Replika *Content-Aware Scaling* Photoshop® Menggunakan *Dynamic Programming*

Ichlasul Amal 13511075
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13511075@std.stei.itb.ac.id

***Abstrak*—Photoshop merupakan perangkat lunak standar industri untuk melakukan manipulasi gambar. Salah satu fitur dari Photoshop adalah Content-Aware Scaling yang mampu meresize gambar tanpa menyebabkan distorsi dan tetap mampu mempertahankan konten yang penting. Teknik yang digunakan Photoshop adalah Seam Carving yang menggunakan Dynamic Programming. Dalam makalah ini akan dilakukan pembahasan tentang Seam Carving dan Dynamic Programming secara umum. Tujuan akhirnya adalah untuk mereplika fitur Photoshop tersebut dan membuat library open source kecil. Dari implementasi menggunakan MATLAB sudah berhasil diperoleh hasil yang cukup baik walaupun masih belum berwujud library.**

***Kata Kunci*—manipulasi gambar, content-aware scaling, seam carving, dynamic programming**

I. PENDAHULUAN

A. Persoalan Instagram

Salah satu situs jejaring sosial yang cukup populer adalah Instagram. Saat ini Instagram memiliki jumlah pengguna aktif sebanyak 150 juta pengguna [1]. Instagram memiliki perbedaan dengan situs jejaring sosial lain karena menggabungkan dengan fitur berbagi foto. Rata-rata 55 juta foto yang dibagikan setiap harinya dan total 16 miliar foto yang sudah pernah dibagikan sejak situs ini didirikan pada Oktober 2010[1].

Uniknya foto yang dibagikan di situs ini hanya bisa berbentuk persegi (aspect ratio 1:1). Hal ini menjadi persoalan ketika pengguna ingin berbagi foto dengan aspect ratio yang berbeda semisal foto orang dengan pemandangan seperti terlihat pada gambar 1.1. Cara umum yang dapat dilakukan antara lain dengan operator (1) cropping, (2) resizing, dan (3) filling. Dapat dilihat pada gambar 1.2-1.4 bahwa ketiga cara ini belum sempurna. Operator 1 menyebabkan sebagian dari konten foto yang penting tidak dapat ditampilkan. Operator 2 menyebabkan konten foto berubah bentuk menjadi aneh dan terdistorsi. Operator 3 menyebabkan ukuran foto menjadi lebih kecil dan ada bagian foto yang menjadi sia-sia.

B. Persoalan Web Page

Persoalan ini hanyalah salah satu contoh kecil dari sekumpulan persoalan sejenis yang lebih luas. Contoh persoalan lainnya adalah pada web. Saat ini orang mengakses web dari berbagai jenis perangkat, misalnya dari desktop, laptop, tablet, smartphone, dan featured phone. Perangkat-perangkat tersebut memiliki bentuk (aspect ratio), ukuran, dan densitas layar yang berbeda-beda. Untuk mengatasinya dikembangkan teknik perancangan web yang berbasis responsive/fluid page. Dengan fluid page maka layout dari halaman mampu menyesuaikan dengan layar dari perangkat. Teknik ini menjadi tren untuk saat ini dan merupakan default dari front-end framework populer semisal Bootstrap dan Foundation [4]. Akan tetapi fluid page hanya melakukan operator 1, 2, dan 3 yang memiliki kelemahan.

C. Content Aware Scaling

Oleh karena itu perlu ada operator baru yang mampu melakukan perubahan aspect ratio dari suatu gambar namun tidak memiliki kelemahan seperti operator 1, 2, dan 3. Operator ini harus mampu mempertahankan konten foto yang penting, tidak mengubah konten menjadi berbentuk aneh, serta tidak menyebabkan ada bagian foto yang menjadi sia-sia. Untuk saat ini operator yang paling mendekati adalah Content-Aware Scaling.

Content-Aware Scaling adalah salah satu fitur baru pada Adobe Photoshop CS4 yang dirilis pada tahun 2008 [2]. Photoshop sendiri merupakan software standar industri untuk manipulasi foto. Content-Aware Scaling adalah fitur "ajaib" yang mampu mengubah aspek ration sebuah foto dengan bentuk yang tidak aneh namun mampu mempertahankan konten yang penting. Contoh

Gambar 1.1 Foto asli, sumber [7]

Gambar 1.3 Menggunakan operator resizing

Gambar 1.4 Menggunakan operator filling

Gambar 1.2 Menggunakan operator cropping

Gambar 1.5 Menggunakan fitur Content-Aware Scaling pada Photoshop CS6

penggunaan fitur ini dapat dilihat pada gambar 1.5. Dapat dilihat bahwa hasil dari fitur ini lebih baik dari cara-cara sebelumnya pada gambar 1.2-1.4.

D. Replika Content Aware Scaling

Akan tetapi harga Photoshop ini sangat mahal bagi pengguna umum yang sekedar ingin mengubah foto menjadi persegi seperti pada kasus Instagram. Untuk saat ini harga Photoshop versi terbaru mencapai $699 [3]. Sedangkan untuk persoalan fluid-page maka Photoshop bukan merupakan solusi karena fitur Content-Aware Scaling ini membutuhkan interaksi pengguna. Selain itu kebutuhan CPU, RAM, dan storage yang besar dari Photoshop membuat solusi ini tidak layak untuk dijalankan di sebuah web server yang bersifat shared hosting ataupun web browser pada perangkat mobile.

Maka dari itu makalah ini akan membahas tentang metode Content-Aware Scaling dan mencoba mengimplementasikan replika dari fungsi tersebut, terutama yang terkait dengan dynamic programming. Harapan akhirnya adalah dapat dihasilkan modul/library kecil berbasis open source yang dapat menyamai fitur Content-Aware Scaling pada Photoshop. Dengan menghasilkan sebuah library open source diharapkan persoalan harga dan kebutuhan sumber daya yang besar dapat diatasi. Selain itu diharapkan akan banyak pengembang lain yang berkontribusi sehingga library ini dapat terus menjadi lebih baik.

II. SEAM CARVING

A. Sejarah

Fitur Content-Aware Scaling pada Photoshop berangkat dari sebuah algoritma yang disebut dengan Seam Carving [6]. Algoritma ini dikembangkan oleh Shai Avidan dan Ariel Shamir di Mitshubishi Electric Research Lab pada tahun 2007. Adobe selanjutnya melisensi algoritma ini secara non-eksklusif untuk digunakan sejak Photoshop CS6 [5]. Karena lisensi non-eksklusif inilah maka dalam makalah ini mencoba untuk mengimplementasikan algoritma tersebut.

B. Ide Dasar

Ide dasar dari Seam Carving adalah dengan melakukan carving terhadap seam yang ada pada suatu gambar. Yang dimaksud seam sendiri adalah sebuah jalur pixel yang menyeberangi gambar dari atas ke bawah atau dari kiri ke kanan. Penentuan seam ini berdasarkan pada perhitungan fungsi energi dari suatu gambar. Fungsi energi ini menghitung seberapa penting atau tidak pixel pada suatu gambar. Pixel-pixel (atau seam) yang paling tidak penting (memiliki nilai fungsi energi yang kecil) inilah yang akan menjadi kandidat utama untuk dibuang (carving) ketika gambar di-resize. Contoh visualisasi seam. dapat dilihat pada gambar 2.1

Gambar 2.1 (kiri) foto asli, (tengah) visualisasi seam, (kanan) hasil carving, sumber [8]

C. Langkah-langkah

Jadi ada dua hal utama dalam algoritma Seam Carving ini. Yang pertama adalah bagaimana menghitung fungsi energi dari suatu gambar, hal ini akan dibahas sekilas pada bab III. Yang kedua adalah bagaimana menentukan seam yang akan dihapus. Penentuan seam yang akan dihapus ini akan menggunakan dynamic programming. Pembahasan mendetail tentang dynamic programming ada bab 4 dan penentuan seam pada bab 5. Untuk algoritma Seam Carving secara garis besar dapat dilihat pada Pseudocode 2.1.

Pseudocode 2.1 Algoritma Seam Carving

```
procedure SeamCarving(image, size)
   energy <- FungsiEnergi(image)
   while (Size(image) > size)
      seam <- GetSeam(image, energi)
      DeletePixel(image, seam)
```

III. FUNGSI ENERGI

A. Definisi Energi

Energi sendiri memiliki pengertian yang berbeda-beda dalam image processing. Dalam konteks Content-Aware Scaling ini energi menunjukkan derajat kepentingan dari suatu konten foto. Misalkan pada gambar 3.1 orang dan kastil memiliki derajat kepentingan yang jauh lebih tinggi daripada rumput dan langit. Persoalannya adalah bagaimana untuk mengetahui dan menghitung derajat kepentingan (energi) tersebut.

B. Perhitungan Fungsi Energi

Perhitungan fungsi energi sendiri bukan merupakan topik utama dari makalah ini. Berbagai jenis fungsi energi secara detail dapat dilihat pada makalah lain seperti [6], [8], dan [9]. Implementasi dari fungsi energi yang akan digunakan juga merupakan fungsi energi yang paling sederhana. Ide dasar fungsi energi pada makalah ini menggunakan prinsip gradien. Semakin besar perbedaan suatu pixel dengan pixel di sekitarnya maka dapat dikatakan pixel tersebut lebih penting (berenergi besar). Hal ini secara intuitif dapat dilihat pada gambar 3.1 yang terlihat rumput dan langit yang kurang penting perbedaan warnanya relatif sedikit.

C. Metode Gradien

Untuk menghitung gradien dapat digunakan turunan parsial dari warna terhadap arah sumbu x (arah horizontal gambar) dan terhadap sumbu y (arah vertikal gambar). Rumus gradien ini dapat dilihat pada persamaan 3.1. Selanjutnya cukup dilakukan penjumlahan dari masing-masing kanal warna yang ada sesuai dengan persamaan 3.2. Untuk gambar yang umum menggunakan kanal R (Red), G (Green), dan B (Blue). Dari perhitungan fungsi energi inilah maka dapat dihasilkan gambar 3.2 yang merupakan visualisasi tingkat energi dari gambar 3.1. Dapat terlihat juga bahwa orang dan kastil memiliki tingkat energi yang lebih tinggi. Hal ini menunjukkan bahwa perhitungan fungsi energi dengan metode gradien pada gambar tersebut sudah tepat.

$$e(\mathbf{I}) = \left|\frac{\partial}{\partial x}\mathbf{I}\right| + \left|\frac{\partial}{\partial y}\mathbf{I}\right| \qquad (3.1)$$

$$e_{\text{total}}(\mathbf{I}) = e_{\text{red}}(\mathbf{I}) + e_{\text{green}}(\mathbf{I}) + e_{\text{blue}}(\mathbf{I}) \qquad (3.2)$$

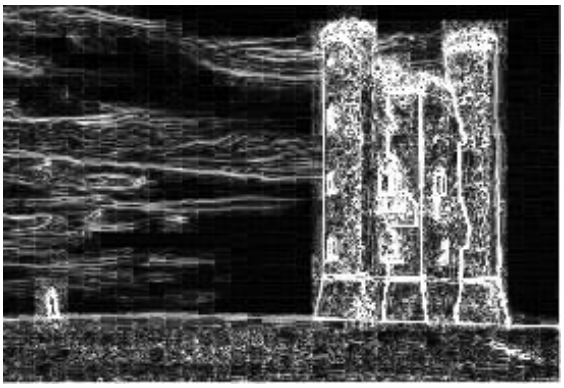
IV. DYNAMIC PROGRAMMING

Sebelum berlanjut ke langkah berikutnya dari Seam Carving di bab ini akan dibahas terlebih dahulu tentang dasar dari Dynamic Programming.

A. Ide Dasar

Ide dasar dari Dynamic Programming mirip dengan Divide and Conquer yaitu dengan membagi sebuah persoalan menjadi subpersoalan yang akan diselesaikan secara rekursif untuk akhirnya menggabungkan solusi dari subpersoalan tersebut [11]. Hal yang membedakan dengan Divide and Conquer adalah pada Dynamic Programming memanfaatkan hasil dari suatu subersoalan untuk persoalan yang lain. Hal ini bisa terjadi karena banyak subpersoalan yang salin overlap [12]. Ini sesuai dengan [15] yang mengatakan Dynami Programming = Rekursifitas + Memoisasi. Karena memanfaatkan memoisasi inilah Dynamic Programming bisa jauh lebih efisien daripada Divide and Conquer. Algoritma tidak perlu melakukan komputasi ulang untuk subproblem yang sama. Akan tetapi Dynamic Programming relatif

Gambar 3.1 Foto awal, sumber [10]



Gambar 3.2 Hasil perhitungan fungsi energi

membutuhkan memori yang lebih besar karena juga harus mencatat hasil dari subproblem yang ada.

B. Langkah-langkah

Dalam menyelesaikan persoalan dengan Dynamic Programming ada beberapa langkah yang dapat dilakukan. Langkah-langkah menurut [11] adalah:

1. Karakteristikkan struktur dari solusi optimal
2. Secara rekursif definisikan nilai ari solusi optimal
3. Hitung nilai dari solusi optimal, umumnya secara bottom-up
4. Bentuk solusi optimal dari hasil yang telah diperoleh

Sedangkan langkah-langkah menurut [14] adalah:

1. Deskripsikan array dari nilai yang ingin dihitung.
2. Membuat relaksi rekrusif dari suatu nilai dalam array ke nilai lainnya, definisikan juga untuk basisi.
3. Hitung nilai dari array menggunakan rekursifitas sebelumnya.
4. Tunjukkan solusi optimal menggunakan nilai dari array yang sudah didapat.

C. Contoh Fibonacci

Pseudocode 4.1 merupakan cara naif untuk menghitung nilai dari deret Fibonacci. Selanjutnya coba perhatikan simulasi ketika fungsi fib(5) dipanggil sebagai berikut:

1. fib(5)
2. fib(4) + fib(3)
3. (fib(3) + fib(2)) + (fib(2) + fib(1))
4. ((fib(2) + fib(1)) + (fib(1) + fib(0))) + ((fib(1) + fib(0)) + fib(1))
5. (((fib(1) + fib(0)) + fib(1)) + (fib(1) + fib(0))) + ((fib(1) + fib(0)) + fib(1))

Dapat dilihat bahwa fib(2) dipanggil sampai 3 kali. Hal ini tentu saja merupakan pemborosan, bayangkan untuk nilai fibonnaci yang lebih besar maka algoritma secara naif ini tidak dapat digunakan karena terlalu lambat. Bandingkan dengan Pseudocode 4.2 yang menerapkan prinsip Dynamic Programming. Setiap perhitungan disimpan di dalam array. Akibatnya perhitungan fib(2) cukup dilakukan sekali.

Pseudocode 4.1 Algoritma Fibonacci secara naif

```
function fib(n)
   if (n = 0)
      -> 0
   if (n = 1)
      -> 1
   -> fib(n - 1) + fib(n - 2)
```

Pseudocode 4.1 Algoritma Fibonacci dengan Dynamic Programming

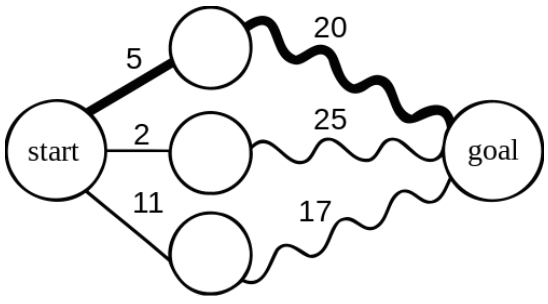
```
val <- {0,1}

function fib(n)
   if val[n] = empty
      val[n] <- fib(n-1) + fib(n-2)
   -> val[n]
```

D. Contoh Shortest Path

Contoh lain dari penerapan Dynamic Programming adalah persoalan Shortest Path. Ide dasar dari Shortest Path menggunakan Dynamic Programming adalah persamaan rekurens 4.1. Persamaan ini menunjukkan bahwa untuk mecari jalur terpendek dari Start menuju Goal sama dengan mencari minimum dari jarak ke node tetangga ditambah jarak dari node tersebut ke Goal. Tentu saja gambaran garis meliuk di Gambar 4.1 hanyalah visualisasi. Pada Graf sebenarnya garis tersebut bisa bertemu dengan garis dari node lainnya. Hal inilah yang dimanfaatkan untuk Dynamic Programming menggunakan memoisasi [15].

$$\mathrm{sp}(a, \mathrm{goal}) = \min\big(\mathrm{edge}(a, x) + \mathrm{sp}(x, \mathrm{goal})\big), \quad \forall x, x \in \mathrm{adj}(\mathrm{start}) \tag{4.1}$$



Gambar 4.1 Visualisasi Shortest Path dengan Dynamic Programming, sumber [16]

E. Persoalan Yang Cocok

Walaupun Dynamic Programming disebut sebagai sledgehammer dari alat untuk menyelesaikan persoalan [12] akan tetapi ada beberapa karakteristik umum persoalan yang cocok untuk menggunakan Dynamic Programming. Beberapa karakteristik [13] tersebut adalah:

- Setiap persoalan harus memiliki tahap dan pilihan yang selanjutnya dapat diambil. Misalkan shortest path memiliki tahap berupa graf dengan pilihan selanjutnya berupa node.
- Setiap tahap harus memiliki status. Misalkan pada shortest path status adalah node sekarang.
- Pilihan dari suatu tahapan akan berubah menjadi status pada tahapan selanjutnya. Misalkan pilihan node selanjutnya dari shortest path menentukan node sekarang dari tahap selanjutnya.
- Pada tahap tertentu, pilihan optimal dari status yang tersisa tidak bergantung pada status atau pilihan sebelumnya.
- Ada relasi rekourens antara stage j dan j + 1.
- Tahap final harus berupa basis.

V. PENENTUAN SEAM

A. Ide Dasar

Selanjutnya adalah tinggal menghapus bagian gambar yang memiliki energi paling rendah. Masalahnya jika kita menghapus pixel yang memiliki energi paling rendah pada bagian horizontal/vertikal secara langsung maka yang terjadi adalah distorsi. Akan tetapi, jika yang dihapus adalah baris/kolom yang memiliki energi paling kecil maka akan terjadi hal yang disebut artifak. Gambar hasil akan terlihat terpotong dan tidak menyatu [17]. Oleh karena itu muncul yang disebut Seam.

B. Definisi

Secara formal Seam vertikal didefinisikan seperti pada persamaan 5.1. Persamaan ini menunjukkan bahwa seam vertikal adalah sebuah jalur dari bagian atas gambar menuju bawah sehingga panjang dari jalur tersebut dalam piksel sama dengan lebar gambar. Serta untuk setiap elemen seam (i,j), elemen selanjutnya dari tiap seam hanya dapat terdiri dari (i+1, j-1), (i+1, j), atau (i+1, j+1). Secara serupa persamaan untuk seam horizontal dapat dilihat pada persamaan 5.2.

$$\mathbf{s^x} = \{s_i^x\}_{i=1}^n = \{(x(i), i)\}_{i=1}^n, \quad \text{s.t.} \forall i, |x(i) - x(i-1)| \leq 1, \quad \text{dengan } x: [1, \dots, \text{n}] \rightarrow [1, \dots, \text{m}] \qquad (5.1)$$

$$\mathbf{s^y} = \{s_j^y\}_{j=1}^n = \{(j, y(j))\}_{j=1}^n, \quad \text{s.t.} \forall j, |y(j) - y(j-1)| \leq 1, \quad \text{dengan } x: [1, \dots, \text{m}] \rightarrow [1, \dots, \text{n}] \qquad (5.2)$$

B. Metode

Selanjutnya adalah bagaimana untuk menentukan seam mana yang akan dihapus. Penghapusan seam akan menggunakan fungsi energi yang sudah diimplementasikan sebelumnya. Tujuan dari metode ini adalah untuk menghasilkan seam dengan total energi paling sedikit sesuai dengan persamaan 5.3. Salah satu metode yang dapat dilakukan adalah menggunakan dynamic programming seperti yang sudah dibahas sebelumnya.

$$s^* = \min \sum_{i=1}^{n} e\big(I(s_i)\big) \qquad (5.3)$$

Teknik Dynamic Programming yang digunakan adalah dengan melakukan traversal dari baris kedua hingga terakhir dan menghitung energi kumulatif minimal (M) dari setiap kemungkinan seam yang terhubung (i,j). Rumus rekursif dari teknik ini ada pada persamaan 5.4. Perhatikan bahwa rumus rekursif ini mirip dengan rumus rekursif untuk algoritma Shortest Path sebelumnya.

$$\text{M(i, j)} = e(i,j) + \min(M(i-1, j-1), M(i-1, j), M(i-1, j+1)) \qquad (4.1)$$

Setelah dilakukan proses ini maka nilai minimum dari baris terakhir akan menunjukkan akhir dari seam yang memiliki energi paling minimal. Untuk memperoleh jalurnya kita bisa melakukan runut balik ke baris awal.

VI. IMPLEMENTASI DAN HASIL

Dalam tahap pengujian awal di makalah ini implementasi dilakukan menggunakan MATLAB. MATLAB dipilih karena sudah memiliki fungsi-fungsi dasar dalam pengolahan gambar. Salah satu hasil dapat dilihat pada gambar 6.1.-6.3.

VIII. SIMPULAN DAN PENGEMBANGAN SELANJUTNYA

A. Hasil

Dari hasil implementasi dan hasil pada bab sebelumnya dapat diperoleh kesimpulan bahwa implementasi Seam Carving yang berusaha mereplika fitur Content-Aware Scaling sudah berhasil dengan baik. Dilihat secara sekilas tidak tampak perbedaan dengan implementasi Photoshop walaupun jika dilakukan perbesaran tetap saja ada perbedaan.

Gambar 6.1 Foto asli, sumber [18]

Gambar 6.2 Hasil Content-Aware Scaling Photoshop

Gambar 6.3 Hasil replika implementasi di MATLAB

B. Pengembangan

Pengembangan selanjutnya yang dapat dilakukan antara lain adalah (1) library c, (2) seam inserting, dan (3) adaptif operator. Pengembangan 1 bertujuan untuk menghasilkan apa yang menjadi tujuan awal dari pembuatan makalah ini. Menterjemahkan dari bahasa MATLAB menjadi bahasa C pada dasarnya cukup straightforward, masalahnya hanya di penggunaan library bawaan MATLAB yang belum ada seutuhnya pada bahasa C.

Pengembangan 2 bertujuan agar resize yang dilakukan tidak hanya untuk memperkecil gambar tetapi juga dapat dilakukan dengan memperbesar gambar. Idenya adalah bukan dengan menghapus seam dengan energi terendah tetapi justru menduplikasinya.

Pengembangan 3 bertujuan agar program bisa secara cerdas menggabungkan operator 1, 2, dan 3 sebelumnya dengan seam carving. Karena ada kondisi Diana Seam Carving sekalipun belum mencukupi dan perlu dikombinasikan dengan operator lain.

C. Kontribusi

Saat makalah ini ditulis sumber kode program memang belum dirilis secara open source. Akan tetapi dalam waktu dekat kode sumber program akan diletakkan di repository publik penulis di [https://github.com/ichlasul]. Pembaca yang tertarik untuk membaca dapat langsung mengunjungi halaman tersebut dan melakukan clone. Pengembang lain yang tertarik untuk mengembangkan menjadi library yang powerful dapat langsung melakukan fork, pull request, atau sekedar menambah issue terhadap repository tersebut. Penulis sangat mengharapkan kontribusi dari pembaca.

IX. UCAPAN TERIMA KASIH

Ucapan terima kasih diberikan kepada Robert Renaud, seorang software engineer di Google karena telah memberikan pencerahan tentang Content-Aware Scaling yang dapat diimplementasikan dengan dynamic programming sederhana.

DAFTAR PUSTAKA

[1] http://instagram.com/press/ (16 Desember 2013)
[2] http://www.photoshopsupport.com/photoshop-cs4/what-is-new-in-photoshop-cs4.html (16 Desember 2013)
[3] http://www.adobe.com/aboutadobe/pressroom/pressreleases/201204/042312AdobePhotoshopCS6.html (18 Desember 2013)
[4] http://responsive.vermilion.com/compare.php (Sudah Lama)
[5] http://www.reuters.com/article/pressRelease/idUS175954+16-Dec-2008+BW20081216 (16 Desember 2013)
[6] S. Avidan dan A. Shamir "Seam Carving for Content-Aware Image Resizing", ACM Transactions on Graphics, 2007.
[7] http://gigaom.com/2007/07/17/how-to-use-your-laptop-outside/ (03 Maret 2013)
[8] J. Kies dll "Seam Carving with Improved Edge Preservation"
[9] M. Rubinstein dll, "Improved seam carving for video retargeting", ACM Transactions on Graphics, 2008
[10] http://commons.wikimedia.org/wiki/File:Broadway_tower.jpg (16 Desember 2013)
[11] T. H. Cormen dkk, *Introduction to Algorithm 3rd Edition*, Cambridge: MIT Press, 2009
[12] S. Dasgupta dik, *Algorithm*, UC Berkeley
[13] http://mat.gsia.cmu.edu/classes/dynamic/node4.html (16 Desember 2013)
[14] CSC 364S Notes, University of Toronto, 2003
[15] http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-006-introduction-to-algorithms-fall-2011/lecture-videos/lecture-19-dynamic-programming-i-fibonacci-shortest-paths/ (18 Desember 2013)
[16] http://commons.wikimedia.org/wiki/File:Shortest_path_optimal_substructure.svg (18 Desember 2013)
[17] http://kirilllykov.github.io/blog/2013/06/06/seam-carving-algorithm/ (16 Desember 2013)
[18] http://www.flickr.com/photos/dago2/5851051153/ (20 Desember 2013)

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, 19 Desember 2013

Ichlasul Amal 13511075