Aplikasi Pohon Merentang Minimum dalam Membuat Rantai Komunikasi

Joshua Christo Randiny 13517063
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13517063@std.stei.itb.ac.id

***Abstrak*—Komunikasi adalah bagian penting dari hubungan manusia, terutama dalam sebuah kelompok. Salah satu metode untuk menyampaikan informasi dalam kelompok adalah rantai komunikasi. Satu orang akan menyampaikan informasi ke satu atau lebih orang setelahnya untuk terus diteruskan hingga orang terakhir. Rantai komunikasi dapat memberi kepastian jumlah orang yang menerima sebuah pesan. Akan tetapi biasa metode ini lambat dan tidak efektif. Oleh karena itu, dibutuhkan suatu cara untuk membuat sistem rantai komunikasi yang efektif. Salah satu cara yang memungkinkan adalah dengan menggunakan pohon merentang minimum untuk membentuk suatu rantai komunikasi berdasarkan kedekatan seseorang.**

***Kata Kunci* — pohon, pohon merentang minimum, komunikasi manusia, rantai komunikasi**

I. PENDAHULUAN

Dalam sebuah struktur organisasi, tentu dibutuhkan komunikasi yang baik antar anggotanya agar segala informasi bisa disampaikan dengan baik. Hal ini akan lebih penting lagi pada suatu susunan organisasi dengan satu sumber perintah, seperti pada kaderisasi, semua orang harus mengikuti perintah dari panitia. Untuk jumlah orang yang sedikit, hal ini adalah mudah. Namun, jika sudah melibatkan ratusan orang, hal tersebut menjadi sangat sulit.

Ada banyak solusi yang biasa ditawarkan dalam situasi ini, beberapa di antaranya adalah konfirmasi terpusat (misalkan melalui grup di sosial media), melalui sistem *buddy*, atau dengan sistem rantai komunikasi. Semua cara tersebut mempunyai kelebihan dan kekurangan masing-masing.

Sebagai salah satu peserta kaderisasi, masalah tersebut sangatlah penulis rasakan. Banyak informasi yang tidak tersampaikan dengan baik karena komunikasi yang kurang efektif. Hal tersebut adalah alasan penulis membuat makalah ini. Pada makalah ini penulis akan membahas salah satu metode yang ada yaitu rantai komunikasi. Rantai komunikasi melibatkan seseorang untuk menyampaikan suatu informasi kepada seseorang atau lebih yang akan diteruskan ke orang berikutnya, membentuk sebuah rantai atau pesan berantai.

Rantai komunikasi ini mempunyai kelebihan karena informasi yakin diterima seseorang. Jika pesan belum diterima, orang yang mengirimkannya akan tahu, berbeda dengan grup yang terkadang sulit untuk tahu siapa yang sudah membaca dan siapa yang belum. Akan tetapi, kekurangan dari rantai komunikasi adalah mungkinnya ada rantai yang putus sehingga informasi terputus.

Lebih spesifiknya yang akan dibahasa adalah cara membentuk rantai komunikasi ini. Cara yang biasa dilakukan adalah dengan mengacak suatu daftar nama untuk dihubungkan, dengan ketua biasa berada di paling pertama. Cara ini tidak efektif karena kemungkinan untuk putus di tengah sangatlah besar. Seseorang bisa saja bermusuhan dengan orang setelahnya, atau tidak mempunyai jalur komunikasi dengan orang tersebut.

Oleh karena itu, pada makalah ini akan dibahas cara lain untuk membentuk rantai ini menggunakan teori pohon. Teori pohon sendiri sudah ada sejak tahun 1857an dan sudah diaplikasikan di berbagai bidang. Salah satu dari teori ini adalah teori pohon merentang minimum.

II. GRAF, POHON, DAN POHON MERENTANG MINIMUM

A. GRAF

Sebuah graf didefinisikan sebagai pasangan himpunan (V,E), dengan V adalah himpunan dari simpul-simpul (*vertices*) yang tidak kosong dan E adalah himpunan sisi (*edges*) yang menghubungkan sepasang simpul. Dapat juga ditulis dengan G=(V,E) [1]. Dua buah simpul dikatakan bertetangga saat terdapat sebuah sisi yang menghubungkan kedua simpul tersebut.

Terdapat beberapa jenis graf, berdasarkan tipe dari sisinya, graf dapat dibedakan menjadi graf sederhana dan graf tak sederhana. Sebuah graf sederhana tidak mengandung sisi ganda (dua simpul yang dihubungkan dengan lebih dari 1 sisi) dan gelang (sisi yang berawal dan berakhir di simpul yang sama). Sedangkan berdasarkan arahnya, graf dabat dibedakan lagi menjadi dua yaitu graf berarah dan graf tak berarah.

Selain itu, terdapat beberapa terminologi dari graf.

- Ketetanggaan
  Dua buah simpul dikatakan bertetangga jika dihubungkan langsung oleh sebuah sisi.
- Bersisian
  Sebuah sisi yang menghubungkan v_m dan v_n disebut bersisian dengan simpul v_m dan v_n.
- Derajat
  Derajat dari suatu simpul adalah jumlah sisi yang bersisian dengan simpul tersebut.

- Lintasan

  Sebuah barisan berselang-seling simpul dan sisi yang berbentuk v_0, e_1, v_1, e_2, ..., v_{n-1}, e_n, v_n yang dibuat sedemikian rupa sehingga $e_1 = (v_0, v_1)$, e2 = (v_1, v_2), dan seterusnya sebagai sisi dari graf dapat disebut lintasan yang panjangnya n dari v_0 ke v_n.
- Siklus

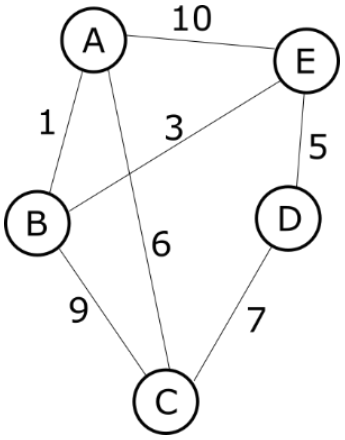
  Lintasan yang bermula dan berakhir di siklus yang sama dapat disebut siklus atau sirkuit.
- Terhubung

  Dua buah simpul dikatakan terhubung jika terdapat lintasan yang menghubungkan kedua simpul tersebut.
- Upagraf

  Jika G = (V,E) adalah sebuah graf, upagraf dari graf G adalah $G_1 = (V_1, E_1)$ dengan $V_1 \subseteq V$ dan $E_1 \subseteq E$.
- Upagraf rentang

  Upagraf yang mengandung semua simpul dari graf awalnya merupakan upagraf rentang. Jika G = (V,E) adalah sebuah graf, upagraf rentang dari graf G adalah $G_1 = (V_1, E_1)$ dengan $V_1 = V$ dan $E_1 \subseteq E$.
- Graf berbobot

  Sebuah graf berbobot adalah graf yang setiap sisinya mempunyai sebuah bobot



Gambar 1. Contoh graf berbobot
(Sumber : Dokumen penulis)

B. Pohon

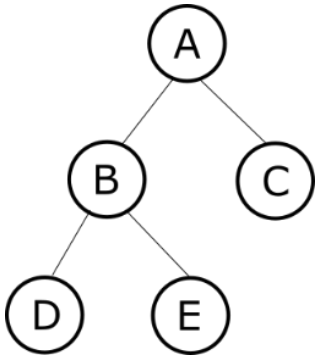
Pohon pada dasarnya merupakan sebuah graf yang tidak mempunyai sirkuit. Karena pada hakikatnya adalah graf, sebuah pohon dapat ditulis dengan cara yang sama dengan sebuah graf yaitu G = (V,E). Syarat lengkapnya adalah

- Setiap simpul di G terhubung dengan lintasan tunggal
- G tidak mengandung sirkuit dan penambahan satu sisi hanya akan membuat satu sirkuit
- G terhubung dan memiliki v-1 sisi (dengan v jumlah simpul)
- G terhubung

Jika keempat syarat tersebut terpenuhi, maka G dapat disebut pohon dengan v simpul.

Jika pada graf terdapat upagraf merentang, pada pohon, jika sebuah upagraf merentang membentuk pohon maka akan disebut pohon merentang. Pohon merentang diperoleh dengan memutus sirkuit pada graf. Setiap graf terhubung pasti memiliki paling sedikit satu buah pohon merentang.

Pada pohon ada beberapa terminologi penting. Mengacu pada contoh pohon berikut



Gambar 2. Contoh pohon
(Sumber : Dokumen penulis)

Simpul A merupakan orang tua dari simpul B dan C. Sedangkan simpul B dan C merupakan anak dari simpul A. Simpul D mempunyai saudara kandung E. D, E, dan C merupakan daun pada pohon tersebut dan B merupakan simpul dalam karena bukan daun dan juga bukan akar. Akar pada pohon tersebut adalah A.

C. Pohon Merentang Minimum

Jika terdapat suatu graf berbobot G, maka terdapat ≥ 1 pohon merentang. Dari berbagai pohon merentang ini, pasti terdapat satu yang mempunyai bobot minimum. Bobot sebuah pohon didapat dengan menjumlahkan bobot semua sisinya. Untuk membuat pohon merentang minimum dapat digunakan beberapa cara, dua algoritma yang cukup terkenal adalah Prim dan Kruskal.

Algoritma Prim bekerja dalam beberapa tahap. Pertama, akan diambil sisi dari graf dengan bobot minimum dan dimasukkan ke pohon. Kemudian, akan dicari sisi minimum yang bersisian dengan simpul di T untuk dimasukkan ke dalamnya. Langkah tersebut akan diulangi hingga pohon terbentuk (sebanyak v-2 kali).

Algoritma Prim mempunyai *pseudocode* berikut. Fungsi lowestCostEdge() akan mengembalikan *edge* dengan bobot terkecil dan tidak dapat membentuk *loop*.

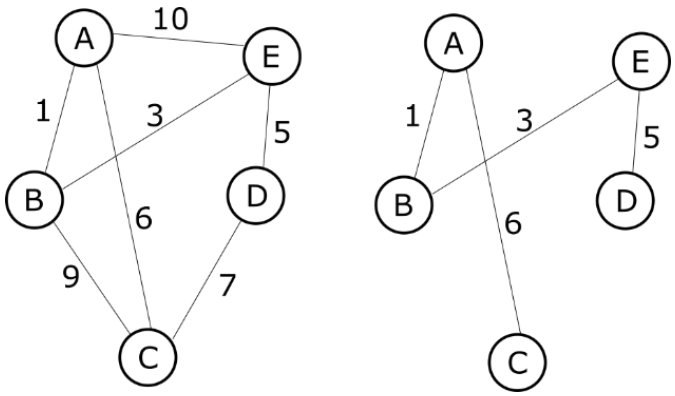
```
V ← allVertex()
T ← ∅
U ← { 1 } {Vertex visited}
while (U ≠ V)
   (u, v) ← lowestCostEdge()
  T ← T ∪ {(u, v)}
  U ← U ∪ {v}
```

Sedangkan untuk algoritma Kruskal, pertama akan diurutkan semua sisi dari bobot minimum hingga maksimum. Kemudian dipilih setiap sisi dari bobot minimum yang tidak membuat sebuah sirkuit pada pohon. Langkah tersebut akan diulangi hingga terbentuk pohon merentang (sebanyak v-1 kali)

Sedangkan untuk *pseudocode* dari algoritma Kruskal adalah sebagai berikut

```
T ← ∅
edge ← sortByWeight(edge)

For each edge (u, v) ∈ G.E
  if not createLoop(edge,T) then
    T ← T ∪ {(u, v)}
```



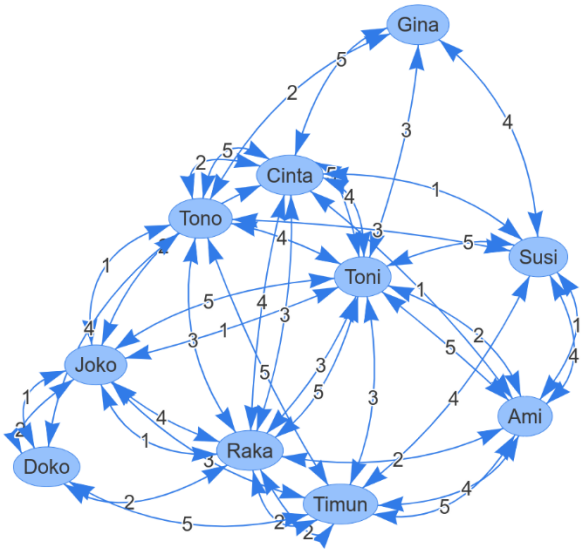
Gambar 3. Contoh pohon merentang minimum dari sebuah graf
(Sumber : Dokumen penulis)

Jika dilihat dari algoritma pembentuknya, dapat diketahui bahwa pohon merentang minimum tidaklah unik, karena ada kemungkinan untuk beberapa sisi dengan bobot yang sama.

III. PEMODELAN HUBUNGAN

Manusia sebagai makhluk sosial tentu memerlukan seorang teman. Akan tetapi tentu sebagai manusia, kita tidak mungkin berteman dengan semua orang. Terdapat batasan tentang jumlah teman dekat yang seseorang miliki. Jika tingkat kedekatan hubungan manusia dibagi menjadi beberapa tingkat. Seseorang yang berada pada tingkat pertama adalah teman yang sangat dipedulikan dan memegang peran penting. Di sisi lain, teman pada tingkat 5, adalah kenalan biasa atau yang biasa disebut *acquaintance*. Orang di atas tingkat 5 sudah tidak mempunyai koneksi dengan orang tersebut, hanya diingat nama atau wajahnya. Jumlah orang di tiap lapisan teman akan berkisar pada kelipatan 2-3 setiap naik tingkat [2]. Misalkan pada tingkat 1 ada 3 orang, maka pada tingkat 2 akan ada 10-15 orang, dan seterusnya hingga tingkat 5 akan berada pada angka sekitar 100-150.

Jika teori sebelumnya meneliti hubungan manusia pada komunikasi di dunia nyata. Menurut riset lain pada dunia maya, didapat bahwa angka-angka tersebut berkisar pada angka 4.1, 11.0, 29.8, 128.9 untuk tiap tingkatnya dengan tingkat pertama untuk teman dekat [3]. Dari hasil tersebut dapat dimodelkan sebuah graf dengan tiap sisinya mempunyai bobot sesuai tingkatan pertemanannya, satu untuk hubungan dekat, 5 untuk kenalan, 6 untuk orang yang hanya diketahui namanya dan 7 untuk yang benar-benar tidak dikenal. Akan tetapi, untuk mempermudah model akan dibuat hanya dari 1-5, selebihnya akan dianggap tidak berhubungan. Diasumsikan informasi yang ingin disampaikan butuh disampaikan dengan cepat, sehingga kemungkinan untuk memberi tahu ke orang di level >5 menjadi terlalu kecil.



Gambar 4. Contoh graf hubungan
(Sumber : Dokumen penulis)

Untuk membuat graf hubungan seperti di atas, akan digunakan sebuah algoritma sederhana. Algoritma ini bekerja dengan melakukan pengulangan sebanyak jumlah *node* yang diingini. Untuk setiap *node* kemudian akan dibuat *edge* sebanyak maksimal 150 (akan diacak dengan variansi sebanyak maksimal 20 persen dari angka tetapan). *Edge* disebut valid jika belum ada *edge* yang menghubungkan kedua simpul (satu sisi dianggap bulak balik).

```
g             : graph
nodes         : integer
dest          : node
i,j           : integer
name          : array of string
levelCount    : array of integer

input(nodes)
name ← getRandomNameList()
levelCount ← [2,5,11,28,89]

i traversal [1..nodes]
  {add node with id i and random label}
  g.addNode(i,getRandom(name))

  {add edge}
  j traversal [1..5]
    repeat variance(levelCount[j]) times
      {add edge from i to random with level j}
    do
      dest ← getRandomNode()
    while (not edgeExists(i,dest))
    g.addEdge(i,dest,j)
```

Waktu yang dibutuhkan untuk menyampaikan pesan ke orang dengan level berbeda tentu tidak sama. Teman pada level 1

(teman dekat) tentu akan lebih mudah didekati. Di sisi lain jika diperintahkan untuk memberitahu informasi ke orang di level 5, tentu akan lebih sulit. Dari riset lain didapat angka perbandingan jumlah komunikasi ke orang-orang di level yang berbeda dan jumlahnya pada sosial media [4].

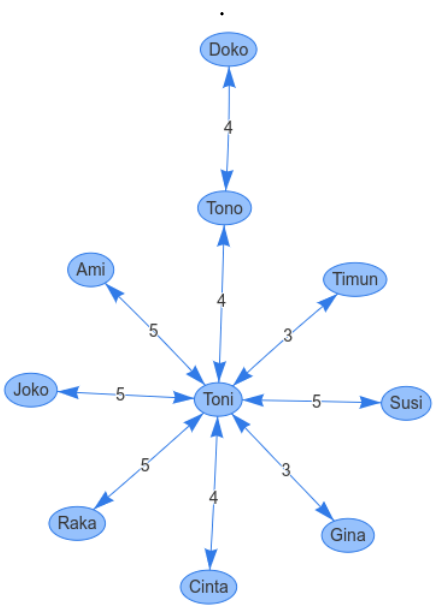
Level Hubungan	Jumlah teman	Rata-rata Komunikasi (komunikasi/satuan waktu)
1	2	275
2	5	113
3	11	49
4	28	17
5	89	3

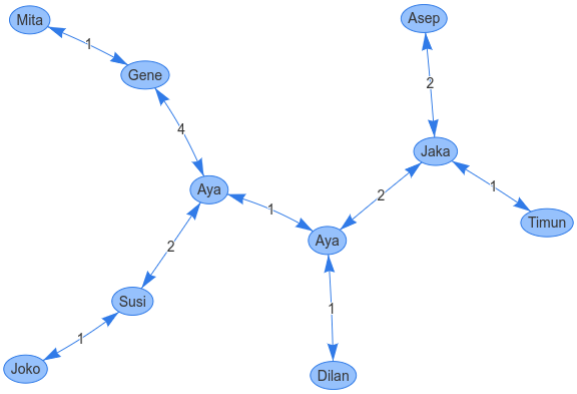
Tabel 1. Rata-rata komunikasi di level hubungan berbeda

Tabel di atas merupakan hasil dari analisis terhadap komunikasi yang dilakukan di sosial media Twitter yang cukup merepresentasikan penggunaan sosial media secara umum. Jika diambil satu jangka waktu tertentu (Misal 1 detik), kemungkinan untuk menghubungi seseorang di level 1 adalah 125 kali lebih besar dibanding orang di level 5.

Dari data tersebut dapat dibuat sebuah model komunikasi dengan tiap orang mempunyai 5 level hubungan yang jumlahnya bervariasi. Variasi akan ditentukan untuk berdeviasi maksimal 20% dari angka pada tabel. Misalkan untuk level hubungan 3, ada kemungkinan antara 8-14 teman pada level tersebut. Setiap orangnya akan direpresentasikan dengan 1 *node*/*vertex* pada sebuah graf berbobot. Sisi dari graf akan merepresentasikan hubungan dengan seseorang, dengan bobotnya berisi level hubungan orang tersebut.

Dari graf yang ada kemudian akan dibuat sebuah pohon merentang biasa dan pohon merentang minimum menggunakan algoritma Prim. Dari kedua pohon akan dijalankan simulasi untuk menentukan waktu yang dibutuhkan agar semua orang pada pohon tersebut menerima informasi yang disampaikan. Untuk itu akan diacak suatu angka dari 1 hingga 275, jika masuk ke dalam *range* rata-rata komunikasi pada tabel 1, maka pesan akan dianggap tersampaikan, jika tidak, pesan tidak tersampaikan dan akan dicoba lagi.





Gambar 5. Contoh pohon merentang minimum dan acak
(Sumber : Dokumen penulis)

IV. Simulasi

A. Catatan Implementasi

Simulasi akan dilakukan dengan mengimplementasi sebuah algoritma sederhana. Algoritma tersebut pertama akan memilih sebuah *node* sebagai akar. Akan diulang terus menerus hingga semua *node* telah dikunjungi. Untuk itu, dibutuhkan sebuah variabel nodeVisited. Saat mengunjungi sebuah *node*, yang pertama dilakukan adalah mengecek apakah *node* tersebut sudah dikunjungi, jika sudah, abaikan. Jika belum dikunjungi atau masih ada *edge* yang belum berhasil ditelusuri, akan dicek untuk semua *edge* di *node* tersebut. Akan terdapat 2 kemungkinan, komunikasi tercapai atau tidak. Keberhasilan suatu komunikasi tergantung pada level pertemanan yang disimpan pada *edge*. Berhasil atau tidaknya operasi tetap akan menambah sebuah *counter*. *Counter* ini berfungsi sebagai penghitung waktu informasi dapat disampaikan ke seluruh *node*.

Berikut ini terdapat *pseudocode* untuk program yang dipakai

```
nodeVisited : array of boolean
g           : graph
e           : edge
tick        : integer
nodes       : array of node
nodeCount   : integer

g ← generateRandomGraph(nodeCount)

g ← calculateMST("prim") {or random}

nodes[0] ← getRandomNode(g)

while (nodeVisited.contain(false)) do
  for every node in nodes
    {Check if node has been visited}
    if not nodeVisited[node.id] then
      {Check every edge}
      for every edge in node.edge
        {If node has not been visited}
        if not nodeVisited(edge.dest) then
```

```
            {Random chance depend on level}
            if successComm(edge.weight) then
              nodes.push(edge.destination)

        if allEdgeExplored(node) then
          nodeVisisted[node.id] ← 1
    tick ← tick + 1
{not nodeVisited.contain(false)}

{all node visited}
output(tick)
```

Misal, ada pohon seperti pada gambar 5 bagian pohon merentang acak. Jika diasumsikan terpilih Raka sebagai akar maka program akan memulai dengan menaruh Raka pada variabel nodes. Setelah itu akan dilanjutkan dengan melihat setiap sisi yang terhubung dengan Raka. Jika dilihat, Yang terhubung dengan Raka hanyalah Toni dan levelnya 5. Akan ada kemungkinan 1/275 untuk sebuah informasi tersampaikan ke Toni.

Jika informasi sudah disampaikan ke Toni, Toni akan ditambahkan ke variabel nodes dan karena *node* Raka hanya mempunyai satu sisi, maka nodeVisited untuk Raka akan bernilai benar/*true*. Berikutnya saat mengulang, *node* Raka tidak akan dicek lagi, karena sudah benar. *Node* Toni akan dicek, terdapat 8 sisi, akan tetapi satu *node* telah dikunjungi (Raka), sehingga hanya ada 7 *node* yang akan dicoba dikomunikasikan.

Setiap kali mencoba memproses satu *node*, tick akan bertambah sebanyak satu. Ini untuk mencatat waktu simulasi, waktu yang dibutuhkan untuk menyampaikan pesan ke semua orang. Saat variabel nodeVisited sudah tidak mengandung nilai *false*, maka dianggap bahwa semua *node* telah dikunjungi. Program akan keluar dengan mencetak jumlah tick yang dibutuhkan untuk menyelesaikannya

Untuk mendapatkan hasil yang baik, simulasi akan diulang sebanyak 10 kali untuk tiap jumlah *node* (5 kali untuk pohon merentang minimum dan 5 kali untuk pohon merentang acak). Jumlah *node* yang akan dicoba adalah 50, 100, 150, 200, 250, dan 300. Jadi total akan terdapat 60 data.

Oleh karena angka yang acak, percobaan dengan hasil yang terlalu jauh akan diabaikan. (Misalkan nilai 10000 di rata-rata 4000). Hal tersebut bisa terjadi karena penggunaan angka acak baik untuk akar maupun dalam menentukan kemungkinan komunikasi

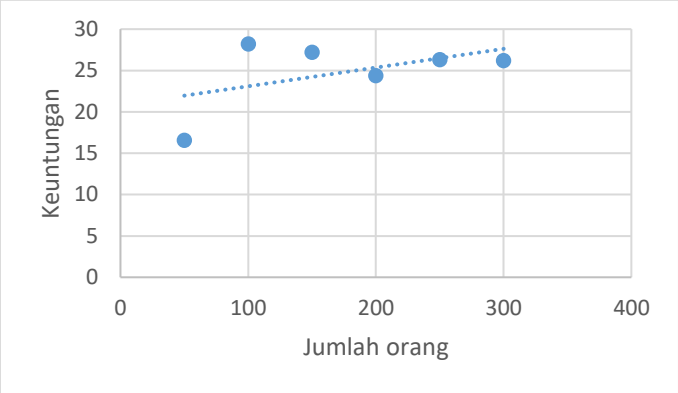
B. Hasil

Dari hasil simulasi, didapat data berikut ini

Pohon merentang acak						
Jumlah Orang	Hasil Simulasi (Satuan Waktu)					Rata-rata (Satuan Waktu)
50	1482	1344	2453	1601	3409	2057,8
100	2220	4515	5443	4729	3449	4071,2
150	3485	4975	4359	4188	5410	4483,4
200	3230	4344	5162	5651	5399	4757,2
250	4993	5474	5419	6231	5852	5593,8
300	5544	4966	5729	5380	7297	5783,2
Pohon merentang minimum						
Jumlah Orang	Hasil Simulasi (Satuan Waktu)					Rata-rata (Satuan Waktu)
50	42	134	52	174	220	124,4
100	112	92	353	84	81	144,4
150	139	118	101	129	338	165
200	193	180	165	319	119	195,2
250	139	159	443	179	144	212,8
300	377	235	203	168	122	221

Tabel 2. Hasil data simulasi

Dari tabel di atas, sangat terlihat jelas bahwa metode pohon merentang minimum jauh lebih cepat dibanding metode acak. Keuntungan dari metode ini juga tidak mengalami perubahan cukup signifikan saat jumlah orang dinaikkan, angkanya berkisar pada sekitar 25 kali lipat lebih cepat dalam menyampaikan informasi.



Grafik 1. Keuntungan waktu pohon merentang minimum

Jika dilihat ada banyak variasi pada data yang didapat. Hal tersebut mungkin disebabkan karena sistem simulasi yang memilih secara acak *node* yang akan dijadikan akar. Selain itu, untuk setiap kali uji coba, graf dibuat ulang, sehingga berbeda tiap kali percobaan. Akan tetapi karena dicoba sebanyak 5 kali, data cukup akurat dalam merepresentasikan hasil.

Peningkatan waktu yang signifikan ini tidak hanya berguna dalam meningkatkan kecepatan pengiriman pesan. Jika dalam keadaan terdesak, penggunaan metode ini dapat membuat perbedaan antara pesan tersampaikan ke berapa persen orang. Jika misalkan pada suatu acara dengan 150 orang, dibutuhkan agar suatu informasi tersampaikan dalam 200 satuan waktu. Jika menggunakan metode pohon merentang acak, hanya mungkin sekitar 5 persen orang yang mendapatkan informasi tersebut (asumsi tidak ada fitur *group chat*), jika menggunakan pohon merentang minimum, angka tersebut bisa mencapai 100 persen.

V. KESIMPULAN DAN SARAN

Komunikasi merupakan hal yang penting dalam dunia ini. Salah satu bentuk komunikasi adalah pesan berantai. Metode ini berguna dalam menyampaikan informasi secara cepat dan pasti (karena tidak ada alasan tidak membuka grup jika dikirim pribadi). Namun, untuk membuatnya lebih efisien, dapat digunakan teori pohon merentang minimum.

Dengan pohon merentang minimum, bisa didapat peningkatan pada kecepatan penyampaian pesan hingga 25 kali dibandingkan dengan metode acak. Angka ini juga konsisten seiring dengan peningkatan jumlah orang.

Untuk pengembangan, dapat dilakukan eksperimen dengan data asli dan bukan simulasi. Dapat juga disimulasikan pada graf yang cukup besar untuk hubungan level >5 berpengaruh. Selain itu juga dapat dibuat suatu cara agar yang menjadi akar dari sebuah pohon bukanlah *node* acak, melainkan *node* yang berada paling dekat dengan semua *node*.

VII. Ucapan Terima Kasih

Puji syukur kepada Tuhan Yang Maha Esa atas segala pertolongannya yang menyanggupkan penulis untuk menyelesaikan makalah ini. Penulis juga ingin mengucapkan terima kasih kepada Bapak Judhi Santoso sebagai pengajar mata kuliah Matematika Diskrit pada kelas K03 selama semester ini, tidak lupa juga kepada Bapak Rinaldi Munir dan Ibu Harlili. Kepada keluarga dan teman yang telah membantu dari berbagai bidang baik langsung maupun tidak langsung, penulis ucapkan terima kasih.

Referensi

[1] R. Munir, *Matematika Diskrit*, Bandung: Penerbit Informatika, 2003, edisi ketiga.

[2] W. Zhou, D. Sornette, R. Hill, R. Dunbar, "Discrete Hierarchical Organization of Social Group Sizes," *Proceedings. Biological sciences / The Royal Society*. Vol. 272, No. 439-44, 2004.

[3] P. M. Carron, K. Kaski, R. Dunbar, "Calling Dunbars numbers," *Social Networks*, Vol. 47, No. 151-155, 2016.

[4] R. Dunbar, V. Arnaboldi, M. Conti, A. Passarella, "The structure of online social networks mirrors those in the offline world," *Social Networks*, Vol. 43, No. 39-47, 2015.

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, 9 Desember 2018

Joshua Christo Randiny
13517063