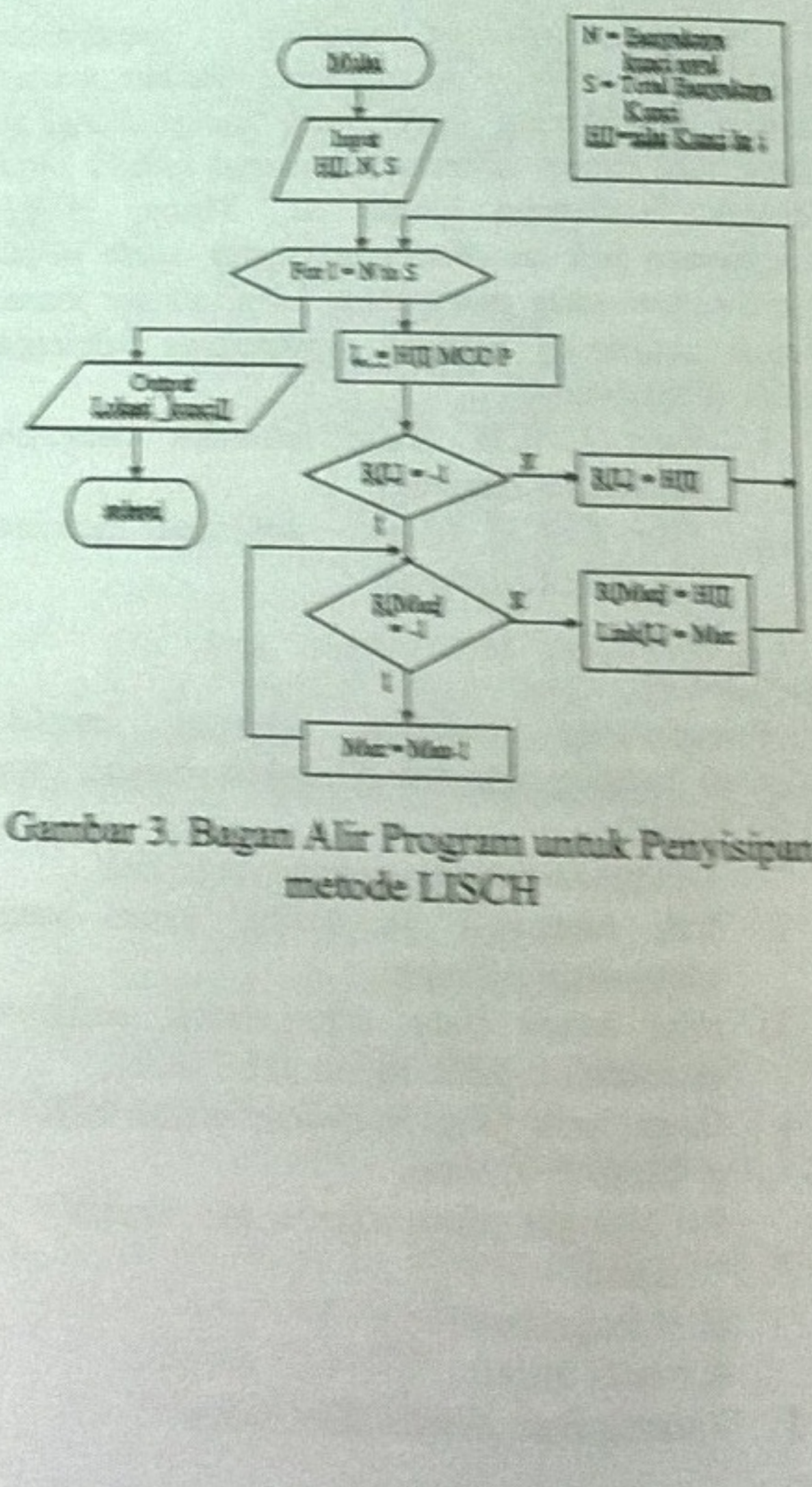
Gambar 1. Bagan Alir Program

Coalesced Hashing

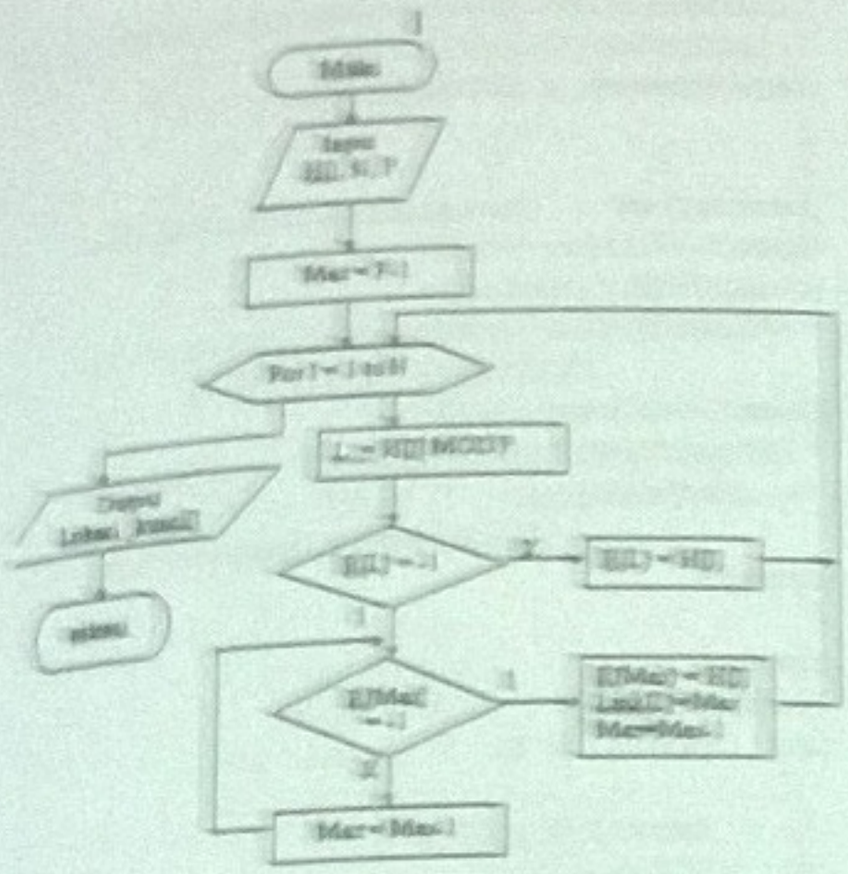
Bagan alir program (*Flowchart*) untuk algoritma metode *Coaleseced Hashing* disusun adalah sebagai berikut :



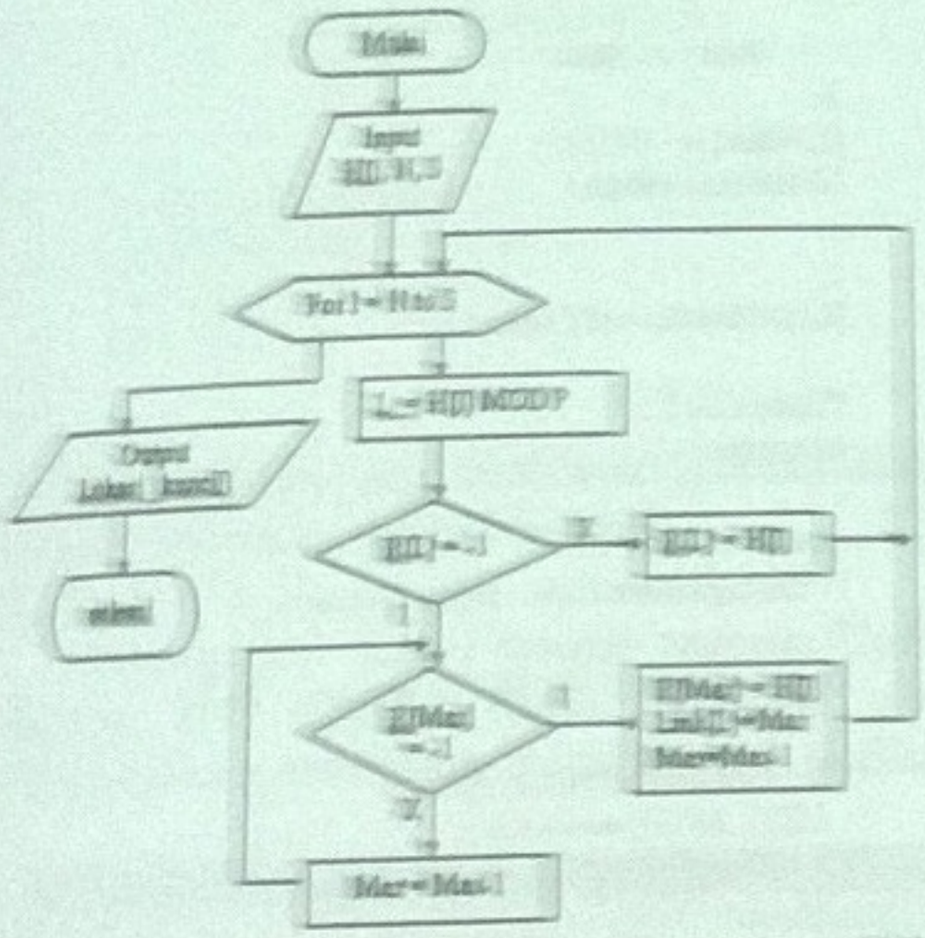
Gambar 2. Bagan Alir Program untuk metode LISCH



Gambar 3. Bagan Alir Program untuk Penyisipan metode LISCH



Gambar 4. Diagram Alir Program Untuk metode EISCH

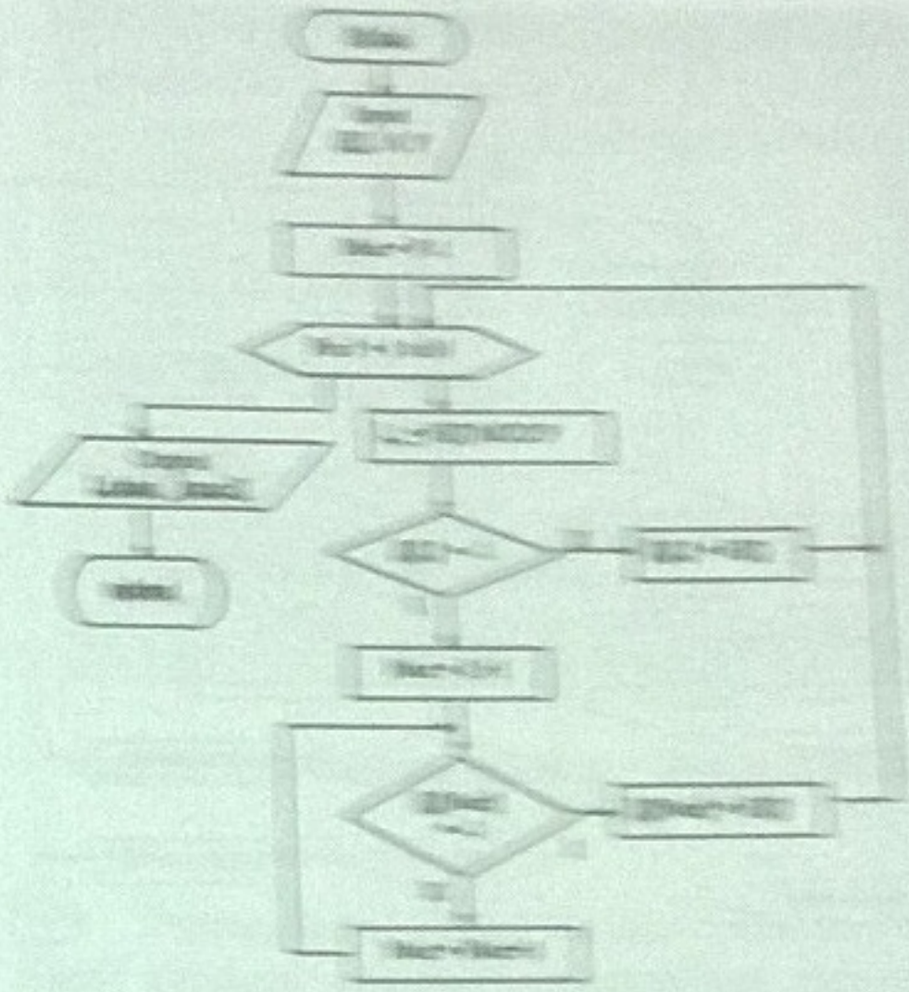


Gambar 5. Bagan Alir Program Untuk Penyisipan metode EISCH

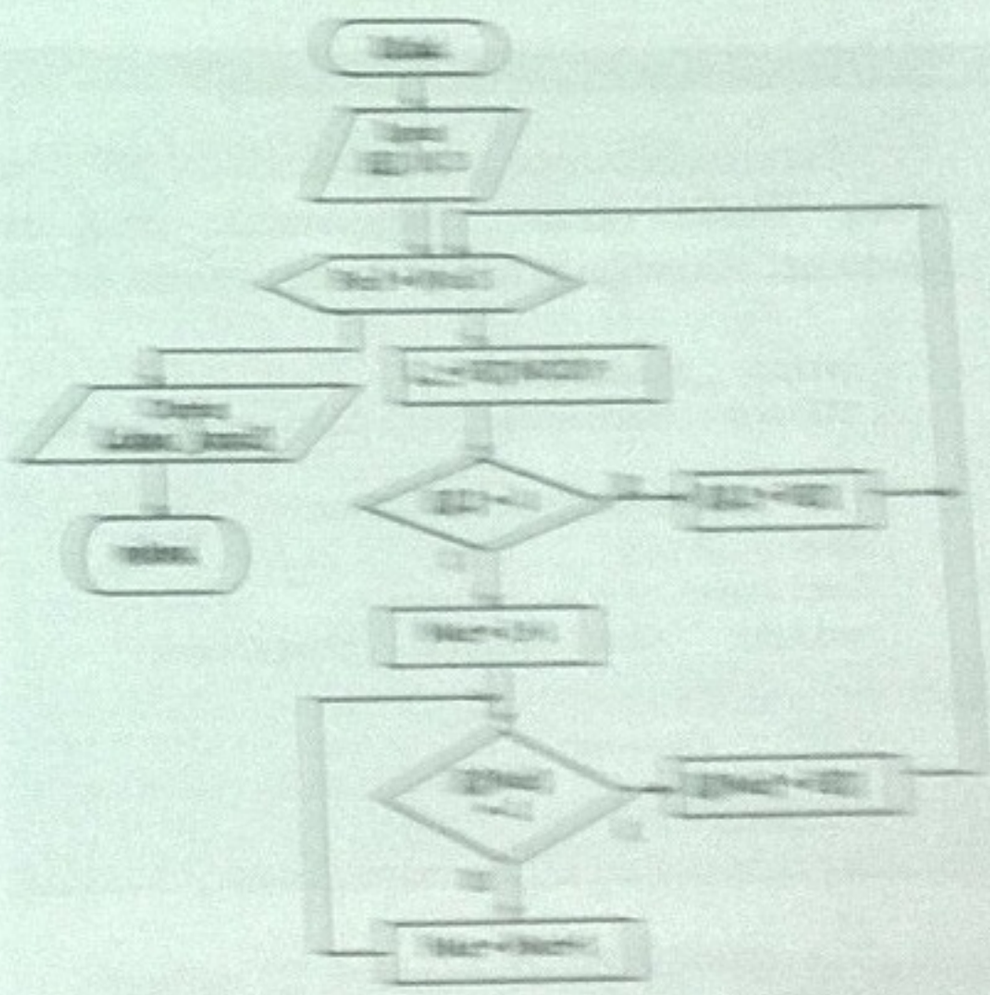
Dalam Bagan alir diatas kebutuhan input untuk penyelesaian masalah ini adalah nilai kunci berupa array, N yaitu banyak nya kunci dan P untuk bilangan prima terkecil, digunakan untuk alokasi lokasi kunci.

Progressive Overflow

Bagan alir program untuk algoritma metode *progressive overflow* disusun adalah sebagai berikut :



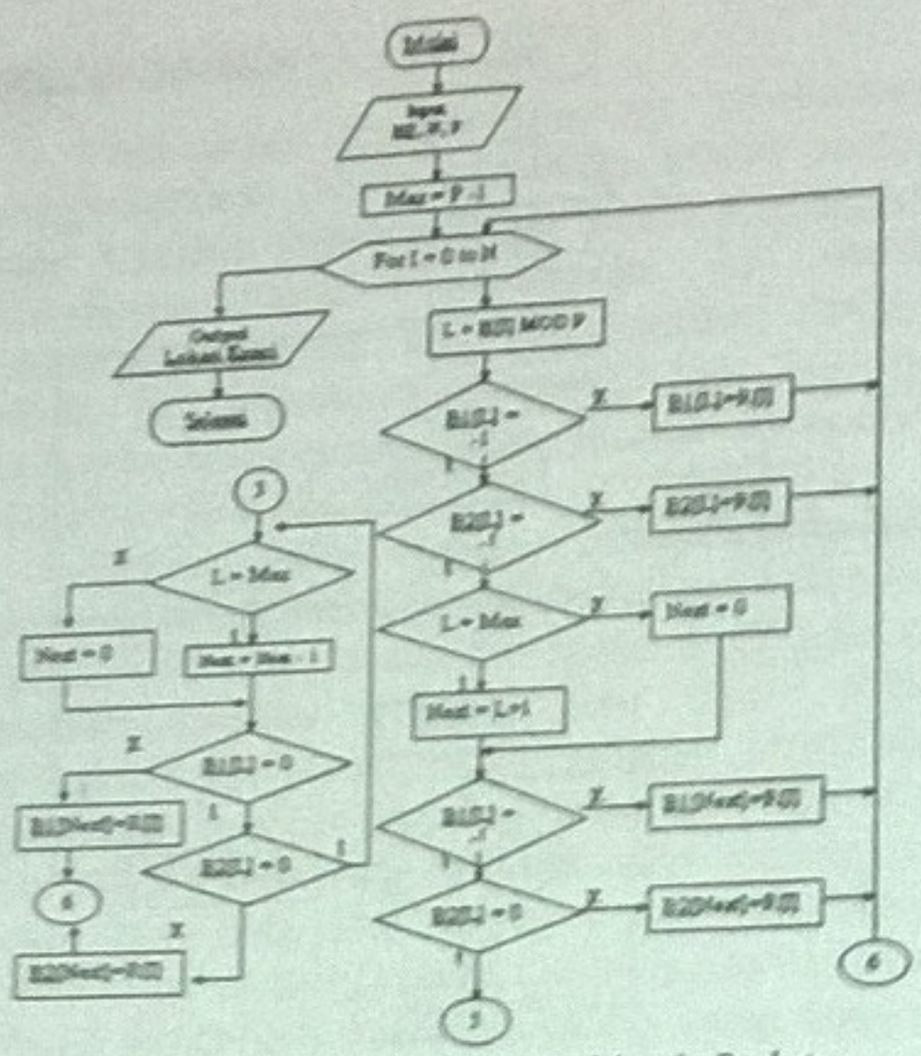
Gambar 6. Bagan Alir Program Metode Progressive Overflow



Gambar 7. Bagan Alir Program Penyisipan Metode Progressive Overflow

Bucket

Diagram alir program untuk algoritma metode Bucket disusun adalah sebagai berikut :



Gambar 8. Bagan Alir Program Metode *Buckets*

HASIL DAN PEMBAHASAAN

Pertama kita inputkan Angka Kunci (H[]), nilai P dan dipilih metode penyelesaian yang akan digunakan. Potongan Kode program sebagai berikut

```
void __fastcall
TForm1::Button9Click(TObject *Sender)
{
P = StrToInt(Edit2->Text);
Max = P-1;
int next = 0;
switch  (RadioGroup1->ItemIndex)
{
case 0 :
{
//LISCH();
for (int i = 1; i <= n; i++)
 {
 L = fmod(H[i],P);
if(R[L]==(-1) )
   R[L]= H[i];
else
{
while (R[Max]!=(-1))
{
   Max = Max - 1;
}
R[Max]= H[i];
Link[L]=Max;
}
}
ListBox1->Clear();
Tampil();
for (int i = 0; i <= P-1; i++)
{
if (Link[i]==(-1))
{
ListBoxLink->Items->Add(IntToStr(i)+"
");
akses = akses + 1;
}
else
{
ListBoxLink->Items->Add(IntToStr(i)+"
"+IntToStr(Link[i]));
penempatan = penempatan + 1;
}
}
pencarian = (penempatan+akses)/n;
Memo1->Lines->Add("=( " +
FloatToStr(akses)+" +
"+FloatToStr(penempatan)+" ) / "+
          IntToStr(n));
Memo1->Lines->Add("=
"+FloatToStr(penempatan + akses)+" /
"+ IntToStr(n)+ " = "
          + FloatToStr(pencarian));
break;
}
case 1 :
(//EISCH();
for (int i = 1; i <= n; i++)
 {
 L = fmod(H[i],P);
if( R[L]==(-1))
   R[L]= H[i];
else
{
while (R[Max]!=(-1))
{
   Max = Max - 1;
}
R[Max]= H[i];
Link[L]=Max;
}
}
ListBox1->Clear();

Tampil();
break;

case 2 :
//progressive overflow
{
for (int i = 1; i <= n; i++)
 {
L = fmod(H[i],P);
if( R[L]==(-1))
  R[L]= H[i];
else
{
  next = L + 1;
  while (R[next]!=(-1))
  {
  next = next + 1;
  }
  R[next]= H[i];
  }
}
ListBox1->Clear();

Tampil();
break;

case 3 :;
//buckets
 {
for (int i = 1; i <= n; i++)
 {
 L = fmod(H[i],P);
if( R[L]==(-1))
   R[L]= H[i];
else
{
if( Bucket1[L]==(-1))
```

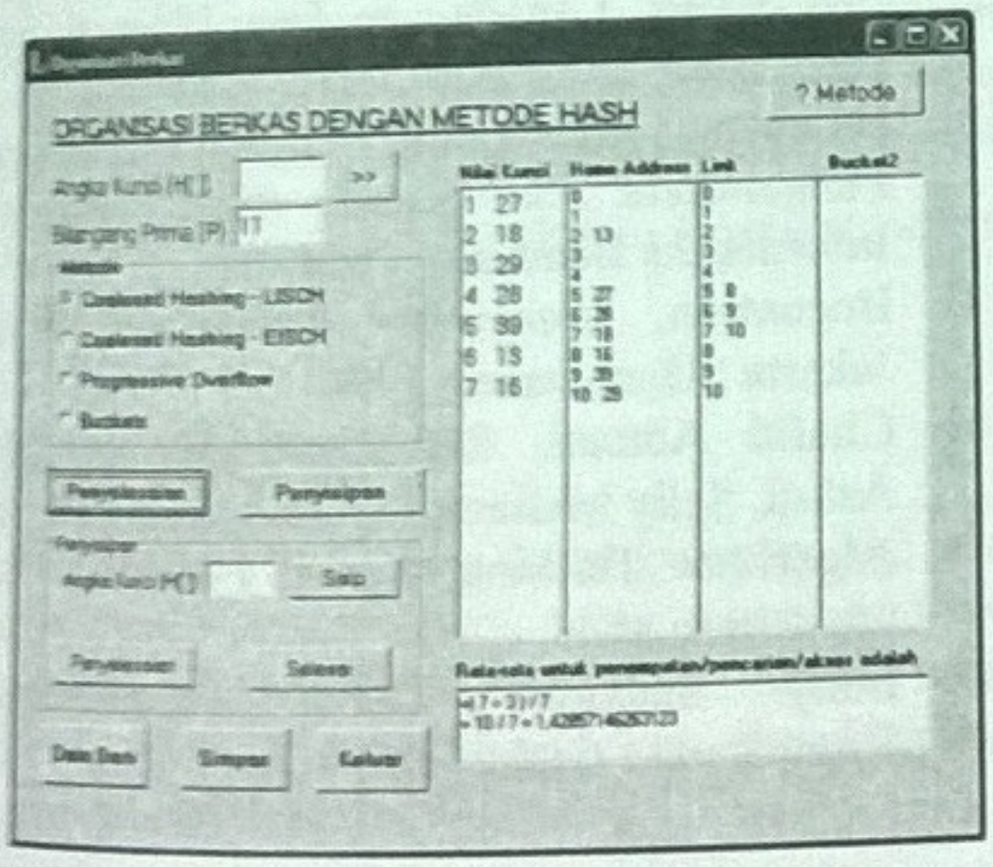
```
   Bucket1[L]= H[i];
else
if( Bucket2[L]==(-1))
   Bucket2[L]= H[i];
else
{
 if (L==Max)
   next = 0;
 else
   next = L + 1;

 while ((R[next]!=(-1)) ||
(Bucket1[next]!=(-1))
||(Bucket1[next]!=(-1)))
   {
   next = next + 1;
   if (next>Max)
      next = 0;
   }
   if  (R[next]==(-1))
      R[next]= H[i];
   else
   if (Bucket1[next]==(-1))
      Bucket1[next]=H[i];
   else
   if (Bucket2[next]==(-1))
       Bucket2[next]=H[i];

   }
 }
 ListBox1->Clear();
 ListBoxLink->Clear();
 ListBox2->Clear();
 Tampil();
}}
```

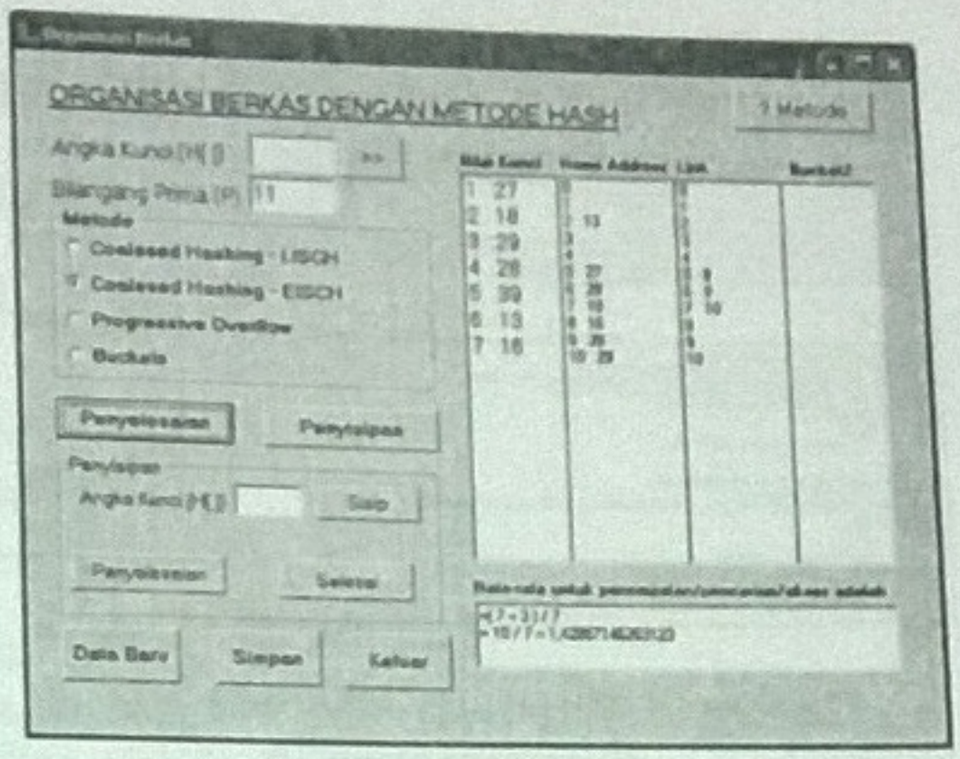
Dari aplikasi yang telah buat kita masukan data dengan nilai kunci sebagai berikut : 27, 18, 29, 28, 39, 13, 16

Penempatan kunci dengan metode Coalesced Hashing tipe LISCH. Hasil eksekusi program menghasilkan penempatan kunci sebagai berikut :



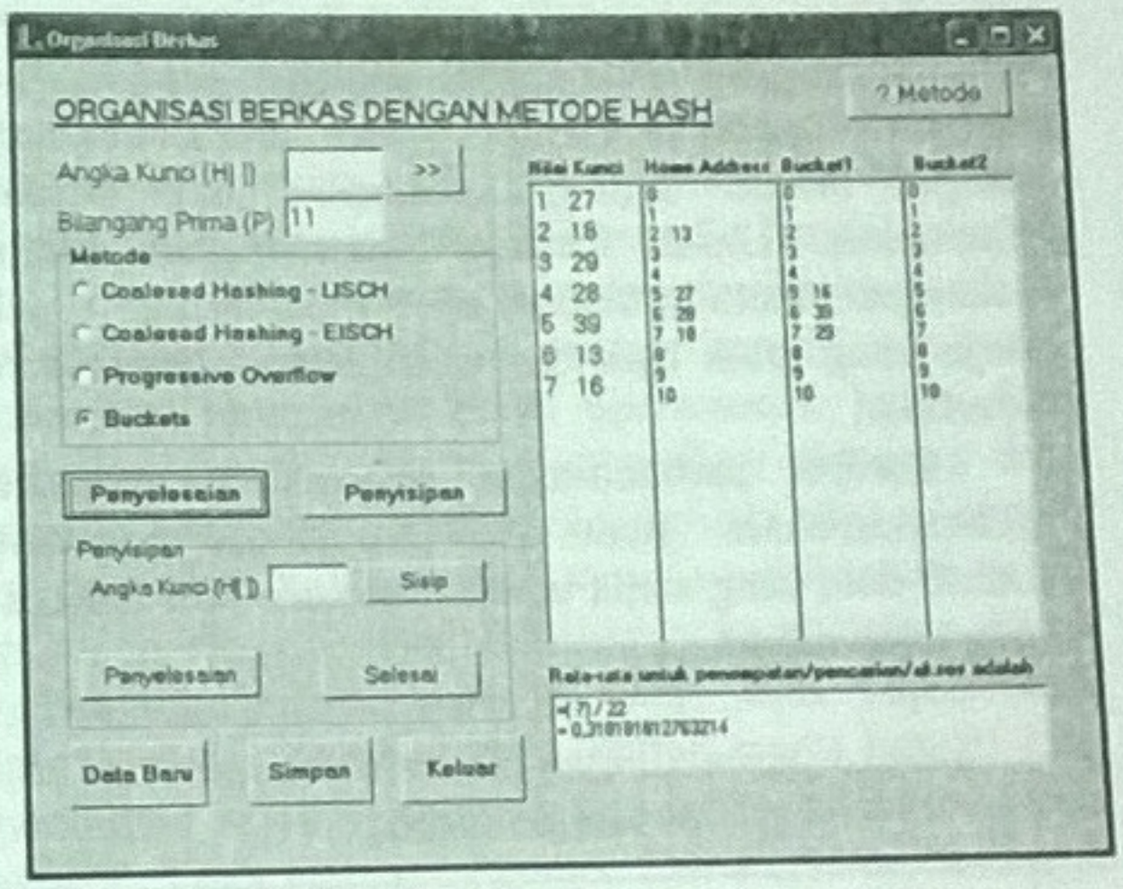
Gambar 9. Penempatan Kunci dengan mentode LISCH

Penempatan kunci dengan metode Coalesced Hashing tipe EISCH Hasil eksekusi program menghasilkan penempatan kunci sebagai berikut :



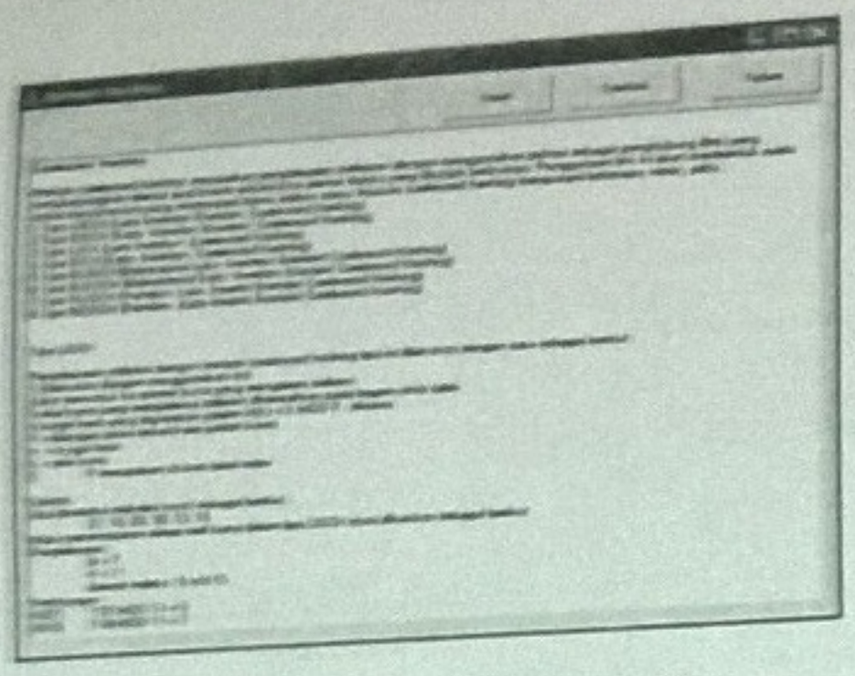
Gambar 10. Penempatan Kunci dengan mentode EISCH

Penempatan kunci dengan metode Buckets Hasil eksekusi program menghasilkan penempatan kunci sebagai berikut :



Gambar 12. Penempatan Kunci dengan metode Bucket

Selain itu juga tombol metode dapat digunakan untuk menampilkan penjelasan tentang metode yang digunakan dengan memilih salah satu metode yang ada pada komponen radiogroup, contoh hasil eksekusi sebagai berikut :



Gambar 13. Penjelasan Algoritma metode Coalesced Hashing

Jika dibandingkan antara tipe LISCH dan EISCH, maka tipe EISCH relative lebih efisien.
Secara umum, kelebihan metode *progressive overflow / linier probling* adalah:

1. Tidak memerlukan ruang tambahan
2. Algoritma sederhana

Sedangkan kelemahan metode *progress-ive overflow/liniear probling* adalah untuk kerjanya sangat buruk, sehingga hampir tidak pernah diterapkan. Untuk metode bucket secara relatif penerapan metoda bucket akan memberikan unjuk kerja yang lebih baik dari pada metode progresive overflow.

Berikut perbandingan rata-rata penempatan /pencarian/akses suatu nilai kunci dari beberapa contoh data yang sama untuk 3 metode dari aplikasi yang telah dibuat.

Tabel 1 perbandingan rata-rata penempatan/ pencarian /akses

Data ke	Metode		
	Coalesced Hashing	Progresive Overflow	Buckets
1	1,43	2,29	0,32
2	1,70	3,20	0,45
3	1,88	22,44	0,40
4	1,82	17,45	0,42
5	1,70	1,89	0,45
6	1,83	16,25	0,46
7	1,50	3,50	0,36
8	1,53	0,13	0,44
9	1,70	4,47	0,45
10	1,63	10,47	0,41
Rata-rata	**1,67**	**8,21**	**0,42**

Dari tabel 1 menunjukkan bahwa metode buckets memberikan nilai rata-rata penempatan/pencarian/akses suatu nilai terkecil dilanjutkan coalesced hashing kemudian metode progresive overflow. Semakin kecil rata-rata penempatan/pencarian/akses suatu nilai maka semakin kecil kemungkinan terjadinya *collision* atau rata-rata waktu pencarian nilai suatu kunci dalam berkas berbanding lurus dengan kemungkinan terjadinya *collision*.

KESIMPULAN

Kesimpulan dari penelitian ini adalah dengan mengetahui metode yang digunakan dapat dipilih metode yang tepat untuk menyelesaikan masalah penyimpanan data, terutama penempatan nilai kunci yang mengakibatkan terjadinya tumbukan / *coalission* dengan metode *Coalesced Hashing, Prograssive Overflow, Buckets*. Aplikasi yang telah dibuat dapat memberikan hasil penempatan kunci ke home address dengan metode yang telah ditentukan. Dari tiga metode yang di gunakan metode buckets memberikan nilai rata-rata penempatan /pencarian/akses terkecil hal ini menunjukkan bahwa kemungkinan terjadinya collsioan atau tumbukan paling kecil. Aplikasi juga dapat menyajikan metode atau algoritma dari metode *Coalesced Hashing, Prograssive Overflow, Buckets* sehingga dapat digunakan sebagai media pembelajaran.

DAFTAR PUSTAKA

[1]. Alan L.Tharp, *"File Organiztion and Processing"*,Canada : by John Wiley & Sons, Inc, 1988.

[2]. Bambang Hariyanto, Ir..,M.T., *"Sistem Pengarsipan dan Metode Akses"*, Bandung: Informatika Bandung, 2000.

[3]. Boenawan, *"Pengantar Berkas dan Akses"*, Jakarta : Gunadarma, 1994.

[4]. Chafid Ahmad, Rina Andriani, Rini Diah Astuti, Selly Meliana, "Pengukuruan Kinerja Algoritme Hashing Dalam Pencarian Kata, www.cs.ui.ac.id

[5]. Edhy Sutanta, *"Sistem Basis Data"*, Yogyakarta : Graha Ilmu, 2004.