JURNAL INFORMATIKA Vol. 8, No. 2, Juli 2014

HASHTABLE SEBAGAI ALTERNATIF DARI ALGORITMA PENCARIAN BINER PADA APLIKASI E-ACESIA

Viska Mutiawani

Jurusan Informatika, FMIPA, Universitas Syiah Kuala

Universitas Syiah Kuala, Jl. Syech Abdurrauf No.3, Darussalam, Banda Aceh 23111

viska@informatika.unsyiah.ac.id

ABSTRAK

Aplikasi e-Acesia merupakan kamus dwibahasa Aceh-Indonesia yang dapat digunakan pada telepon genggam berbasis Java MIDP (Mobile Information Device Profile). Aplikasi kamus ini menyimpan data berupa kata dan terjemahannya dalam file teks. Proses utama pada kamus adalah proses pencarian. Aplikasi ini mencoba dua jenis pencarian yaitu pencarian biner dan pencarian pada struktur data Hashtable. Kedua algoritma ini dipilih karena data kamus yang terurut dan tetap serta algoritmanya mudah diimplementasikan pada Java MIDP yang memiliki jumlah Class terbatas. Pengujian terhadap kedua-dua algoritma menggunakan file teks berisi jumlah kata 1000, 2000, 3000 dan 4000 kata. Pengujian pada emulator di komputer menghasilkan waktu pencarian yang sama untuk kedua-dua algoritma yaitu 0 milidetik. Sedangkan pengujian pada telepon genggam dengan menggunakan algoritma pencarian biner menghasilkan waktu 0 milidetik untuk 1000 kata, 0.042 milidetik untuk 2000 kata dan 0.125 milidetik untuk 3000 dan 4000 kata. Sebaliknya waktu pencarian pada telepon genggam dengan menggunakan struktur data Hashtabel menghasilkan waktu rata-rata pencarian yang konstan yaitu 0 milidetik. Namun demikian ukuran milidetik adalah sangat kecil dan tidak terdeteksi oleh pengguna aplikasi. Selain waktu pencarian, pengujian juga mendata besarnya ukuran file jar. Ternyata ukuran file jar bertambah berdasarkan jumlah kata yang disimpan dalam file teks dan ukurannya sama untuk kedua-dua algoritma. Struktur data Hashtable ternyata dapat menjadi alternatif struktur data dan algoritma pada aplikasi kamus e-Acesia karena waktu pencarian yang konstan dan dapat menampung data yang lebih banyak berbanding dengan struktur data array pada pencarian biner.

Kata kunci : *aplikasi kamus, telepon genggam, Java MIDP, pencarian biner, Hashtable*

1. PENDAHULUAN

Kamus dwibahasa Aceh-Indonesia e-Acesia merupakan kamus yang digunakan untuk menterjemahkan kata dari bahasa Aceh ke bahasa Indonesia maupun sebaliknya dengan menggunakan telepon genggam berbasis Java MIDP (*Mobile Information Device Profile*). Pembangunan kamus ini didasari atas keprihatinan terhadap berkurangnya penggunaan bahasa Aceh terutama di perkotaan (Razi 2007). Kamus ini dibangunkan dengan harapan dapat digunakan dalam proses pembelajaran bahasa Aceh terutama di tingkat sekolah. Karena menurut Al-Gayoni (2010), keberadaan bahasa Aceh yang mulai pudar penggunaannya dapat sedikit dicegah dengan memperluas penggunaannya terutama di ranah informal. Kamus e-Acesia menggunakan pedoman dari kamus berbentuk buku

yaitu kamus Aceh Indonesia yang dirangkum oleh Bakar dkk (1985) dan kamus Indonesia-Aceh yang dirangkum oleh Basry (1994).

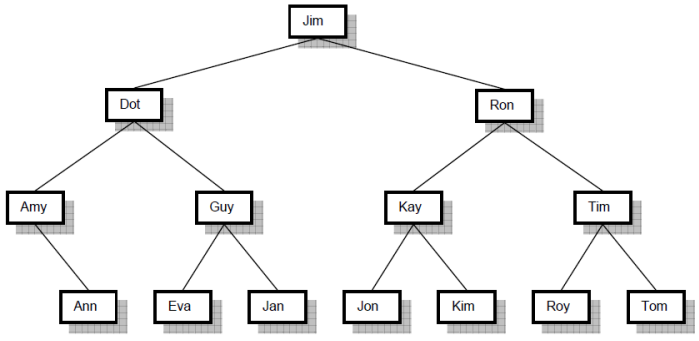
Kamus yang dibuat merupakan sumber kosakata dan penjelasan maknanya. Proses utama pada kamus adalah proses pencarian. Proses pencarian biner sebenarnya telah dibahas di Mutiawani (2011), walau tidak terlalu mendalam. Namun artikel ini akan membahas dan melakukan perbandingan pada dua algoritma yang dipilih yaitu algoritma pencarian biner dan algoritma pencarian pada struktur data Hashtable. Kedua algoritma ini dipilih karena dapat diimplementasikan pada telepon genggam yang menggunakan profil MIDP pada Java Edisi Mikro (*Java Micro Edition/Java ME*).

2. LANDASAN TEORI

2.1. Algoritma Pencarian Biner

Pencarian biner (*Binary Search*) adalah metode pencarian data pada array yang telah terurut. Metode ini lebih efisien daripada metode pencarian linier dimana semua elemen di dalam array diuji satu per satu sampai ditemukan elemen yang diinginkan. Selain dari pencarian biner, terdapat juga pencarian interpolasi (*interpolation search*), pencarian lompat (*jump search*), yang sama-sama bekerja pada data yang terurut. Pencarian pada data yang terurut menghasilkan pencarian yang cepat. Pencarian interpolasi mempunyai kekompleksan waktu rata-rata adalah O (log log n), sedangkan pencarian lompat adalah $O(kn^{1/(k+1)})$. Kekompleksan waktu untuk pencarian biner adalah O (log n) seperti yang dikemukakan oleh Knuth (1998).

Pada pencarian biner, data harus dalam keadaan terurut. Proses pencarian bermula dengan membagikan array menjadi dua. Jika data yang dicari lebih kecil dari data yang terletak di tengah-tengah, maka proses pencarian akan dilanjutkan ke sebelah kiri dengan cara membagi array sebelah kirim menjadi dua. Sebaliknya jika data yang dicari lebih besar dari data yang terletak di tengah, maka proses pencarian akan dilanjutkan ke sebelah kanan dengan kembali membagi array menjadi dua bagian dan mencari titik tengahnya. Proses pembagian akan terus berulang hingga data yang dicari ditemukan. Proses algoritma pencarian biner dapat dideskripsikan melalui *tree* seperti pada gambar 1.



Gambar 1. Contoh *Tree* dari Algoritma Pencarian Biner

Pada contoh berikut ini, sebuah algoritma pencarian biner akan mencari sebuah bilangan integer 5 dalam sebuah array yang terurut yang berukuran 10 slot.

2	5	6	8	10	12	15	18	20	21

Langkah 1 : Lihat keseluruhan array, identifikasikan sebuah indeks paling bawah = 0 dan indeks paling tinggi adalah 9, Pilih elemen dengan indeks (0+9)/2 = 4

2	5	6	8	10	12	15	18	20	21

Bandingkan nilai 10 dengan target yaitu 5. Karena 10 lebih besar dari pada 5 maka target akan berada pada sebelah kiri dari nilai 10. Lalu aturkan indeks paling tinggi adalah (4-1) = 3

Langkah 2 : Identifikasikan indeks paling bawah = 0 dan indeks paling atas = 3. Pilih elemen pada indeks (0+3)/2 = 1.

2	5	6	8	10	12	15	18	20	21

Bandingkan nilai 5 dengan nilai target yaitu 5. Karena 5 sama dengan nilai target 5 maka nilai target telah ditemukan yaitu pada indeks 1.

Algoritma pencarian biner dapat dituliskan sebagai berikut:

L ← 0

R ← N – 1

Ketemu ← false

Selama (L <= R) dan (tidak ketemu) kerjakan

m ← (L + R) / 2

Jika (Data[m] = x) maka ketemu ← true

Jika (x < Data[m]) maka R ← m – 1

Jika (x > Data[m]) maka L ← m + 1

Jika (Ketemu) maka m adalah indeks dari data yang dicari, jika tidak data tidak ditemukan.

Pada kamus e-Acesia, data kamus disimpan dalam file teks yang kemudian dibaca dan setiap kata dan terjemahannya disimpan dalam array. Data yang tersimpan telah terurut sehingga proses pencarian biner dapat dengan mudah dilakukan.

2.2. Algoritma Hash

Algoritma Hash dapat melakukan proses tambah, hapus dan pencarian dalam waktu yang konstan (Weiss, 2012). Waktu konstan ini juga disebut sebagai kekompleksan waktu O(1). Teknik Hash merupakan suatu metode yang secara langsung mengakses data *record* dalam suatu tabel dengan melakukan penghitungan pada *key* yang menjadi alamat *record* pada tabel. *Key* merupakan suatu data yang unik dapat berupa nomor atau karakter string.

Proses Hash terdiri dari dua langkah yaitu:

a. **Menghitung fungsi Hash**. Fungsi Hash adalah suatu fungsi yang mengubah *key* menjadi alamat dalam tabel dengan memetakan *key* tersebut ke alamat dalam tabel. Secara teori, *key-key* yang berbeda haruslah dipetakan ke alamat-alamat yang berbeda pula. Namun kenyataannya ada kemungkinan dua atau lebih *key* yang dipetakan ke alamat yang sama dalam tabel. Peristiwa ini disebut dengan tabrakan (*collision*).
b. **Resolusi terhadap tabrakan (collision resolution)**. Resolusi terhadap tabrakan merupakan proses untuk menangani kejadian dua atau lebih *key* yang dipetakan ke alamat yang sama pada tabel. Caranya yaitu dengan mencari lokasi yang kosong dalam tabel Hash secara terurut. Atau bisa juga dengan menggunakan fungsi Hash yang lain untuk mencari lokasi yang kosong.

Algoritma Hash pada java MIDP ada dalam class Hashtable. Class Hashtable ini berjalan sebagaimana konsep algoritma Hash. Data *record* yang ingin dimasukkan dalam Hashtable harus memiliki *key*. Semua objek bukan *null* dapat menjadi *key*. Jadi setiap unsur pada Hashtable adalah pasangan *key* dan *value*.

Data kamus e-Acesia yang tersimpan dalam file teks akan dibaca dan disimpan dalam Hashtable. *Key* yang digunakan adalah kata, sedangkan *value* merupakan terjemahannya. Kata yang tersimpan tidak ada yang sama sehingga *collision* diharapkan tidak terjadi. Apalagi data sudah tetap dan tidak ada penambahan selama aplikasi kamus ini dijalankan.

3. METODE PENGUJIAN

Pengujian terhadap pencarian biner dan pencarian dengan menggunakan struktur data Hashtable menggunakan data kamus file teks yang telah ada. File teks yang digunakan adalah file teks yang berisi kata dalam bahasa Indonesia dan terjemahannya dalam bahasa Aceh. Karena data kata-kata yang dimasukkan dalam file ini telah mencapai 4000 kata. Hal-hal yang akan diuji adalah:

1. Jumlah kata dalam file teks yang mampu diproses oleh masing-masing pencarian. Jumlah kata pada kamus akan dimulai dari 1000 dan naik sebanyak 1000 hingga jumlah katanya 4000.
2. Ukuran file jar yang terhasil untuk setiap pengujian terhadap jumlah kata yang berbeda.
3. Waktu dalam melakukan pencarian terhadap 24 kata bahasa Indonesia yaitu ada, babak, cabai, daerah, edan, faedah, gabah, habis, ia, jadi, kabar, laba, maaf, nada, obat, pabrik, raba, saat, taat, uang, vakum, wabah, ya, zaman. Pencarian dibedakan menjadi dua lingkungan yaitu emulator telepon genggam pada komputer dan telepon genggam sebenarnya. Pencarian juga dibedakan berdasarkan algoritma yang dipakai. Selain itu pencarian juga dibedakan berdasarkan jumlah kata yang tersimpan dalam file teks.
4. Penggunaan file teks *dummy* hingga 14000 kata untuk menguji kemampuan maksimal penyimpanan data oleh struktur data array pada pencarian biner dan struktur data Hashtable.

4. HASIL DAN PEMBAHASAN

4.1. Hasil Penelitian

Prototipe aplikasi kamus e-Acesia diuji dengan menjalankannya di dua tempat yaitu: emulator telepon genggam pada komputer dan telepon genggam sebenarnya. Contoh hasil pencarian kata dapat dilihat pada gambar 2.

Berdasarkan metode pengujian yang telah dijelaskan sebelumnya, maka hasil pencarian dibedakan sesuai lingkungan pengujian, jumlah kata dan algoritma pencarian. Waktu pencarian untuk setiap algoritma didapat dengan menghitung selisih waktu antara sebelum dan sesudah proses pencarian. Detil hasil pencarian pada lingkungan emulator telepon genggam di komputer dapat dilihat di tabel 1.

Gambar 2. Contoh Tampilan Hasil Pencarian

Tabel 1. Hasil Pencarian pada Emulator Telepon Genggam di Komputer

Kata	Pencarian Biner di Emulator Telepon Genggam				Pencarian pada Hashtable di Emulator Telepon Genggam			
	Waktu Pencarian untuk Jumlah Kata (milidetik)							
	1000	2000	3000	4000	1000	2000	3000	4000
Ada	0	0	0	0	0	0	0	0
Babak	0	0	0	0	0	0	0	0
Cabai	0	0	0	0	0	0	0	0
Daerah	0	0	0	0	0	0	0	0
Edan	0	0	0	0	0	0	0	0
Faedah	0	0	0	0	0	0	0	0
Gabah	0	0	0	0	0	0	0	0
Habis	0	0	0	0	0	0	0	0
Ia	0	0	0	0	0	0	0	0

Jadi	0	0	0	0	0	0	0	0
Kabar	0	0	0	0	0	0	0	0
Laba	0	0	0	0	0	0	0	0
Maaf	0	0	0	0	0	0	0	0
Nada	0	0	0	0	0	0	0	0
Obat	0	0	0	0	0	0	0	0
Pabrik	0	0	0	0	0	0	0	0
Raba	0	0	0	0	0	0	0	0
Saat	0	0	0	0	0	0	0	0
Taat	0	0	0	0	0	0	0	0
Uang	0	0	0	0	0	0	0	0
Vakum	0	0	0	0	0	0	0	0
Wabah	0	0	0	0	0	0	0	0
Ya	0	0	0	0	0	0	0	0
Zaman	0	0	0	0	0	0	0	0
Rata-Rata Waktu Pencarian	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Semua jenis telepon genggam yang dapat menjalankan Java MIDP bisa dipakai. Namun pada percobaan di lingkungan telepon genggam, contoh telepon genggam yang dipakai adalah Sony Ericsson G705. Detil hasil pencarian dapat dilihat di tabel 2.

Tabel 2. Hasil Pencarian pada Telepon Genggam

Kata	**Pencarian Biner di Telepon Genggam**				**Pencarian pada Hashtable di Telepon Genggam**			
	Waktu Pencarian untuk Jumlah Kata (milidetik)							
	1000	**2000**	**3000**	**4000**	**1000**	**2000**	**3000**	**4000**
ada	0	0	0	1	0	0	0	0
babak	0	0	0	0	0	0	0	0
cabai	0	0	0	0	0	0	0	0

daerah	0	0	0	0	0	0	0	0
edan	0	0	0	0	0	0	0	0
faedah	0	0	0	0	0	0	0	0
gabah	0	0	0	1	0	0	0	0
habis	0	0	0	0	0	0	0	0
ia	0	0	1	0	0	0	0	0
jadi	0	0	0	0	0	0	0	0
kabar	0	0	1	0	0	0	0	0
laba	0	0	0	0	0	0	0	0
maaf	0	0	0	0	0	0	0	0
nada	0	0	0	0	0	0	0	0
obat	0	0	0	0	0	0	0	0
pabrik	0	0	0	0	0	0	0	0
raba	0	0	0	0	0	0	0	0
saat	0	0	0	0	0	0	0	0
taat	0	0	0	0	0	0	0	0
uang	0	0	0	0	0	0	0	0
vakum	0	0	1	0	0	0	0	0
wabah	0	0	0	0	0	0	0	0
ya	0	0	0	0	0	0	0	0
zaman	0	1	0	1	0	0	0	0
Rata-Rata Waktu Pencarian	**0**	**0.041667**	**0.125**	**0.125**	**0**	**0**	**0**	**0**

Ukuran file jar yang dihasilkan untuk masing-masing algoritma dengan jumlah kata yang berbeda juga didata. File jar adalah file distribusi aplikasi berbasis Java yang kemudian dapat diinstall di telepon genggam. Ukuran file jar dapat dilihat di tabel 3.

Tabel 3. Perbandingan Ukuran File jar

Algoritma	Ukuran jar File untuk Setiap Jumlah Kata			
	1000	2000	3000	4000
Biner	63KB	72KB	81KB	90KB
Hashtable	63KB	72KB	81KB	90KB

Pengujian juga dilakukan dengan menggunakan file teks *dummy* hingga 14000 kata untuk menguji kemampuan maksimal penyimpanan data oleh struktur data array pada pencarian biner dan struktur data Hashtable. Detil hasilnya dapat dilihat di tabel 4.

Tabel 4. Kemampuan Penyimpanan Struktur Data

Jumlah Kata	Array pada Biner	Hashtable
1000	berhasil	berhasil
2000	berhasil	berhasil
3000	berhasil	berhasil
4000	berhasil	berhasil
5000	berhasil	berhasil
6000	berhasil	berhasil
7000	gagal	berhasil
8000	gagal	berhasil
9000	gagal	berhasil
10000	gagal	berhasil
11000	gagal	berhasil
12000	gagal	berhasil
13000	gagal	berhasil
14000	gagal	gagal

4.2. Pembahasan Hasil Penelitian

Proses pengujian dengan menggunakan dua lingkungan yang berbeda, dua algoritma yang berbeda dan jumlah kata yang berbeda telah berhasil dilakukan. Hasil pencarian pada emulator telepon genggam di komputer ternyata menghasilkan waktu

pencarian yang sama yaitu 0 milidetik, walaupun algoritma yang digunakan berbeda dan jumlah katanya berbeda. Hal ini wajar terjadi karena pencarian tersebut dilakukan di emulator komputer yang performanya jauh lebih tinggi dari performa telepon genggam sebenarnya. Sedangkan hasil pencarian pada telepon genggam memberikan hasil yang tidak sama antara kedua algoritma yang diuji. Sesuai dengan landasan teori, pengujian dengan menggunakan struktur data Hashtable terbukti menghasilkan waktu yang konstan yaitu 0 milidetik untuk jumlah kata 1000, 2000, 3000 dan 4000. Di lain pihak, algoritma pencarian menghasilkan waktu pencarian yang berbeda untuk setiap jumlah kata. Rata-rata waktu pencarian biner pada 1000 kata adalah 0 milidetik, pada 2000 kata adalah 0.042 milidetik, pada 3000 dan 4000 kata adalah 0.125 milidetik. Ternyata jumlah kata berpengaruh pada lama waktu pencarian kata dengan menggunakan algoritma pencarian biner walaupun tidak terlalu signifikan. Karena jumlah kata yang digunakan juga baru mencapai 4000 kata. Dan ukuran milidetik adalah sangat kecil dan sebenarnya tidak terdeteksi oleh pengguna aplikasi.

Algoritma yang berbeda tidak berpengaruh terhadap besarnya file jar karena jumlah baris kode yang terhasil relatif sama. Besarnya file jar dipengaruhi oleh jumlah kata. Semakin banyak kata yang dimasukkan dalam file teks maka semakin besar pula ukuran file jar. Ukuran file jar pada jumlah kata 1000 adalah 63KB, jumlah kata 2000 adalah 72KB, jumlah kata 3000 adalah 81KB dan jumlah kata 4000 adalah 90KB. Jadi setiap penambahan 1000 kata, ukuran file jar bertambah sebanyak 9KB. Namun ukuran KB adalah sangat kecil jika dibandingkan dengan memori telepon genggam yang bisa mencapai 1GB untuk memori internal dan 16GB untuk memori eksternal.

Penelitian ini memang hanya menggunakan jumlah kata dalam file teks hingga 4000 kata. Namun peneliti ingin mengetahui kemampuan maksimal penyimpanan data dari struktur data array pada pencarian biner dan struktur data Hashtable. Oleh karena itu, peneliti membuat file teks *dummy* dimulai dari 1000 kata dan naik 1000 hingga mencapai 14000 kata. Terdapat perbedaan kemampuan maksimal penyimpanan data pada struktur daya array yang digunakan untuk algoritma pencarian biner dan struktur data Hashtable. Pengujian untuk jumlah data hingga 6000 kata masih berhasil pada struktur data array yang digunakan oleh pencarian biner. Namun mulai dari jumlah kata 7000, aplikasi memunculkan *exception*. *Exception* merupakan pesan *error* (kesalahan) di aplikasi berbasis Java. Sebaliknya struktur data Hashtable berjalan dengan baik pada pengujian dengan jumlah kata 1000 hingga 13000. Namun baru memunculkan *exception* pada jumlah kata 14000. Struktur data Hashtable ternyata mampu menyimpan lebih banyak data berbanding dengan struktur data array pada pencarian biner.

5. KESIMPULAN DAN SARAN

Pengujian algoritma pencarian biner dan pencarian pada struktur data Hashtable berhasil berjalan dengan baik pada emulator di komputer dan telepon genggam. Pengujian melibatkan 24 buah contoh kata yang diambil dari file teks tempat penyimpanan data kamus. Besarnya jumlah kata dalam file teks juga dibedakan menjadi 1000, 2000, 3000 dan 4000. Proses pencarian pada emulator telepon genggam di komputer menghasilkan waktu pencarian 0 milidetik untuk kedua-dua algoritma. Namun pengujian pada telepon genggam, pencarian biner menghasilkan waktu pencarian yang berbeda untuk setiap jumlah kata dalam file teks. Semakin besar jumlah kata dalam file teks maka rata-rata waktu pencarian menjadi lebih lama.

Namun waktu pencarian dalam milidetik jelas tidak terdeteksi oleh pengguna aplikasi. Sedangkan struktur data Hashtable menghasilkan rata-rata waktu pencarian kata yang konstan yaitu 0 milidetik hingga jumlah kata 4000. Jumlah kata yang tersimpan dalam file teks berpengaruh terhadap besarnya file jar dimana semakin banyak kata dalam file teks maka file jar akan menjadi lebih besar. Namun ukuran file jar masih sangat kecil dibandingkan dengan rata-rata ukuran memori telepon genggam, sehingga aplikasi ini tidak memerlukan sumber daya yang banyak. Struktur data Hashtable juga dapat menampung data lebih banyak dari struktur data array pada pencarian biner. Oleh karena itu Hashtable dapat menjadi alternatif struktur data dan algoritma pada aplikasi kamus e-Acesia.

DAFTAR PUSTAKA

Al-Gayoni, Y.U. 2010. Memartabatkan Bahasa Aceh, *Koran Serambi Indonesia*, 3 Juli.

Bakar, A., Sulaiman, B., Hanafiah, M.A., Ibrahim, Z.A. dan Syarifah. 1985. *Kamus Aceh Indonesia*. Departemen Pendidikan dan Kebudayaan, Jakarta.

Basry, M.H. 1994. *Kamus Umum Indonesia-Aceh*. Yayasan Cakra Daru, Jakarta.

Knuth, D.E. 1998. *The Art of Computer Programming*. Addison-Wesley, Massachusetts.

Mutiawani, V., Juwita, Irvanizam. 2011. Penerapan Algoritma Pencarian Biner dalam Aplikasi Kamus e-Acesia. *Prosiding Seminar Nasional Informatika 2011*: 17-19

Razi, K. 2007. Sikap Siswa SMU Negeri Banda Aceh terhadap Bahasa Indonesia dan Bahasa Daerah (Studi Kasus di SMU Negeri 5 Banda Aceh), *Langgam Bahasa*, *Jurnal Ilmiah Pendidikan Bahasa Sastra Indonesia dan Daerah* 1 (1): 45-46.

Weiss, M.A. 2012. *Data Structures and Algorithm Analysis in Java 3rd edition*. Pearson Education, United States of America.