PENERAPAN TEORI GRAF UNTUK MENYELESAIKAN MASALAH *MINIMUM SPANNING TREE* (*MST*) MENGGUNAKAN ALGORITMA KRUSKAL

Swaditya Rizki
Program Studi Pendidikan Matematika, Fakultas Keguruan dan Ilmu Pendidikan,
Universitas Muhammadiyah Metro, Lampung
Email: swaditya.rizki@gmail.com

ABSTRACT: *One of useful graph theory to solve the real problems is Minimum Spanning Tree (MST). MST is network optimization problems that can be applied in many fields such as transportations problems and communication network design (Gruber and Raidl, 2005). MST begins from tree namely a connected graph has no circuits. From the graph, there is a sub-graph that has all the vertex or spanning tree. If that graph has the weight/cost, then the spanning tree that has the smallest weight/cost is called Minimum Spanning Tree. Basic algorithm used to determine the MST is Kruskal's algorithm. This algorithm is known as one of the best algorithms for the optimization problems, especially for MST. In this paper is developed a source code program to determine MST using Kruskal's algorithm and then implemented on several data representing a complete graph.*

Keywords: Graph, Tree, Minimum Spanning Tree (MST), Kruskal's Algorithm.

1. Latar Belakang dan Masalah

Perkembangan ilmu pengetahuan dan teknologi sampai saat ini terus mengalami peningkatan yang signifikan. Para peneliti terus melakukan penelitian untuk selalu memunculkan penemuan-penemuan baru yang dapat memberikan sumbangan ilmu pengetahuannya sebagai penunjang berkembangnya ilmu-ilmu lain. Salah satunya adalah ilmu matematika yang sudah ditemukan oleh ilmuan-ilmuan ratusan tahun yang lalu, sekarang sangat berguna dalam perkembangan teknologi yang sampai saat ini dapat memudahkan manusia dalam menyelesaikan permasalahan yang ada. Teknologi komputer sendiri pembuatannya sebagian besar menggunakan logika matematika yang sampai saat ini terus berkembang pesat, sehingga ilmu matematika dapat digunakan untuk menyelesaikan berbagai permasalah-an dalam dunia nyata.

Salah satu contoh bagian dari ilmu matematika adalah teori graf. Berawal dari permasalahan jembatan Konigsberg yang memiliki tujuh buah jembatan,

penduduk kota tersebut ingin melewati jembatan tersebut tepat satu kali dan kembali lagi ke tempat awal keberangkatan. Dari permasalahan tersebut, seorang matematikawan Swiss, Leonard Euler menemukan jawaban dari permasalahan tersebut yaitu memodelkan masalah ini dengan cara merepresentasikan ke dalam graf. Dengan representasi tersebut dapat mempermudah menganalisis dan menemukan solusi dari suatu permasalahan yang sangat membutuhkan dana besar dan waktu lama untuk membuktikannya secara langsung.

Graf juga mempunyai banyak manfaat untuk optimasi dalam membangun jalan raya, rel kereta api, desain jaringan komunikasi, dll. Sebagai contoh, untuk membangun sebuah jalan raya yang menghubungkan beberapa kota, sangat dibutuhkan suatu desain dalam graf agar dapat mengoptimalkan pembangunan suatu jalan tersebut dengan dana yang minimal. Masalah ini dapat direpresentasikan ke dalam graf yaitu dengan menyatakan kota-kota sebagai titik/*node*/*vertex*, jalan raya sebagai garis/*edge*, dan biaya pembangunan jalan raya sebagai bobot dalam graf.

Penelitian ini berfokus pada algoritma Kruskal untuk masalah *Minimum Spanning Tree* (*MST*) yang terkait tentang optimasi dalam permasalahan transportasi khususnya pembuatan jalan raya yang menghubungkan lebih dari *n* kota dan permasalahan desain jaringan komunikasi untuk kecepatan transfer data. Permasalahan ini direpresentasi-kan ke dalam suatu graf, kemudian dicari pohon merentang minimum (*Minimum Spanning Tree*) menggunakan algoritma Kruskal.

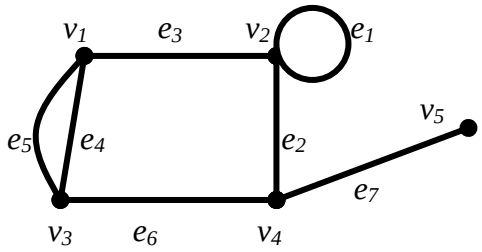
2. Tujuan Penelitian

Tujuan penelitian ini adalah memberikan suatu solusi untuk mendeskripsikan permasalahan transportasi dan desain jaringan komunikasi ke dalam suatu graf lengkap. Kemudian membentuk graf tersebut menjadi *Minimum Spanning Tree (MST)* dengan menggunakan algoritma Kruskal sehingga didapatkan biaya minimum pada permasalahan transportasi dan desain jaringan komunikasi dengan titik/*node* 10, 20, 30, 40, 50, 60, 70, 80, 90, 100. Proses ini kemudian diimplementasikan ke dalam sebuah program komputasi sehingga dengan mudah dan cepat untuk

mendapatkan solusi dari permasalahan tersebut.

3. Graf (*Graph*)

Graf *G* adalah suatu struktur (*V,E*) dimana $V = \{v_1, v_2, \ldots\}$ himpunan tak kosong dengan elemen-elemennya disebut *vertex* (titik), sedangkan $E = \{e_1, e_2, \ldots\}$ (mungkin kosong) adalah himpunan pasangan tak terurut dari elemen-elemen di *V(G).* Anggota dari *E(G)* disebut *edge* (sisi). Secara geometri graf digambarkan sebagai sekumpulan noktah (simpul) di dalam bidang dua dimensi yang dihubungkan dengan sekumpulan garis (sisi).

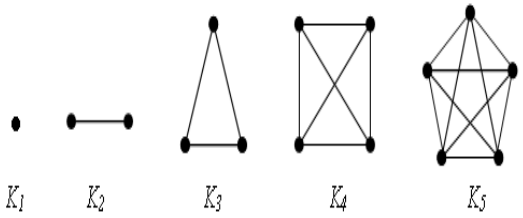


Gambar 1. Graf *G* dengan sebuah *loop* dan *parallel edges.*

e_4 dan e_5 dinamakan sisi ganda (*parallel edges*), sedangkan e_1 dinamakan *loop* karena ia berawal dan berakhir pada titik yang sama (Deo, 1989).

Definisi 1 Graf Lengkap (*Complete Graph*)

Graf lengkap adalah graf sederhana yang setiap titiknya mempunyai sisi ke semua titik lainnya atau graf yang setiap titiknya saling bertetangga. Graf dengan *n* buah titik dilambangkan dengan K_n. Setiap titik pada K_n berderajat *n-1*. K_1 sampai K_5 adalah contoh graf lengkap.



Gambar 2. Graf Lengkap $K_1 - K_5$.

Jumlah sisi pada graf lengkap yang terdiri dari *n* buah titik adalah $\frac{n(n-1)}{2}$. (Munir, 2001).

Definisi 2 Graf Berbobot (*Weighted Graph*)

Graf berbobot adalah graf yang setiap sisinya diberi sebuah harga (bobot). Bobot pada tiap sisi dapat menyatakan jarak antar dua buah kota, biaya perjalanan antar dua buah kota, waktu tempuh atau biaya instalasi dari sebuah simpul komunikasi ke simpul komunikasi lainnya dalam jaringan komputer, atau biaya pembangunan jalan raya antar dua buah kota (Munir, 2001).

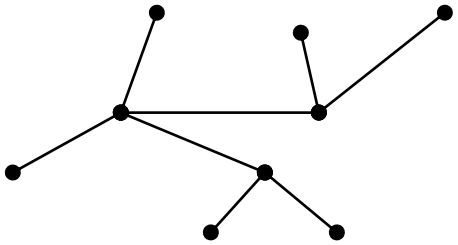
Definisi 3 Sirkuit *(Cycle)*

Lintasan yang berawal dan berakhir pada simpul yang sama disebut sirkuit atau siklus (Munir, 2001).

a. Pohon *(Tree)*

Definisi 4 Pohon

Tree adalah graf terhubung yang tidak mengandung sirkuit (Deo, 1989).



Gambar 3. Pohon *(Tree).*

Sifat-sifat dari pohon (*tree*) adalah sebagai berikut:

Misalkan $G = (V, E)$ adalah graf tak-berarah sederhana dan jumlah simpulnya n. Semua pernyataan di bawah ini adalah ekivalen:

1. G adalah pohon.
2. Setiap pasang simpul di dalam G terhubung dengan lintasan tunggal.
3. G terhubung dan memiliki $n - 1$ buah sisi.
4. G tidak mengandung sirkuit dan memiliki $n - 1$ buah sisi.
5. G tidak mengandung sirkuit dan penambahan satu sisi pada graf akan membuat hanya satu sirkuit.
6. G terhubung dan semua sisinya adalah jembatan (jembatan adalah sisi yang bila dihapus menyebabkan graf terbagi menjadi dua komponen)

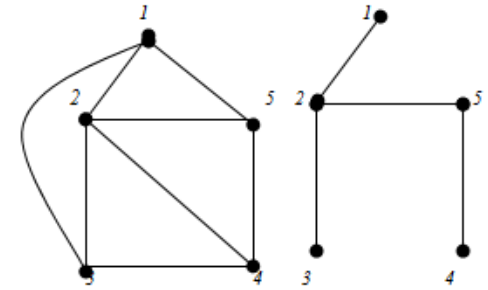
(Munir, 2001).

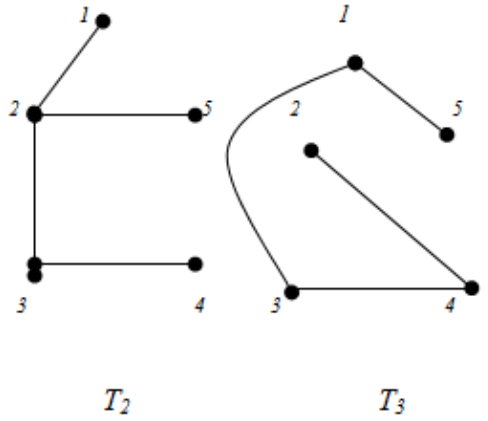
b. Pohon Merentang *(Spanning Tree)*

Misalkan G graf terhubung. *Tree* T dikatakan *spanning tree* dari G jika T adalah subgraf dari G yang memuat semua titiknya (Deo, 1989).

Misalkan $G = (V,E)$ adalah graf tak-berarah terhubung yang bukan pohon, artinya pada G terdapat sirkuit. G dapat diubah menjadi pohon $T = (V,E)$ dengan cara memutuskan sirkuit-sirkuit yang ada. Caranya yaitu dengan memutuskan salah satu sisi pada sirkuit hingga tidak ada sirkuit pada G. Jika di G tidak ada lagi sirkuit maka pohon T ini disebut dengan pohon merentang *(spanning tree)*. Disebut merentang karena semua simpul pada pohon T sama dengan simpul pada graf G, dan sisi pada $T \subseteq$ sisi pada G, dengan kata lain $V_1 = V$ dan $E_1 \subseteq E$ (Munir, 2001).

Contoh pembentukan *spanning tree.*



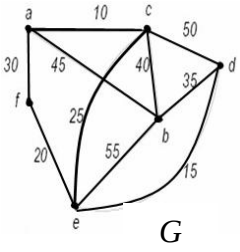


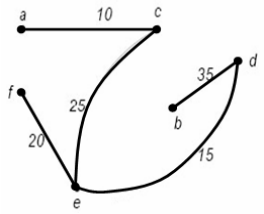
Gambar 4. Graf dan tiga buah *spanning tree* T_1, T_2, T_3.

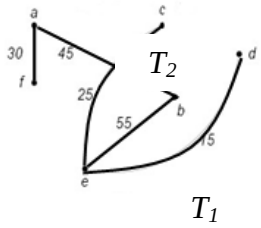
c. *Minimum Spanning Tree (MST)*

Jika *G* pada merupakan graf berbobot, maka bobot pohon merentang T_1 atau T_2 didefinisikan sebagai jumlah bobot semua sisi di T_1 atau T_2. Di antara pohon merentang yang ada pada *G*, yang paling penting adalah pohon merentang dengan bobot minimum. Pohon merentang dengan bobot minimum ini disebut dengan pohon merentang minimum atau *Minimum Spanning Tree (MST)*. Contoh aplikasi *MST* yang sering digunakan adalah masalah transportasi seperti pemodelan proyek pembangunan jalan raya menggunakan graf. *MST* digunakan untuk memilih jalur dengan bobot terkecil yang akan meminimalkan biaya pembangunan jalan.

Contoh graf dan pohon berbobot:







Gambar 5. *G* graf berbobot, T_1 danT_2 rentang pohon berbobot.

Dari graf berbobot *G*, akan ditentukan pohon merentang mana yang paling minimum. Apakah T_1 atau T_2?. Hal tersebut yang akan dicari dengan membangun pohon merentang minimum.

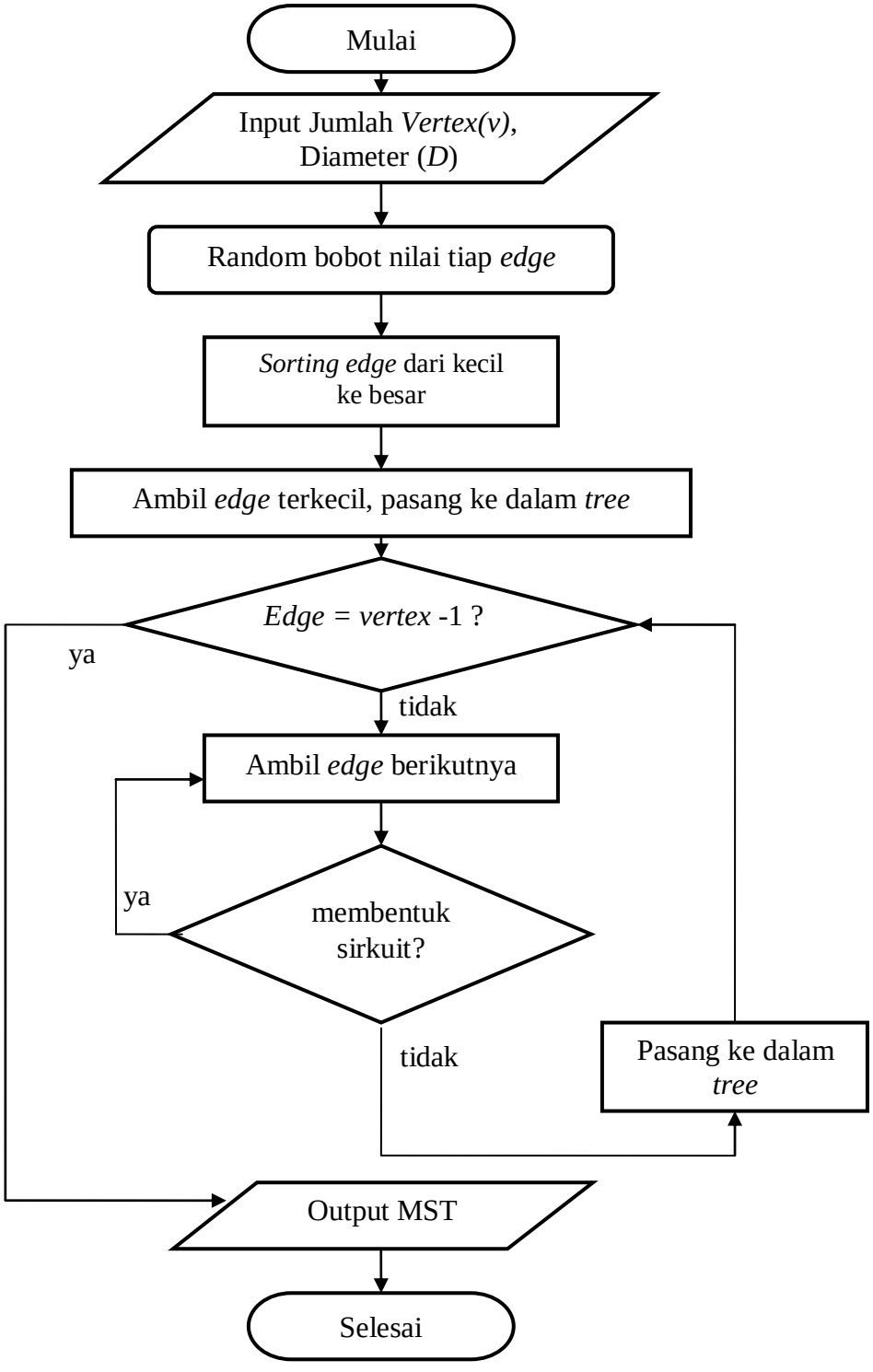
4. Algoritma Kruskal

(Thomas, *et.al*, 2001)

Adapun langkah-langkah algoritma Kruskal yaitu sebagai berikut:

1. Urutkan (*sorting*) biaya/bobot setiap *edge* dari biaya yang terkecil ke biaya terbesar.
2. Kemudian pilih *edge* terkecil dan pasang ke dalam *tree*.
3. Cek apakah jumlah $|E| = |V| - 1$. Jika tidak, ke Langkah 4. Jika ya, stop.
4. Pilih *edge* berikutnya dalam *sorting*.
5. Cek apakah pemasangan *edge* tersebut menyebabkan sirkuit/*cycle*. Jika tidak, masukkan *edge* ke dalam *tree* dan ke Langkah 6.
   Jika ya, ke Langkah 7.
6. Kembali ke Langkah 3.
7. Buang *edge* pada Langkah 5 dan kembali ke Langkah 4.

Diagram alir *Minimum Spanning Tree (MST)* adalah sebagai berikut:



Gambar 6. Diagram alir (*flowchart*) *MST*.

5. Hasil dan Pembahasan

Input program berupa jumlah *vertex*/titik. Dari jumlah *vertex* tersebut dibangkitkan bobot/biaya tiap *edge* untuk suatu graf lengkap. Kemudian data tersebut diurutkan berdasarkan bobot/biaya terkecil dan kemudian diperoleh solusi optimal *MST*. Sebagai contoh, akan dibangkitkan data secara acak dengan jumlah *vertex* 10, dari 10 *vertex* tersebut dihubungkan sebagai graf lengkap sehingga mempunyai *edge* sebanyak 45. Untuk proses menentukan graf lengkap dapat dilihat pada gambar 2. Kemudian dari graf lengkap tersebut dibangkitkan biaya/bobot secara acak untuk masing-masing *edge* dengan nilai yang bervariasi. Berikut ini hasil simulasi data untuk 10 *vertex*.

Tabel 1. Data graf lengkap dengan jumlah *vertex* 10.

Edge	Dari	Ke	Biaya
1	1	2	10
2	1	3	57
3	1	4	44
4	1	5	43
5	1	6	35
6	1	7	41
7	1	8	64
8	1	9	92
9	1	10	47
10	2	3	10
11	2	4	38
12	2	5	32
13	2	6	27
14	2	7	77
15	2	8	79
16	2	9	49
17	2	10	24
18	3	4	33
19	3	5	92
20	3	6	21
21	3	7	44
22	3	8	78
23	3	9	2
24	3	10	73
25	4	5	86

Edge	Dari	Ke	Biaya
26	4	6	22
27	4	7	23
28	4	8	75
29	4	9	44
30	4	10	44
31	5	6	59
32	5	7	71
33	5	8	9
34	5	9	52
35	5	10	44
36	6	7	32
37	6	8	77
38	6	9	5
39	6	10	13
40	7	8	96
41	7	9	91
42	7	10	64
43	8	9	96
44	8	10	6
45	9	10	48

Tabel 2. Proses mengurutkan data berdasarkan bobot/biaya.

Edge	Dari	Ke	Biaya
1	3	9	2
2	6	9	5
3	8	10	6
4	5	8	9
5	1	2	10
6	2	3	10
7	6	10	13
8	3	6	21
9	4	6	22
10	4	7	23
11	2	10	24
12	2	6	27
13	2	5	32
14	6	7	32
15	3	4	33
16	1	6	35
17	2	4	38
18	1	7	41
19	1	5	43
20	1	4	44
21	3	7	44
22	4	9	44
23	4	10	44
24	5	10	44
25	1	10	47

Edge	Dari	Ke	Biaya
26	9	10	48
27	2	9	49
28	5	9	52
29	1	3	57
30	5	6	59
31	1	8	64
32	7	10	64
33	5	7	71
34	3	10	73
35	4	8	75
36	2	7	77
37	6	8	77
38	3	8	78
39	2	8	79
40	4	5	86
41	7	9	91
42	1	9	92
43	3	5	92
44	7	8	96
45	8	9	96

Setelah proses pengurutan data, *edge* dipasang satu persatu ke dalam *tree* dimulai dari bobot biaya yang terendah sampai memenuhi *MST* dan proses akan berhenti jika jumlah *edge* = *vertex -1*. Adapun langkah-langkah dalam pemasangan *edge* ke dalam *tree* adalah sebagai berikut:

1. Pasang *edge* dari *vertex* 3 ke 9.
2. Pasang *edge* dari *vertex* 6 ke 9.
3. Pasang *edge* dari *vertex* 8 ke 10.
4. Pasang *edge* dari *vertex* 5 ke 8.
5. Pasang *edge* dari *vertex* 1 ke 2.
6. Pasang *edge* dari *vertex* 2 ke 3.
7. Pasang *edge* dari *vertex* 6 ke 10.
8. *Edge* dari *vertex* 3 ke 6 tidak boleh dipasang karena akan membentuk sirkuit.
9. Pasang *edge* dari *vertex* 4 ke 6.
10. Pasang *edge* dari *vertex* 4 ke 7.
11. Karena jumlah *edge* = *vertex-1*, maka proses pemasangan *edge* selesai sehingga didapatkan nilai minimum dari *spanning tree* sebagai berikut:

Tabel 3. Solusi optimal *MST* menggunakan algoritma Kruskal.

Edge		Biaya
Dari	Ke	
3	9	2
6	9	5
8	10	6
5	8	9
1	2	10
2	3	10
6	10	13
4	6	22
4	7	23
Total Biaya		100

Tabel 4. Solusi *MST* tanpa menggunakan algoritma.

Edge		Biaya
Dari	Ke	
1	2	10
1	3	57
1	4	44
1	5	43
1	6	35
1	7	41
1	8	64
1	9	92
1	10	47
Total Biaya		433

Dari Tabel 3 diatas tersebut didapat solusi optimal dari MST dengan total bobot/ biaya yaitu 100. Sedangkan pada Tabel 4 yaitu solusi MST tanpa menggunakan algoritma didapatkan bobot/biaya yaitu 433. Dari nilai tersebut dapat disimpulkan bahwa algoritma Kruskal dapat digunakan untuk mencari solusi optimal MST, bahkan nilai yang didapatkan jauh lebih minimum dibandingkan dengan solusi yang tidak menggunakan algoritma.

6. Kesimpulan

Data yang digunakan untuk contoh simulasi diatas hanya data dengan jumlah *vertex* 10, sedangkan untuk jumlah *vertex* lebih dari 10 bisa menggunakan program komputasi yang disini menggunakan bahasa pemrograman C/C++ sehingga akan lebih mengefisienkan waktu perhitungan. Program komputasi ini juga nantinya dapat dikembangkan untuk MST dengan algoritma yang lain ataupun dengan memodifikasi algoritma Kruskal untuk aplikasi graf lainnya.

DAFTAR PUSTAKA

Deo, N. 1989. *Graph Theory with Applications to Engineering and Computer Science*. Prentice Hall, Inc. Englewood Cliffs, New Jersey. 461 hlm.

Gruber, M. and Raidl, G.R. 2005. *Variable Neighborhood Search for the Bounded Diameter Minimum Spanning Tree Problem.* Institute of Computer Graphics and Algorithms, Vienna University of Technology. 18th Mini Euro Conference . Austria.

Munir, R. 2001. *Matematika Diskrit.* Informatika, Bandung. Hlm 353-456.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to Algorithms, Second Edition., 2001. ISBN 0-262-03293-7. Section 23.2: The algorithms of Kruskal and Prim, pp.567–574. MIT Press and McGraw-Hill.