# Induksi untuk Greedy pada Huffman Code

Gabrielle Wicesawati Poerwawinata-13510060[1]
*Program Sarjana Informatika*
*Sekolah Teknik Elektro dan Informatika*
*Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia*
*[1]13510060@std.stei.itb.ac.id*

***Abstract*—Pembentukan pohon Huffman merupakan salah satu contoh dari pemakaian algoritma Greedy. Hal ini dikarenakan untuk membentuk pohon Huffman harus mengambil simpul dengan frekuensi yang paling kecil. Pohon Huffman yang terbentuk diharapkan dapat meminimalkan panjang kode binary yang dibentuk untuk sebuah huruf. Dalam pembuatan sebuah algoritma diperlukan pembuktian untuk algoritma tersebut untuk menguji keoptimalan dari hasil yang dikeluarkan oleh algoritma tersebut.**

**Oleh karena itu diperlukan langkah-langkah induksi untuk dapat menguji kebenaran dari set hasil algoritma yang digunakan. Pada paper ini akan dibahas pembuktian hasil dari algoritma Greedy. Terdapat beberapa metode yang dapat digunakan untuk pembuktian Greedy, salah satunya yaitu metode *Greedy Stay Ahead.***

***Keywords*—Greedy, solusi optimal, *Greedy Stay Ahead*, Pohon Huffman**

## I. INTRODUCTION

Algoritma Greedy adalah salah satu algoritma yang digunakan untuk menyelesaikan masalah optimisasi . Tujuan dari nilai yang ingin didapatkan adalah yang paling minimal atau maksimal. Untuk mewujudkan hal tersebut setiap pemilihan elemen dilakukan untuk memenuhi tujuan akhir dari permasalahan. Greedy dapat diaplikasikan pada berbagai contoh masalah. Sebagai contoh pada masalah penyusunan jadwal, *Huffman code*, Knapsnack Problem, dan pembentukan *minimum Spannint Tree*. Greedy merupakan pendeketan yang paling sederhana untuk masalah optimisasi.

Induksi matematika dapat digunakan untuk membuktikan bahwa algoritma Greedy selalu menghasilkan solusi yang optimal. Pembuktian dilakukan dari sebagian hasil yang dihasilkan oleh algoritma Greedy. Jika sebagian hasil tersebut menghasilkan hasil yang optimal maka hal ini juga berlaku untuk hasil keseluruhan.

Salah satu aplikasi algoritma Greedy adalah Huffman code. Dalam pemilihan daun dari Huffman code ini adalah dengan memilih huruf yang mempunyai frekuensi yang paling kecil. Untuk memilih frekuensi huruf yang paling kecil ini digunakanlah algoritma Greedy. Pohon Huffman yang terbentuk akan dibuktikan dengan menggunkan induksi matematika dengan membuat pohon turunan yang lainnya.

## II. ALGORITMA GREEDY

Masalah optimisasi dimuali dengan adanya sederetan atau sebuah set elemen$\{x_1,\ldots,x_k\}$ yang memenuhi aturan dan tujuan dari sebuah fungsi. Algoritma dari metode Greedy adalah :

```
for i ← 1 to k do
        pilih elemen yang terlihat paling baik saat itu
```

Algoritma Greedy tidak selalu menghasilkan solusi yang optimal, tetapi untuk kebanyakan kasus Greedy menghasilkan yang optimal. Ketika mendesain sebuah algoritma greedy,harus dilakukan hal-hal sebagai berikut:

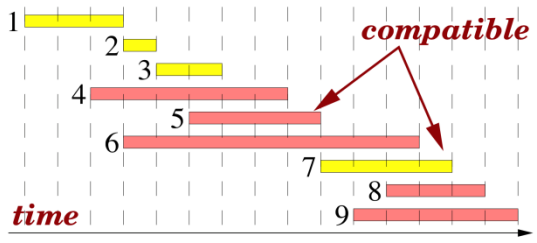
1. Membuktikan algoritma selalu menghasilkan solusi yang optimal.
2. Membuktikan algoritma yang dibangun mendekati hasil yang optimal.
3. Mensimulasikan bahwa algoritma yang dibangun menghasilkan solusi yang baik.

Berikut akan diberikan masalah pemilihan aktifitas. Tujuan utama dari permasalahan ini adalah memilih aktivitas-aktivitas yang akan menghasilkan keuntungan yang paling maksimum. Anggap kita mempunyai sebuah set $S = \{1,2,..,n\}$ dengan $n$ adalah sejumlah aktivitas yang akan dijadikan sumber. Setiap aktivitas mempunyai waktu mulai $s_i$ dan waktu berakhir $a_i$, dengan $s_i \leq a_i$. Jika aktivitas $i$ terjadi dalam rentang waktu $[s_i,a_i]$. Dua buah aktivitas $i$ dan $j$ tidak saling bersamaan waktunya contohnya jika $s_i \geq a_j$ atau $s_j \geq a_i$. Diasumsikan masukan dari aktivitias dirutukan berdasarkan waktu selesesainya :

$$a_1 \leq a_2 \leq \ldots \leq a_n \quad (1)$$

Pseudocode dari algoritma Greedy adalah:

```
n ← panjang[s]
A ← {1}
j ← 1
for i ← 2 to n
        do if s_i ≥ a_j
                then A ← A ∪ {i}
                     j ← i
return A
```




*Gambar 1 Susunan Aktivitas*

Greedy adalah strategi yang berjalan dengan baik pada masalah optimisasi dengan karakteristik sebagai berikut:

1. Pemilihan properti dengan Greedy
   Global optimum dapat dicapai dengan memilih *local optimum*.
2. Struktur yang optimal
   Sebuah solusi yang optimal dari sebuah permasalahan mengandung solusi yang optimal juga untuk submasalah.

## III. INDUKSI MATEMATIKA

Induksi matematika pada desain sebuah algoritma dilakukan untuk menunjukkan kebenaran pada kasus dasar algoritma tersebut dan pembuktian tersebut diikuti sampai size ke $n$ yang nilainya lebih besar daripada nilai $n$. Induksi digunakan untuk membuktikkan kebenaran pada iterativ dan algoritma recursive.

Prinsip dari induksi matematika adalah dengan membuktikan $P(n)$ bernilai benar untuk semua bilangan positif integer n, dimana $P(n)$ adalah fungsi proposional, akan dilakukan dua langkah:

**Langkah basis**: Nilai $P(1)$ diasumsikan bernilai benar

**Langkah induksi**: Jika $P(n)$ benar maka $P(n+1)$ juga benar untuk semua bilangan positif k.

Asumsi nilai $P(k)$ adalah benar dinamakan hipotesis induksi. Jika kedua langkah pembuktian telah dilakukan, kita telah membuktikan bahwa nilai $P(n)$ semua benar untuk semua bilangan positif integer.

Induksi matematika adalah teknik pembuktian yang valien. Hal ini dikarenakan induksi matematika akan mencari nilai kebenaran dari setiap elemen dalam set hasil.

## IV. PEMBUKTIAN GREEDY DENGAN INDUKSI MATEMATIKA

Terdapat 5 teknik yang dapat digunakan untuk membuktikkan kebenaran pada sebuah algoritma Greedy. Teknik-teknik tersebut adalah *Staying Ahead*, *Exchange Arguments*, *Matroids*, *Greedoids*, dan *Matroid embeddings*. Teknik yang akan dibahas pada paper ini adalah *Greedy Stay Ahead*. Ide dasar dari pembuktian ini adalah dengan membuktikan secara induksi bahwa dengan fungsi tertentu,bahwa sebagian solusi yang digunakan oleh Greedy adalah yang terbaik dari sebagian solusi yang dihasilkan oleh algoritma yang lainnya.

Jika pengukuran dari algoritma greedy setiap langkahnya menghasilkan hasil yang optimal, maka pada langkah-langkah yang selanjutnya akan menhasilkan hasil yang optimal juga.

Misalnya pada contoh penjadwalan. Dimiliki n buah interval dimana $s(i)$ adalah waktu awal dan $f(i)$ adalah waktu akhir. Lalu dilakukan langkah-langkah greedy seperti pada yang telah dibahas pada bab II. Lalu dilakukan pembuktian untuk membuktikan $A$ merupakan kumpulan solusi yang optimal atau bukan. Diasumsikan terdapat $O$ yang merupakan kumpulan solusi yang optimal. Sehingga akan dibuktikan $|A| = |O|$ untuk membuktikan algoritma yang dibangun menghasilkan hasil yang terbaik dengan membandingkan solusi awal dari $O$. Jika terdapat pada $A$ terdapat k buah elemen solusi pada $i_1,...,i_k$, dan $O$ terdapat m buah solusi pada $j_1,...,j_m$ maka akan dibuktikan nilai $k=m$. Dengan catatan elemen $A$ dan elemen $O$ harus terurut sesuai kedatangan.

Kita akan membuktikan untuk semua indeks $r \leq k$ didapatkan $f(i_r) \leq f(j_r)$. Pembuktian ini akan dilakukan dengan induksi. Kasus induksi yang paling dasar adalah untuk $r$ =1 adalah benar. Algoritma Greedy akan memilih interval $i_r$ dengan waktu akhir yang paling minimum. Jika $r > 1$ dan asumsikan nilainya juga benar sampai indeks $r$-1. Dengan kata lain $f(i_r\text{-}1) \leq f(j_r\text{-}1)$. Dengan demikian tidak memungkinkan nilai $f(i_r) > f(j_r)$ karena nilai elemen pada A tidak mungkin melebihi nilai $O$.

Jika $A$ tidak optimal maka set optimal pada $O$ harus mempunyai rentang interval dengan $m>k$. Untuk $r=k$ kita ketahui bahwa $f(i_k) \leq f(j_k)$. Sejak $m > k$ maka terdapat interval $j_k+1$ pada $O$. Nilai $j_k+1$ diletakkan setelah $j_k$ dan juga setelah nilai terakhir pada $i_k$. Sehingga setelah menghapus semua interval yang tidak sesuai dengan $i_1,..,i_k$, $R$ masih mengandung $j_k+1$. Tetapi pembuktian algoritma Greedy hanya akan berhenti sampai $i_k$.

Terdapat 4 langkah yang dapat dilakukan untuk melakukan teknik greedy stay ahead, yaitu:

Langkah 1 : Berikan label pada sebagian dan hasil keseluruhan dari algoritma yang dipakai. Sebagai contoh, terdapat A = $\{a_1,a_2,..,a_k\}$ adalah solusi yang dihasilkan oleh algoritma yang digunakan. Lalu terdapat $O$ =$\{o_1,o_2,..,o_m\}$ yang dianggap sebagai kumpulan hasil yang optimal.

Langkah 2: Berikan sebuah pengukuran. Pengukuran yang diberikan dapat berupa sebuah fungsi f(.) dimana pada fungsi ini hasil greedy menjadi yang terbaik pada solusi yang optimal. Sebagai contoh, f($a_1,..,a_j$) akan menjadi waktu yang dibutuhkan untuk menyelesaikan sejumlah $j$ pekerjaan. Sehingga f($o_1,..,o_n$) juga dapat didefinisikan.

Langkah 3: Buktikkan bahwa hasil dari greedy adalah yang terbaik. Tunjukkan bahwa sebagian solusi yang dihasilkan oleh Greedy selalu baik seperti pada sebagian awal dari solusi berdasarkan pengukuran yang telah dipilih. Untuk semua indeks $r \leq k$, buktikkan bahwa f($a_1,..,a_r$)[$\geq$ , $\leq$]f($o_1,..,o_r$).

Langkah 4: Buktikan keoptimalan dari solusi yang didapatkan. Buktikan greedy stay ahead pada solusi-solusi tertentu dengan pengukuran tertentu akan menghasilkan output final yang optimal juga.

## V. GREEDY PADA HUFFMAN CODE

Huffman code digunakan untuk membuat code menjadi lebih singkat. Terdapat sebuah alfabet $C = \{a_1,...,a_n\}$ beserta dengan frekuensinya yaitu $f_i,...,f_n$ dan set dari *prefix-free binary code* $W=\{w_1,...,w_n\}$ yang merupakan panjang kode minimal.

$$B(T) = \sum_{i=1}^{n} fi.|wi|$$

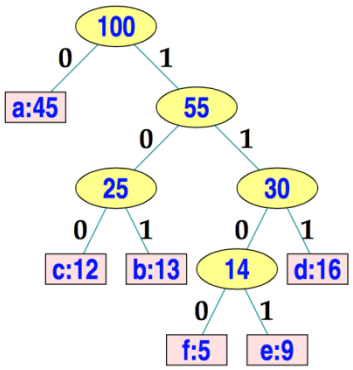
Pseudocode algoritma Huffman adalah sebagai berikut:

```
n ← |C|
Q ← C
for i ← 1 to n-1
        do z ← ALOKASI_NODE()
        x ← left[z] ← EKSTRAK_MIN(Q)
        y ← right[z] ← EKSTRAK_MIN(Q)
        f[z] = f[x] + f[y]
        INSERT (Q,z)
return EKSTRAK_MIN(Q)
```

Dalam pembangunan pohon nilai 1 diletakkan pada cabang sebelah kanan dan nilai 0 untuk cabang sebelah kiri. Code sebuah alfabet dihitung dari akar pohon sampai daun pohon tersebut. Lalu setiap node v diberi label dengan menggunakan jumlah frekuensi dari sebuah simbil yang terdapat pada *subtree*(v). Terdapat contoh sebuah pohon kode Huffman dengan frekuensi pada masing-masing alfabet sebagai berikut:

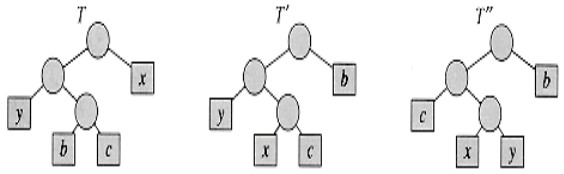
*Tabel 1 Frekuensi pada Huruf*

| Huruf | A | b | C | d | e | f |
|---|---|---|---|---|---|---|
| Frek. | 45 | 13 | 12 | 16 | 9 | 5 |




*Gambar 2 Pohon Huffman*

Kompleksitas waktu yang dibutuhkan pada algoritma Huffman adalah $O(n \log n)$.




*Gambar 3 Pohon Ilustrasi untuk Pembuktian Lemma*

Gambar 3 adalah pembuktian dari Lemma A dan Lemma B. Untuk melakukan pembuktian maka algoritma Huffman dapat dituliskan kembali menjadi:

```
Huffman(C)
if |C| =1 then return single-node tree
x dan y adalah dua karakter pada C dengan
frekuensi yang paling minimal
T' ← Huffman(C)
T ← T' dengan dua anak x,y ditambahkan pada z
return (T)
```

**Lemma A**

JIka *T'* optimal untuk *C'*, maka *T* juga optimal untuk *C*.

**Pembuktian**

Jika *T* adalah pohon yang optimal dari pembentukan kode Huffman dan *h* adalah tinggi dari *T* maka akan terdapat dua buah daun pada ketinggan *h* yang merupakan saudara. Jika daun tersebut bukan *x* dan *y*, maka tukar daun tersebut dengan *x* dan *y*. Hal ini akan meminimalkan rata-rata dari panjang kode.

**Lemma B**

Dari kedua daun *x* dan *y* akan dibentuk sebuah alfabet *D* yang akan menggantikan nilai *x* dan *y* dalam sebuah huruf *z*, seperti $f[z]=f[x]+f[y]$. Sehingga terdapat korespondensi antara pohon *D* dengan *z* sebagai daunnya dan pohon *C* dengan *x* dan *y* sebagai daunnya.

**Pembuktian**

Biaya yang dibutuhkan untuk masing-masing pohon dibuat berbeda. Biaya untuk pohon *C* adalah $B(T)$ sedangkan biaya untuk pohon *D* adalah $B(T')$. Hal ini dapat dibentuk sebuah persamaan.

$B(T) = B(T')+f[x]+f[y]$

$B(T') = B(T)-f[x]-f[y]$

Terdapat biaya pohon lainnya yang dinamakan $B(T'')$, dengan $B(T'')< B(T)$.

Bentuk pohon $T'''$ dengan menggabungkan x dan y pada $T''$. Sehingga terbentuk persamaan

$B(T''') = B(T'') - f[x] - f[y]$

$< B(T) - f[x] - f[y]$

$= B(T')$

## IV. KESIMPULAN

Algoritma Greedy hasil solusi yang dihasilkan kebanyak tepat pada tujuannya. Solusi yang diinginkan adalah yang paling minimal atau yang paling maksimal bergantung pada nilai akhir yang ingin dicapai dari setiap elemen.

Untuk membuktikan bahwa suatu algorima Greedy benar untuk menghasilkan nilai yang optimal, maka dilakukan pembuktian. Pembuktian dilakukan dengan induksi matematika.

Algoritma Greedy dan induksi matematika sama-sama melakukan pengecekan nilai terhadap masing-masing elemen set nya. Algoritma Greedy terdapat sebuah set elemen yang harus mempunyai nilai yang optimal. Lalu pada induksi matematika akan dibuktikan untuk basis dan induksi apakah akan selalu bernilai benar sampai dengan nilai n.

Jika sebagian dari kumpulan solusi itu benar maka akan didapatkan solusi yang benar juga untuk hasil finalnya.

## REFERENCES


[1] cs.rochester.edu/~gildea/csc282/slides/C16-greedy.pdf
[2] crypto.stanford.edu/~zhandry/2012-Summer-CS161/docs/lectures/06-Greedy-1.pdf
[3] Hetland. Magnus Lie, "Proof Methods and greedy algorithm", 2008
[4] Rosen. Kenneth H, *Discrete Mathematics and Its Applications*, 7th ed. New York: McGraw-Hill
[5] Roughgarden. Tim, Alexa Sharp, Tom Wexler, "Guide to Greedy Algorithms", 2013
[6] staff.ustc.edu.cn/~csli/graduate/algorithms/book6/chap17.htm
[7] web.cse.ohio-state.edu/~lai/6331/4.greedy.pdf
[8] www.cs.cornell.edu/courses/cs482/2004su/handouts/greedy_ahead.pdf
[9] www.cs.mun.ca/~kol/courses/3719-w12/greedy.pdf
[10] www.cs.umd.edu/class/sum2005/cmsc451/scheduling.pdf


## PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 11 November 2014

ttd

Gabrielle Wicesawati Poerwawinata / 13510060