PENGKAJIAN MASALAH *LONGEST COMMON SUBSEQUENCES*

Dominikus Damas Putranto, NIM 13506060

Program Studi Teknik Informatika ITB
Jalan Ganesha 10, Bandung 40132
e-mail: if16060@students.if.itb.ac.id

ABSTRAK

Mencari uparangkaian bersama terpanjang (*longest common subsequence / LCS*) dari beberapa buah rangkaian adalah sebuah masalah klasik dalam ilmu komputer (*computer science*).

Masalah ini adalah masalah yang penyelesaiannya sederhana, yaitu berupa sebuah fungsi rekursif. Yang kemungkinan besar tidak efisien karena kita melakukan penghitungan yang sama secara berulang-ulang. Namun dengan program dinamis hal tersebut dapat dihindari, karena setiap penghitungan untuk uparangkaian dilakukan, penyimpanan terhadap hasil tersebut dilakukan. Sehingga nanti penghitungan untuk rangkaian yang lebih besar dapat dilakukan dengan hanya melihat hasil yang telah disimpan tersebut, daripada harus melakukan penghitungan lagi untuk uparangkaian yang sama.

Dalam makalah ini akan dikaji masalah LCS ini, dimulai pada definisi pada Bab 2, kemudian algoritma pemecahannnya dan pembuktiannya pada Bab 3, dan pada Bab 4, penerapan algoritma pemecahan dengan menggunakan teknik program dinamis akan dijelaskan, dan implementasinya menggunakan bahasa Pascal akan dibeberkan di Bab 5.

Kata kunci: LCS, *longest common subsequences*, subrangkaian terpanjang, *dynamic programming*, program dinamis.

1. PENDAHULUAN

Mencari uparangkaian bersama terpanjang (longest *common subsequence*) dari beberapa buah rangkaian adalah sebuah masalah klasik dalam ilmu komputer (*computer science*).

Aplikasinya dipakai secara luas, tidak hanya terbatas di lingkup *computer science*, namun juga di bidang biologi (misalnya dalam algoritma pemrosesan dan pengujian kecocokan DNA), dan di bidang-bidang lain.

Walaupun begitu, masalah ini adalah masalah yang penyelesaiannya sederhana, yaitu berupa sebuah fungsi rekursif. Yang kemungkinan besar tidak efisien karena kita melakukan penghitungan yang sama secara berulang-ulang. Namun dengan program dinamis hal tersebut dapat dihindari, karena setiap penghitungan untuk uparangkaian dilakukan, penyimpanan terhadap hasil tersebut dilakukan. Sehingga nanti penghitungan untuk rangkaian yang lebih besar dapat dilakukan dengan hanya melihat hasil yang telah disimpan tersebut, daripada harus melakukan penghitungan lagi untuk uparangkaian yang sama.

2. DEFINISI *LONGEST COMMON SUBSEQUENCES*

Longest Common Subsequences adalah masalah mencari uparangkaian bersama (*common subsequence*) terpanjang dari beberapa (biasanya hanya 2) buah rangkaian.

Uparangkaian (*subsequence*) dari sebuah string S adalah sekumpulan karakter yang ada pada S yang urutan kemunculannya sama. Atau dalam definisi formalnya

> Rangkaian Z adalah uparangkaian dari X $<x_1,x_2,...,x_m>$, jika terdapat urutan menaik $<i_1,i_2,...,i_k>$ yang merupakan indeks X untuk semua j=1,2,..,k, yang memenuhi $x_i=z_j$.

Misal pada S = GAATACA, beberapa uparangkaian yang mungkin adalah : GAT, TCA, dan GC.

Uparangkaian bersama (*common subsequence*) dari 2 rangkaian adalah uparangkaian yang terdapat pada kedua rangkaian tersebut. Misal S1 = GATTTAC dan S2 = AAGATC, maka GAT, GATC, maupun GAC adalah uparangkaian bersama dari S1 dan S2.

Uparangkaian bersama terpanjang (*longest common subsequences*) adalah uparangkaian bersama dari 2 rangkaian yang paling panjang.

3. ALGORITMA PENYELESAIAN LCS

3.1. Algoritma

Algoritma untuk masalah LCS ini berdasarkan teorema di bawah ini :

Misal $X = < x_1, x_2, .. , x_m >$ dan $Y = < y_1, y_2, .. , y_n >$ adalah rangkaian dan $Z = < z_1, z_2, .. , z_k >$ adalah suatu LCS dari X dan Y.

1. Jika $x_m = y_n$, maka $z_k = x_m = y_n$ dan Z_{k-1} adalah suatu LCS dari X_{m-1} dan Y_{n-1}.

2. Jika $x_m \neq y_n$ maka $z_k \neq x_m$ mengimplikasikan bahwa Z adalah suatu LCS dari X_{m-1} dan Y

3. Jika $x_m \neq y_n$ maka $z_k \neq y_n$ mengimplikasikan bahwa Z adalah suatu LCS dari X dan Y_{n-1}

3.2. Pembuktian Algoritma

Adapun bukti dari teorema di atas akan dijabarkan di sini :

1.
Jika $z_k \neq x_m$ maka kita bisa meng-*append* $x_m = y_n$ pada Z untuk mendapatkan *common subsequence* dari X dan Y untuk panjang k + 1, yang kontradiksi dengan dugaan bahwa Z adalah LCS dari X dan Y. Untuk menunjukkan bahwa itu adalah LCS :

Misal, kita mempunyai $z_k = x_m = y_n$. Sekarang prefiks dari Z_{k-1} adalah *common subsequence* dari X_{m-1} dan Y_{n-1} yang panjangnya adalah k-1.
Lalu ada sebuah *common subsequence* W dari X_{m-1} dan Y_{n-1} dengan panjang lebih besar daripada k-1. Melakukan *append* $x_m = y_n$ pada W menghasilkan *common subsequence* dari X dan Y dengan panjang lebih dari k. Yang adalah kontradiksi dengan pernyataan di atas.

2.
Jika $z_k \neq x_m$ maka Z adalah *common subsequence* dari X_{m-1} dan Y. Jika terdapat sebuah *common subsequence*W dari X_{m-1} dan Y yang panjangnya lebih dari k, maka hal ini berkontradiksi dengan asumsi bahwa Z adalah LCS dari X dan Y.

3.
Pembuktian simetris dengan kasus 2.

4. PENYELESAIAN MENGGUNAKAN PROGRAM DINAMIS

Penyelesaian masalah LCS sebenarnya bisa dilakukan dengan banyak sekali cara selain dengan menggunakan program dinamis, antara lain, *BackTracking*, *DFS*, *BruteForce*, dan *Greedy*. Namun penyelesaian dengan program dinamis dipilih karena kemangkusan dan keoptimalannya walaupun memerlukan ruang memori tambahan karena diperlukannya tabel untuk menyimpan hasil sementara.

Ide dalam menggunakan Program Dinamis ini adalah kita membuat tabel matriks yang menyimpan hasil penghitungan LCS untuk uparangkaian-uparangkaian dari S1 dan S2 dengan kalkulasi secara rekursif seperti pada Bab 3. Tabel yang akan terbentuk adalah tabel 2 dimensi dengan panjang kolom sama dengan panjang S1 dan panjang baris sama dengan panjang S2.

4.1. Mendapatkan Panjang LCS

Misal *m* adalah panjang S1, *n* adalah panjang S2, dan tab adalah matriks berukuran *m* x *n*, dan tab[*i*, *j*] adalah panjang LCS dari uparangkaian pertama yang terdiri dari *i* karakter pertama S1 dengan uparangkaian kedua yang terdiri dari *j* karakter pertama S2, maka untuk $1 \leq i \leq m$ dan $1 \leq j \leq n$, sesuai dengan fungsi rekursif di atas, maka :

$$\text{tab}[i, j] = \begin{cases} \text{tab}[i-1, j-1] + 1, S1[i] = s2[j] \\ \max(\text{tab}[i-1, j], \text{tab}[i, j-1]) \end{cases}$$

Misal, untuk S1 = AGCGA dan S2 = CAGATAGAG, tabel yang terbentuk adalah tabel berukuran 5 x 9 yang isinya sebagai berikut :

Tabel hasil kalkulasi LCS dari AGCGA dan CAGATAGAG

		C	A	G	A	T	A	G	A	G
		1	2	3	4	5	6	7	8	9
A	1	0	1	1	1	1	1	1	1	1
G	2	0	1	2	2	2	2	2	2	2
C	3	1	1	2	2	2	2	2	2	2
G	4	1	1	2	2	2	2	3	3	3
A	5	1	2	2	3	3	3	3	4	4

Dan sesuai dengan arti notasi dari setiap tab[*i*, *j*] bahwa LCS dari S1 dan S2 adalah tab[*m*, *n*] atau dalam kasus ini adalah 4.

4.2. Mendapatkan LCS

Sedangkan untuk mendapatkan LCS yang dimaksud, kita bisa merunut balik tabel dimulai dari tab[*m*, *n*]. Dalam runutbalik ini, yang kita lakukan sebenarnya hanyalah merunutbalik pilihan yang dilakukan pada saat kalkulasi tab[*i*, *j*].

Untuk setiap $1 \le i \le m$ dan $1 \le j \le n$, jika S1[*i*] sama dengan S2[*j*], maka maka karakter itu pasti terdapat pada LCS. Selain itu, cek, darimana dia mendapatkan LCS saat itu, lakukan pemilihan yang sama. Lakukan hal tersebut dimulai dari $i = m$ dan $j = n$.

Setelah hal itu dilakukan maka akan terbentuk tabel di bawah ini. Di mana *cell* yang dihitamkan menandakan karakter yang masuk ke dalam LCS.

Tabel hasil perunutan balik untuk mendapatkan LCS dari AGCGA dan CAGATAGAG

		C	A	G	A	T	A	G	A	G
		1	2	3	4	5	6	7	8	9
A	1	0	1	1	1	1	1	1	1	1
G	2	0	1	2	2	2	2	2	2	2
C	3	1	1	2	2	2	2	2	2	2
G	4	1	1	2	2	2	2	3	3	3
A	5	1	2	2	3	3	3	3	4	4

Berdasarkan tabel tersebut di atas terlihat bahwa LCS-nya adalah AGGA.

5. IMPLEMENTASI

Implementasi dilakukan menggunakan bahasa Pascal.

Source di bawah ini hanyalah implementasi langsung dari fungsi-fungsi pada Bab 3. Namun pada bagian fungsi untuk mendapatkan LCS nya, hanya menghasilkan salah satu LCS yang mungkin. Source code-nya dalam bahasa pascal adalah sebagai berikut :

```
(***********
Implementasi    LCS    menggunakan    Program
Dinamis
************)

program LCS;

type
  tabel      =  array[0..100,  0..100]  of
word;
  teks = string;

var
  S1, S2     : teks;
  tab        : tabel;

function max (a : word; b : word) : word;
begin
  if (b > a) then max := b else max := a;
end;

procedure BentukTabelLCS (S1 : teks; S2 :
teks; var T : tabel);
var
  i, j : word;
  m, n : word;
begin
  fillchar (T, sizeof(T), 0);

  m := length(S1); n := length(S2);
  for i:=1 to m do
    for j:=1 to n do
      if (S1[i] = S2[j]) then
        T[i,j] := T[i-1,j-1] + 1
      else
        T[i,j] := max (T[i,j-1], T[i-1,j]);
end;


function RunutBalikLCS (tab : tabel; S1 :
teks; S2 : teks; i : word; j : word) :
teks;

begin
  if (i = 0) or (j = 0) then
    RunutBalikLCS := ''
  else if  S1[i] = S2[j] then
    RunutBalikLCS := RunutBalikLCS(tab, S1,
S2, i-1, j-1) + S1[i]
  else begin
    if tab[i,j-1] > tab[i-1,j] then
      RunutBalikLCS  :=  RunutBalikLCS  (tab,
S1, S2, i, j-1)
    else
      RunutBalikLCS  :=  RunutBalikLCS  (tab,
S1, S2, i-1, j);
  end;
end;


begin
  readln (S1);
  readln (S2);

  BentukTabelLCS (S1, S2, tab);

  writeln (tab[length(S1), length(S2)]);

  writeln  (RunutBalikLCS  (tab,  S1,  S2,
length(S1), length(S2)));
end.
```

6. KESIMPULAN

Masalah uparangkaian bersama terpanjang (*longest common subsequences*) dengan karakteristiknya sangat cocok bila diselesaikan dengan menggunakan program dinamis karena bisa mangkus dan optimal.

REFERENSI

[1] Wikipedia, (2008). Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Longest_common_subsequence_problem.htm Tanggal akses : 17 Mei 2008 pukul 08.00 GMT +7

[2] www.columbia.edu/~cs2035/courses/csor4231.F07/lcs.pdf Tanggal akses : 17 Mei 2008 pukul 08.00 GMT +7

[3] http://www-igm.univ-mlv.fr/%7Elecroq/seqcomp/node4.html Tanggal akses : 17 Mei 2008 pukul 08.00 GMT +7