PENGGUNAAN ALGORITMA BACTRACKING DALAM PEMBUATAN ANAGRAM

Firman Rickson Saragih

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika, Insitut Teknologi Bandung
Jalan Ganesha 10 Bandung
e-mail: if16096@students.if.itb.ac.id

ABSTRAK

Anagram merupakan salah satu bentuk dari kriptografi yang menyamarkan suatu pesan dengan mengubah susunan huruf dalam suatu kata atau suatu kalimat. Namun salah satu keunikan dari anagram yang membedakannnya dengan bentuk kriptografi lainnya adalah bahwa setiap kata dalam anagram harus merupakan suatu kata atau kalimat yang memiliki arti. Hal ini sekaligus merupakan tantangan bagi kriptografer yang ingin menggunakannya sebagai metode untuk membuat sandi.
Ada berbagai macam cara dan algoritma yang dapat diterapkan untuk membuat sebuah anagram, namun dalam makalah ini penulis akan mencoba menyajikan penggunaan salah satu algoritma untuk membantu pembuatan anagram yakni algoritma backtracking. Algoritma bactracking merupakan algoritma yang didasarkan pada DFS dan bekerja dengan pohon ruang status. Algoritma ini akan menguji setiap simpul yang dihasilkan dan apabila terbentur pada fungsi pembatas pencarian akan mundur (bactrack) ke simpul sebelumnya.
Selain algoritma bactracking, pembuatan anagram juga membutuhkan algoritma string matching yang tepat, karena kata-kata dalam sebuah anagram harus memiliki makna, dalam hal ini algoritma string matching yang tepat dapat membantu untuk mencari kata yang dibuat dalam database kamus yang tersedia, sehingga angram dapat dibuat dengan lebih cepat dan mangkus.

Kata kunci: Anagram, algoritma, backtracking, kriptografi, string matching.

1. PENDAHULUAN

Akhir-akhir ini minat masyarakat pada kriptografi kuno khususnya anagram tampak semakin meningkat karena buku-buku fiksi mengenai *secret society* dan segala macam rahasianya yang cukup laris di pasaran, seperti *The Da Vinci Code, Angels and Demons* dan sebagainya.

Sebenarnya, seni membuat anagram memiliki sejarah yang panjang dalam bidang kriptografi sepanjang sejarah. Ornag-orang Roma pernah menyebut anagram sebagai *ars magna*—seni besar. Spiritualitas Kabbalah menggunakan banyak menggunakan angram untuk menyusun kembali kata-kata berbahasa Ibrani dalam kitab suci mereka untuk menemukan kata-kata baru dan makna-makna baru. Raja-raja Prancis di zaman renaissance percaya pada kekuatan anagram dan menunjuk ahli-ahli anagram untuk membantu mereka menetapkan keputusan dengan menganalisa dokumen-dokumen penting.

Namun sejak penemuan metode-metode kriptografi yang lebih mudah untuk dibuat dan memiliki tingkat keamanan yang lebih tinggi, anagram semakin ditinggalkan dan lebih sering digunakan untuk permainan sandi antara dua orang atau teka-teki kata saja. Hal ini diakibatkan oleh sulitnya membuat anagram terutama karena setiap kata dalam anagram harus memiliki makna dan bukannya sekedar deretan karakter tanpa makna yang sering digunakan pada masa kini. Selain sulit untuk dibuat, anagram juga cenderung sulit untuk di-*decipher* karena sulitnya membuat kunci yang baku, sehingga diperlukan seorang yang benar-benar akrab bdengan anagram dan memiliki pengetahuan bahasa serta kosa kata yang luas untuk menafsirkannya. Walaupun demikian anagram memiliki nenerapa kelebihan, yakni pesan hasil *cipher* pada umumnya tidak akan dicurigai sebagai sandi karena terlihat sebagai kalimat biasa sehari-hari.

Beberapa contoh anagram dapat dilihat di bawah ini :
Leonardo da Vinci → O, Draconian devil
So dark the con of man → Madonna of the Rocks dsb.
Bahkan kata "planets" dalam bahasa Inggris dikatakan dapat dibentuk menjadi 92 kata lainnya dalam bahasa Inggris.

Dalam makalah ini penulis berusaha menampilkan salah satu algoritma yang dapat dipakai untuk mempermudah pembuatan anagram, yakni algoritma backtracking. Sealin itu untuk membuat anagram, diperlukan adanya database kamus serta algoritma string matching yang tepat agar setiap kata memiliki makna.

Algoritma bactracking adalah suatu algoritma yang merupakan perbaikan dari algoritma brute force dengan

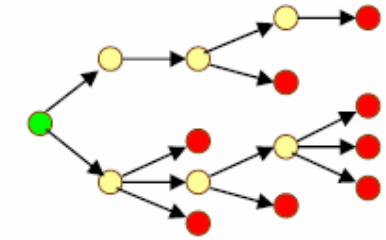
menggunakan algoritma rekursif dan berbasis pada DFS dalam mencari solusi. Selain itu algoritma ini juga merupakan metode yang mencoba beberapa pilihan sampai menemukan satu pilihan yang merupakan solusi.

Pengguna algoritma ini tidak perlu memeriksa seluruh kemungkinan solusi yang ada (seperti sequential search) atau membangkitkan seluruh simpul sampai ke batas akhir, karena hanya simpul yang mengarah ke solusi yang di-expand dan simpul yang tidak mungkin mengarah ke solusi akan di"bunuh" dan tidak perlu di-expand sehingga menghemat proses komputasi dan waktu.

Algoritma ini banyak diterapkan untuk program games dan permasalahan pada bidang kecerdasan buatan (artificial intelligence).

2. METODE PENCARIAN SOLUSI

Seperti telah dijelaskan sebelumnya, algoritma bactracking berbasis pada DFS dan menggunakan pohon ruang status.



Gambar 1. Contoh pohon ruang status

Langkah-langkah umum pencarian solusi dengan algoritma backtracking adalah sebagai berikut :

1. Solusi dicari dengan membentuk lintasan dari akar ke daun. Aturan pencarian yang digunakan adalah mengikuti metode DFS. Simpul yang sudah dilahirkan dinamakan simpul hidup (live node). Simpul yang sedang diperluas dinamakan simpul-E (expand node). Simpul dinomori dari atas ke bawah sesuai urutan kelahirannya.
2. Tiap kali simpul-E diperluas, lintasan yang dibangun olehnya bertambah panjang. Jika lintasan yang dibentuk tidak mengarah ke solusi, simpul tersebut dibunuh sehingga menjadi simpul mati (dead node). Fungsi yang digunakan untuk membunuh simpul-E adalah dengan menerapkan fungsi pembatas (bounding function). Simpul yang sudah mati tidak akan diperluas lagi.
3. Jika pembentukan lintasan berakhir dengan simpul mati, maka proses pencarian dilanjutkan dengan membangkitkan simpul anak yang lainnya. Bila tidaka ada lagi simpul anak yang dapat dibangkitkan, maka pencarian dilanjutkan dengan melakukan backtracking ke simpul hidup terdekat (simpul orangtua). Selanjutnya simpul ini menjadi simpul-E yang baru. Lintasan baru dibangun kembali sampai lintasan tersebut membentuk solusi.
4. Pencarian dihentikan bila solusi telah ditemukan atau tidak ada lagi simpul hidup untuk dirunut balik, namun dalam hal pembuatan anagram ini pencarian solusi lainnya tetap diteruskan karena ada beberapa solusi.

Alasan digunakannya algoritma runut-balik dalam permasalahan pembuatan anagram ini antara lain :

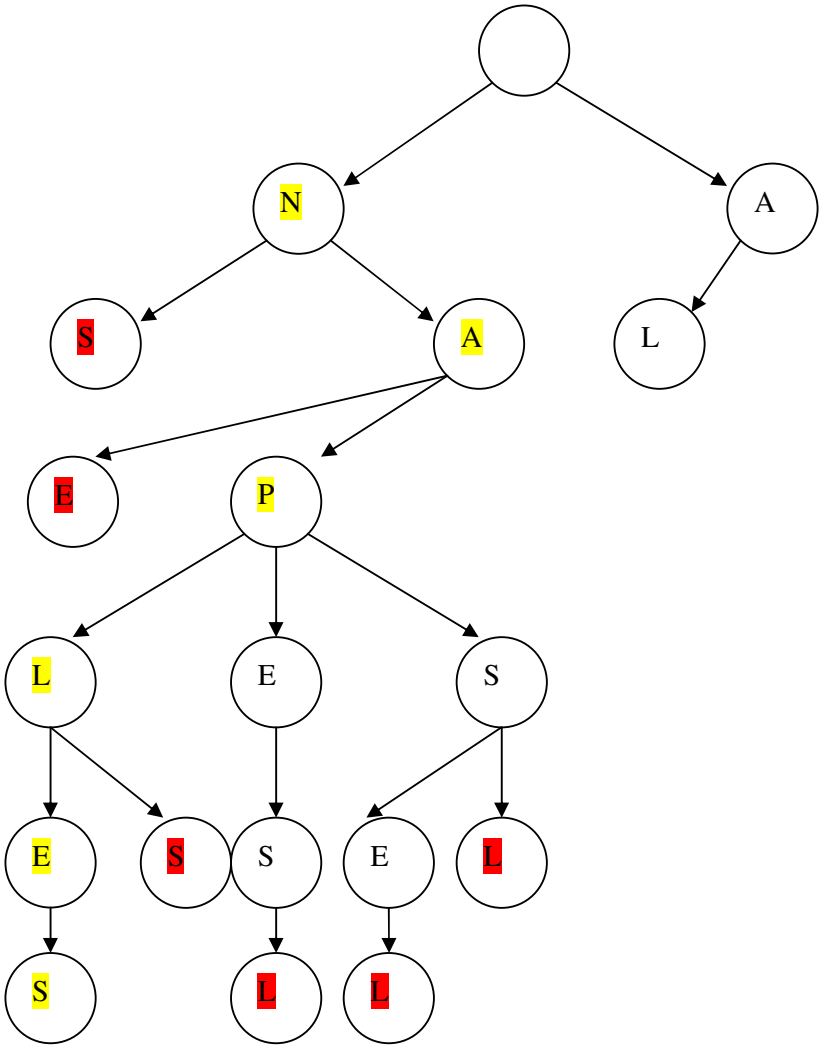
1. Banyak kemungkinan kata yang dapat dibentuk dari huruf-huruf dalam kata yang akan dibuat anagramnya, dan setiap huruf memiliki peluang yang relatif sama untuk menjadi huruf selanjutnya.
2. Setiap pilihan yang diambil mengarah pada sekumpulan pilihan selanjutnya.
3. Adanya fungsi pembatas yang terdapat dalam berbagai bahasa.
4. Terdapat banyak solusi untuk persoalan ini.

Karena banyaknya solusi, pada umumnya seorang pembuatan anagram akan menerapkan cara yang cenderung brute force dan berdasarkan ingatannya sendiri untuk membuat anagram, namu hal tersebut akan memakan banyak waktu dan kurang mangkus.

3. PENCARIAN SOLUSI DENGAN ALGORITMA BACKTRACKING

Hal pertama yang perlu dilakukan adalah membuat file kamus yang berisi daftar kata yang boleh dipergunakan. File ini harus dapat menyatakan apakah masukan kata yang diterima valid atau tidak. Setelah itu program akan secara iteratif mengacak huruf-huruf dalam kata dengan mengingat fungsi pembatas, misalnya dalam bahasa Inggris tidak terdapat kata yang diawali dengan pasangan huruf 'xz', dalam bahasa Indonesia tidak terdapat kata yang diawali dengan 3 konsonan sekaligus dsb. Kemudian kata-kata tersebut akan diinput ke dalam file kamus untuk dinyatakan apakah valid atau tidak.

Salah satu contoh skema pencarian anagram untuk suatu kata dapat ditunjukkan dalam contoh pohon ruang status di bawah ini. Pohon ruang status ini menunjukkan pencarian angram dalam bahasa Inggris untuk kata "planes".



Gambar 2. Pohon ruang status untuk pencarian solusi

Pada gambar di atas dapat dilihat sebuah contoh penggunaan backtracking dalam pembuatan anagram. Simpul yang berwarna kuning menyatakan solusi sedangkan simpul yang berwarna merah menyatakan simpul yang dibunuh selama proses pencarian. Dari proses di atas terlihat bahwa dihasilkan kata "naples" dari kata "planes" dan masih banyak kata lain yang dapat dihasilkan dari proses ini jika proses dilanjutkan.

Secara sederhana algoritma backtracking untuk proses ini dapat dituliskan sebagai berikut

1. Pilih sebuah huruf dalam kata asal (kata yang akan dianagramkan). Masukkan ke dalam tabel kata hasil anagram.
2. Pilih huruf yang selanjutnya, lalu cek ke file kamus atau ke fungsi pembatas, jika tidak ada kata yang berawalan demikian, hapus tabel dan mulai kembali dari langkah pertama.
3. Jika terdapat dalam kamus, maka ulangi langkah kedua dan seterusnya hingga diperoleh solusi-solusi dan kemungkinan yang ada sudah habis (tidak ada simpul yang dapat di-expand lagi).

4. KESIMPULAN

Penggunaan algoritma backtracking dalam mencari solusi untuk persoalan anagram ini mempercepat dan mempermudah pencarian solusi jika dibandingkan dengan pencarian solusi menggunakan memori pembuat atau dengan brute force.

Dalam pencarian solusi, digunakan pohon ruang status berbasis DFS yang terus menelusuri simpul demi simpul sampai menemukan solusi sampai tidak ada lagi simpul hidup yang dapat ditelusuri. Selain itu dibutuhkan adanya fungsi pembatas yang sesuai dengan bahasa asli anagram tersebut untuk membantu pencarian, dalam hal ini fungsi dari fungsi pembatas itu sendiri mirip dengan fungsi dari teknik heuristik dalam sequential search, yakni untuk mempercepat dan mengefisienkan proses pencarian solusi.

Penggunaan algoritma string matching yang tepat juga sangat dibutuhkan dalam hal ini karena algoritma yang tepat dapat membantu memangkuskan pencarian kata dalam kamus, sebab algoritma yang dibuat penulis perlu mengecek ke file kamus berkali-kali sehingga jika algoritma string matching yang digunakan adalah brute force akan sangat memakan waktu dan memiliki kompleksitas yang tinggi.

REFERENSI

[1] Anany Levitin, “Introduction to The Design and Analysis of Algorithms”, Addison-Wesley, 2003.

[2] Rinaldi Munir, “Diktat Kuliah IF2251 Strategi Algoritmik”, Prodi Teknik Informatika, Sekolah Teknik Elektro dan Informatika ITB, 2006.

[3] Simon Singh,”The Code Book : The Science of Secrecy from Ancient Egypt to Quantum Cryptography”, Anchor Books, 1999.