# BAB 2

# LANDASAN TEORI

## 2.1 *Problem, Algorithm,* dan Analisis Algoritma

### 2.1.1 Definisi *Problem*

*Problem* (permasalahan) adalah sebuah kendala atau hambatan di mana membuat suatu tujuan sulit untuk dicapai, hal ini berhubungan dengan siuasi, kondisi, atau persoalan-persoalan yang belum terselesaikan. Sebuah *problem* timbul dikarenakan seorang individu menjadi sadar akan perbedaan yang berarti antara apakah yang sebenarnya dan apakah yang merupakan hasrat atau keinginan. Setiap *problem* membutuhkan jawaban atau solusi. Suatu *problem* dikatakan telah selesai apabila tujuan telah dicapai atau sudah tidak ada masalah lagi.

### 2.1.2 Definisi *Algorithm*

*Algorithm* berasal dari kata *Algoris* dan *Ritmis*, kata-kata ini berasal dari seorang ahli matematika, Mohammed ibn-Musa al-Khwarizmi, yang merupakan bagian dari *royal court*, *Baghdad*, dan hidup antara tahun 750 sampai 850. *Algorithm* adalah sebuah prosedur yang terstruktur dan dituliskan secara sistematis untuk menyelesaikan sebuah tugas di mana, memberikan *initial state* (keadaan awal), dan

akan *terminate* di akhir (*end state*) dengan batuan computer

Teori *Algorithm* menurut Horowitz:

1. *Input*, nol atau sejumlah kuantitas yang disuplai secara eksternal.
2. *Output,* paling sedikit dihasilkan satu kuantitas.
3. *Definiteness*, setiap instruksi jelas dan tidak ambigu.
4. *Finiteness*, jika suatu instruksi algoritma akan ditelusuri, dan dalam semua kasus, algoritma berakhir dalam beberapa langkah terbatas.
5. *Effectiveness*, setiap instruksi harus bersifat mendasar sehingga mudah diterapkan, secara prinsip dapat dikerjakan oleh seseorang walaupun dengan menggunakan pensil dan kertas.


Gambar 2.1 Hubungan masalah, algoritma, dan solusi

### 2.1.3 Definisi Analisis Algoritma

Algoritma tidak selalu memberikan hasil terbaik yang dapat diperoleh, maka diharapkan adanya suatu evaluasi hasil dari algoritma tersebut. Jika sebuah algoritma diberikan untuk menyelesaikan suatu permasalahan dan akan memberikan hasil yang

diharapkan, maka langkah selanjutnya adalaah menganalisis algoritma. Menganalisis algoritma adalah untuk menetapkan sejumlah sumber (seperti waktu dan *storage*) yang dibutuhkan untuk pengeksekusian, sehingga kita dapat menentukan besar biaya yang diperlukan algoritma tersebut untuk memperoleh hasil tersebut. Ukuran biaya eksekusi suatu algoritma yang paling sering digunakan adalah lamanya waktu yang diperlukan, di samping itu ada juga tolak ukur lainnya, misalnya besarnya memori yang diperlukan.

Kebanyakan algoritma dirancang untuk bekerja dengan *input* yang panjangnya tidak terbatas. Biasanya efisiensi atau kompleksitas dari sebuah algoritma disamakan dengan sebuah fungsi yang berkaitan dengan panjangnya *input*- banyaknya langkah (kompleksitas waktu) atau lokasi penyimpanan.

## 2.2 Jaringan Komputer

### 2.2.1 Definisi Jaringan computer

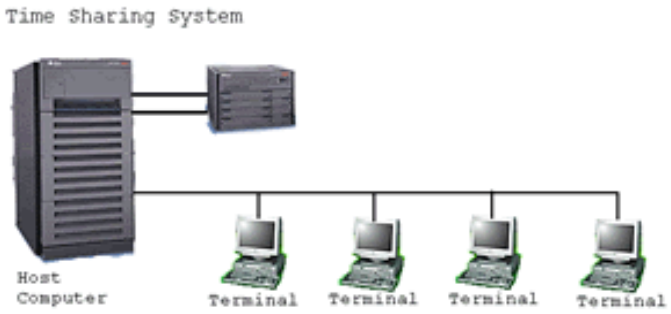
Jaringan komputer adalah sebuah kumpulan komputer, printer dan peralatan lainnya yang terhubung dalam satu kesatuan. Informasi dan data bergerak melalui kabel-kabel atau tanpa kabel sehingga memungkinkan pengguna jaringan komputer dapat saling bertukar dokumen dan data, mencetak pada *printer* yang sama dan bersama-sama menggunakan *hardware*/*software* yang terhubung dengan jaringan. Setiap komputer, printer atau periferal yang

terhubung dengan jaringan disebut *node*. Sebuah jaringan komputer dapat memiliki dua, puluhan, ribuan atau bahkan jutaan node.

### 2.2.2 Sejarah Jaringan Komputer

Konsep jaringan komputer lahir pada tahun 1940-an di Amerika dari sebuah proyek pengembangan komputer MODEL I di laboratorium Bell dan group riset Harvard University yang dipimpin profesor H. Aiken. Pada mulanya proyek tersebut hanyalah ingin memanfaatkan sebuah perangkat komputer yang harus dipakai bersama. Untuk mengerjakan beberapa proses tanpa banyak membuang waktu kosong dibuatlah proses beruntun (*Batch Processing*), sehingga beberapa program bisa dijalankan dalam sebuah komputer dengan dengan kaidah antrian.

Ditahun 1950-an ketika jenis komputer mulai membesar sampai terciptanya super komputer, maka sebuah komputer mesti melayani beberapa terminal (lihat Gambar 2.2) Untuk itu ditemukan konsep distribusi proses berdasarkan waktu yang dikenal dengan nama TSS (*Time Sharing System*), maka untuk pertama kali bentuk jaringan (*network*) komputer diaplikasikan. Pada sistem TSS beberapa terminal terhubung secara seri ke sebuah host komputer. Dalam proses TSS mulai nampak perpaduan teknologi komputer dan teknologi telekomunikasi yang pada awalnya berkembang sendiri-sendiri.




Gambar 2.2 Jaringan komputer model TSS

Memasuki tahun 1970-an, setelah beban pekerjaan bertambah banyak dan harga perangkat komputer besar mulai terasa sangat mahal, maka mulailah digunakan konsep proses distribusi (*Distributed Processing*). Seperti pada Gambar 2.3, dalam proses ini beberapa host komputer mengerjakan sebuah pekerjaan besar secara paralel untuk melayani beberapa terminal yang tersambung secara seri disetiap host komputer. Dala proses distribusi sudah mutlak diperlukan perpaduan yang mendalam antara teknologi komputer dan telekomunikasi, karena selain proses yang harus didistribusikan, semua host komputer wajib melayani terminal-terminalnya dalam satu perintah dari komputer pusat.


Gambar 2.3 Jaringan komputer model distributed processing

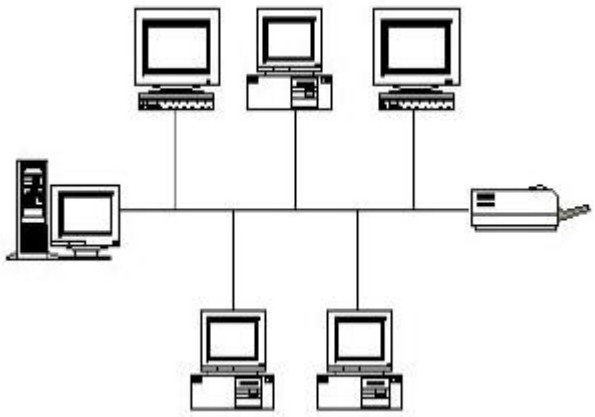
Selanjutnya ketika harga-harga komputer kecil sudah mulai menurun dan konsep proses distribusi sudah matang, maka penggunaan komputer dan jaringannya sudah mulai beragam dari mulai menangani proses bersama maupun komunikasi antar komputer (*Peer to Peer System*) saja tanpa melalui komputer pusat. Untuk itu mulailah berkembang teknologi jaringan lokal yang dikenal dengan sebutan LAN. Demikian pula ketika Internet mulai diperkenalkan, maka sebagian besar LAN yang berdiri sendiri mulai berhubungan dan terbentuklah jaringan raksasa WAN.

### 2.2.3 Topologi Jaringan Komputer

Topologi jaringan adalah hal yang menjelaskan hubungan geometris antara unsur-unsur dasar penyusun jaringan, yaitu *node, link*, dan *station*. Topologi

jaringan dapat dibagi menjadi topologi *bus*, topologi *ring*, topologi *star*, dan topologi *mesh*. Masing-masing dari topologi tersebut mempunyai kelebihan dan kekurangan. Macam-macam topologi jaringan:

1. Topologi *Bus*



Gambar 2.4 skema topologi *bus*

Kelebihan: - Hemat kabel

- Layout kabel sederhana
- Pengembangan atau penambahan *workstation* baru dapat dilakukan dengan mudah tanpa mengganggu *workstation* lain.

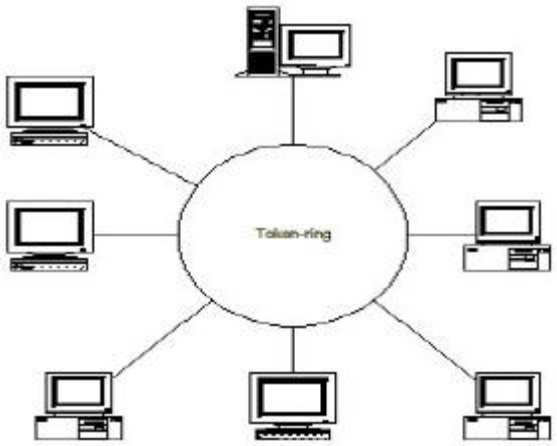
Kelemahan: - Deteksi dan isolasi kesalahan sangat kecil.

- Kepadatan lalu lintas.

- Bila salah satu *client* rusak maka jaringan tidak dapat berfungsi.
- Diperlukan *repeater* untuk jarak jauh

2. Topologi cincin (*ring*)

Topologi cincin adalah topologi jaringan di mana setiap titik terkoneksi ke titik lainnya membentuk jaringan melingkar yang menyerupai cincin. Pada topologi ring, setiap *node* mempunyai tingkatan yang sama. Jaringan akan disebut sebagai *loop*, data dikirimkan ke setiap *node* dan setiap informasi yang diterima *node* diperiksa alamatnya apakah data itu untuknya atau bukan.



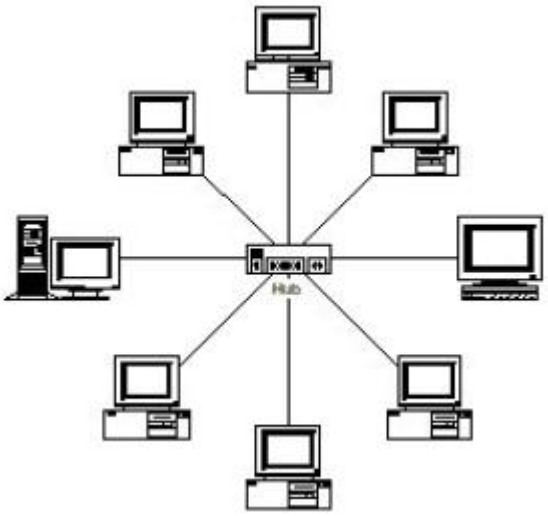

Gambar 2.5 skema topologi *ring*

Kelebihan: - Hemat kabel

Kelemahan: - Komunikasi dapat terganggu jika satu *node* mengalami gangguan.

3. Topologi bintang (*star*)

Topologi bintang merupakan bentuk topologi jaringan yang berupa konvergensi dari node tengah ke setiap *node* atau pengguna. Topologi jaringan bintang termasuk topologi jaringan dengan biaya menengah.




Gambar 2.6 skema topologi *star*

Kelebihan: - Pemasangan/perubahan stasiun sangat mudah dan tidak mengganggu bagian jaringan lain.

- Tahan terhadap lalu lintas jaringan yang sibuk.
- Kemudahan deteksi dan isolasi kesalahan/kerusakan
- Kemudahaan pengelolaan jaringan

- Kerusakan pada satu saluran hanya akan mempengaruhi jaringan pada saluran tersebut dan *station* yang terpaut.

Kelemahan: - Boros kabel

- Perlu penanganan khusus
- Jika *node* tengah mengalami kerusakan, maka seluruh jaringan akan terhenti.

## 2.3 Pewarnaan Graf (*Graph Coloring*)

### 2.3.1 Pengertian Pewarnaan Graf (*Graph Coloring*)

Pengertian Graf menurut Kamus Istilah Teknologi Informasi adalah himpunan simpul yang dihubungkan oleh sisi-sisi. Sedangkan secara matematis, sebuah graf G didefinisikan sebagai pasangan himpunan (V,E), yang dalam hal ini:

V = himpunan tidak kosong dari simpul-simpul (*vertex*) = $\{v_1,v_2,...,v_n\}$

E = himpunan sisi (*edge*) yang menghubungkan sepasang simpul = $\{e_1,e_2,\ldots,e_n\}$

Atau dapat ditulis singkat notasi G = (V,E)

Kegunaan graf sangat banyak. Umumnya graf digunakan untuk memodelkan suatu masalah sehingga menjadi lebih mudah, yaitu dengan cara merepresentasikan objek-objek diskrit dan hubungan antara objek-objek tersebut. Contoh pemodelan suatu masalah dengan menggunakan graf dapat dilihat pada penggambaran rangkaian listrik, senyawa kimia, jaringan komunikasi, jaringan *network computer,* analisis algoritma, peta, struktur hierarki sosial, dan lain-lain. Salah satu topik menarik pada graf adalah masalah pewarnaan graf (*graph coloring).*

Pengertian pewarnaan graf (*graph coloring)* adalah pemberian warna, yang biasanya direpresentasikan sebagai bilangan terurut mulai dari 1, pada objek tertentu pada graf. Objek tersebut dapat berupa simpul, sisi,wilayah ataupun kombinasi ketiganya.

### 2.3.2 Klasifikasi Pewarnaan Graf

Ada tiga macam pewarnaan graf, yaitu:

- Pewarnaan simpul (*vertex coloring*)

  Merupakan pemberian warna atau label pada setiap simpul sehingga tidak ada 2 simpul bertetangga yang memiliki warna sama.

- Pewarnaan sisi (*edge coloring*)

  Merupakan pemberian warna pada setiap sisi pada graf sehingga sisi-sisi yang berhubungan tidak memiliki warna yang sama.

- Pewarnaan wilayah (*region coloring*)

  Merupakan pemberian warna pada setiap wilayah pada graf sehingga tidak ada wilayah yang bersebelahan yang memiliki warna yang sama.

### *2.4 Distributed Coloring Algorithm*

Keamanan dalam sebuah jaringan bisa dicapai jika berbagai macam perangkat lunak dimanfaatkan. Dengan memanfaatkan berbagai macam sistem perangkat lunak, serangan terhadap sistem bisa diminimalkan. Dalam pembahasan ini, kita menunjukan bahwa menggunakan sistem perangkat lunak secara acak tidak cukup untuk meningkatkan keanekaragaman dalam sebuah jaringan. Disini akan ditunjukkan bahwa, dengan mendistribusikan perangkat lunak ke dalam sebuah jaringan dapat meningkatkan keefektifan dari keanekaragaman perangkat lunak untuk memperlambat atau mencegah datangnya worm atau serangan dari hacker. Terdapat beberapa algoritma distribusi dimana melalui pengenalan sistematik ke dalam sebuah jaringan akan mengurangi kemampuan sebuah serangan untuk berpindah dari satu sistem ke sistem yang lain.

Dengan begini, hacker tidak akan mampu menyerang seluruh sistem di dalam jaringan karena di dalam jaringan tersebut terdapat berbagai macam sistem. Sistem-sistem lain yang ada didalam jaringan tetap dapat diserang, namun masing-masing sistem memiliki sistem ketahanan yang berbeda-beda.

Algoritma-algoritma distribusi tersebut diurutkan berdasarkan kompleksitasnya. *The Randomized Coloring algorithm* mengharuskan masing-masing node untuk memilih sebuah warna secara random dan tidak menggantinya sepanjang durasi dari operasi jaringan tersebut. Algoritma yang kedua mengijinkan sebuah node untuk mengevaluasi lingkungan lokalnya, kemudian mengganti warnyanya jika jumlah tetangga dengan warna yang sama sangat besar. Algoritma ini disebut *Color Flipping Algorithm*. algoritma berikutnya dinamakan *Color Swapping Algorithm.* Algoritma ini mengijinkan sepasang node untuk saling bertukar warna pada interval tertentu untuk mengurangi jumlah sisi yang rusak. Terakhir, *Hybrid Algorithm.* Algoritma ini menggabungkan algoritma *color flipping* dengan *color swapping strategy.*

### *2.4.1 Randomize Coloring Algorithm*

Algoritma yang pertama dan paling dasar adalah *Randomized Coloring Algorithm.* Algoritma ini memberikan **m/k** sisi rusak secara rata-rata, dimana **m** menyatakan jumlah sisi yang rusak yang terdapat dalam jaringan dan **k**

menyatakan jumlah node yang terdapat dalam jaringan. Untuk membuktikan hal ini: setelah setiap node memilih sebuah warna secara acak, kemudian pilih satu sisi. Probabilitas bahwa kedua node disisi itu memiliki warna yang sama adalah **1/k**. Total dari semua sisi, jumlah rata-rata dari sisi yang rusak adalah **m/k**. Algoritma ini memerlukan O(1) kompleksitas untuk setiap node memilih warna secara acak, dan tidak ada komunikasi antara node. Karena tidak ada komunikasi antar node, algoritma ini dapat dianggap aman terhadap serangan.

Pewarnaan graph yang dihasilkan dari algoritma ini, sub-optimal. Dalam keadaan terburuk, algoritma ini dijalankan dengan jelek. Randomize algorithm dapat menyebabkan setiap *link* membentuk hubungan antara dua sistem yang sama. Walaupun probabilitas hal ini terjadi adalah $\mathbf{(1/k)^{n-1}}$, hal ini mempunyai pengaruh yang signifikan terhadap keamanan jaringan.

### *2.4.2 Color Flipping Algorithm*

Dalam algoritma ini, setiap node mewarnai diri mereka sendiri dengan menggunakan *Randomized Algorithm*. Setelah interval tertentu, setiap node mengevaluasi node-node yang bertetanggaan dengannya untuk menentukan apakah mengganti warna dapat mengurangi jumlah sisi yang rusak. Karena setiap node perlu mengevaluasi node tetangganya, waktu yang diperlukan untuk

menjalankan algoritma ini adalah O(Δ(G)). Δ(G) adalah jumlah maksimum dari graph tersebut. Setelah data-data node tetangga terkumpul, O(Δ(G)+k) operasi harus dilaksanakan untuk menghasilkan sensus dari warna-warna lokal dan menentukan warna yang menjadi minoritas. Jika mengganti warna node dengan warna minoritas dapat mengurangi jumlah sisi yang rusak sampai dibawah **d(v)/k**, maka penggantian warna dilaksanakan, dimana **d(v)** menyatakan jumlah sisi yang mempunyai warna node yang sama. Setiap penggantian warna, jumlah sisi yang rusak berkurang setidaknya satu sisi. Jumlah sisi yang ada di dalam graph adalah **m**. Maksimum penggantian warna yang bisa dilakukan adalah sebanyak **m**. Setelah algoritma ini selesai dieksekusi, jumlah sisi yang rusak akan berkurang dibawah jumlah rata-rata sisi yang rusak pada algoritma *randomized coloring*.

**Teori**: batas atas dari jumlah sisi yang rusak yang dihasilkan dari *color flipping algorithm* tidak lebih dari jumlah rata-rata dari sisi rusak yang dihasilkan dari *randomized algorithm*.

**Bukti**: Pada saat konvergen (selesai), setiap node terhubung maksimum $\lfloor d(v)/k \rfloor$ sisi yang rusak. Jumlah kedua sisi dari sisi yang rusak adalah $\sum_v \lfloor d(v)/k \rfloor$. Jadi, jumlah dari sisi yang rusak adalah $1/2 \sum_v \lfloor d(v)/k \rfloor$. Sebagai perbandingan dengan *randomized algorithm*:

$$\frac{1}{2}\sum_{v}\left\lfloor\frac{d(v)}{k}\right\rfloor \leq \frac{1}{2}\sum_{v}\frac{d(v)}{k} = \frac{m}{k}$$

### *2.4.3 Color Swapping Algorithm*

Algoritma ini merupakan perpanjangan dari *Kernighan-Lin heuristic* untuk menghitung *balanced cuts*. Pada algoritma ini, masing-masing node berusaha untuk mengurangi jumlah sisi yang rusak dengan bernegosiasi untuk menukar warna dengan tetangganya. Setelah mengumpulkan sisi-sisi yang rusak yang akan dibuang dari node tetangga dan diri sendiri, node yang mengesekusi algoritma ini akan memilih tetangga yang dilihat paling optimal dan mengajukan pertukaran warna. Jika node tetangga bersedia menukar warna, maka pertukaran warna akan dilaksanakan.

Untuk terjadinya pertukaran warna pada algoritma pertama yang disebut sebagai *Mutually beneficial swapping* (pertukaran yang saling menguntungkan), pertukaran warna harus mengurangi jumlah sisi yang rusak pada kedua node yang bertukar warna. Algoritma kedua, dinamakan sebagai *greater good swapping,* akan menyebabkan pertukaran jika total jumlah sisi yang rusak antara kedua node tersebut berkurang setelah pertukaran. Jumlah node yang tersedia untuk *greater good swapping* berarti kualitas dari solusi yang dihasilkan diharapkan lebih baik dari *mutually beneficial swapping algorithm.* Namun,

meningkatnya jumlah partner yang digunakan untuk bertukar warna, juga akan meningkatkan kemungkinan algoritma itu untuk diserang.

### *2.4.4 Hybrid Algorithm*

Algoritma ini merupakan gabungan antara algoritma *color flipping* dan *color swapping. Randomized hybrid* algoritma adalah algortima yang mengijinkan node yang ingin mengubah warnanya, untuk memilih melakukan *color flipping* atau *color swapping* secara acak.

Algoritma yang kedua adalah *best choice hybrid algorithm* (pilihan terbaik algoritma hybrid). Algoritma ini mengijinkan sepasang node unttuk mengevaluasi jumlah sisi rusak yang dapat dikurangi dengan melakukan pertukaran warna antara pasangan itu atau masing-masing melakukan *color flip.* Jika masing-masing node dapat mengurangi sisi rusak lebih banyak daripada masing-masing node melakukan *color flip,* maka *color swap* akan dilaksanakan. Jika tidak, maka kedua node itu akan melakukan *color flip* masing-masing. Jika node yang ingin mengubah warnanya mengetahui bahwa *color swapping* tidak dapat dilakukan atau menguntungkan, maka dia akan melakukan *color flip* sendiri.

## 2.5 Permutasi, Kombinasi dan Peluang

### 2.5.1 Definisi Permutasi

Permutasi adalah suatu susunan yang berbeda yang dibentuk oleh sebagian atau keseluruhan unsur yang diambil dari sekelompok unsur yang tersedia. Banyaknya permutasi-r dari n unsur yang berbeda adalah:

$P(n,r) = n!/(n-r)!$

**Bukti:**

Asumsikan bahwa permutasi-r dari n unsur yang berbeda merupakan aktivitas yang terdiri dari r langkah yang berurutan. Langkah pertama adalah memilih unsur pertama yang bisa dilakukan dengan n cara. Langkah kedua adalah memilih unsur kedua yang bisa dilakukan dengan n-1 cara karena unsur pertama sudah terpilih. Lanjutkan langkah tersebut sampai pada langkah ke-r yang bisa dilakukan dengan n- r + 1 cara. Berdasarkan Prinsip Perkalian, diperoleh:

$$n(n-1)(n-2)\ldots(n-r+1) = \frac{n(n-1)(n-2)\ldots 2.1}{(n-r)(n-r-1)\ldots 2.1} = \frac{n!}{(n-r)!}$$

Jadi, $P(n, r) = \frac{n!}{(n-r)!}$

### 2.5.2 Definisi Kombinasi

Kombinasi adalah himpunan sekelompok unsur atau objek tanpa menghiraukan susunannya atau urutannya. Banyaknya kombinasi *r* dari sebuah himpunan berisi *n* elemen dapat dihitung tanpa harus memperhatikan isi dari himpunan tersebut. Besarnya dinyatakan dengan fungsi:

$$C_r^n = \frac{n!}{r!(n-r)!}$$

Fungsi $C_r^n$ dalam banyak literatur dinyatakan juga dengan notasi $\binom{n}{r}$.

Yang dapat dengan mudah dibuktikan:

$$C_r^n = \frac{P_r^n}{r!}$$

$$= \frac{\frac{n!}{(n-r)!}}{r!}$$

$$= \frac{n!}{r!(n-r)!}$$

Sebagai contoh, tanpa harus mengetahui elemen himpunan {*apel, jeruk, mangga, pisang*}, banyaknya kombinasi 3 dari himpunan tersebut dapat dihitung:

$$C_3^4 = \frac{4!}{3!(4-3)!} = 4$$

Dalam skripsi ini, rumus kombinasi digunakan untuk mencari jumlah total sisi dari jumlah komputer (n) yang ditentukan, yaitu dengan rumus C(n,2).

### 2.5.3 Definisi Peluang

Besarnya kemungkinan terjadinya sebuah kejadian disebut peluang kejadian atau probabilitas kejadian. Penentuan nilai peluang kejadian didasarkan kepada banyak anggota kejadian dan banyak anggota ruang sampelnya. Ruang sampel adalah himpunan semua hasil suatu percobaan. Ruang sampel umumnya dinotasikan dengan S. setiap anggota ruang sampel disebut titik sampel. Kejadian adalah himpunan bagian dari ruang sampel. Misalnya dalam pemutaran sebuah mata uang, kejadian yang mungkin muncul adalah sisi angka dan sisi gambar. Banyak anggota ruang sampel adalah 2. Ini berarti peluang munculnya sisi angka sama dengan ½ dan peliuang munculnya sisi gambar sama dengan ½.

Missal dalam suatu percobaan setiap hasil mempunyai kemungkinan yang sama untuk terjadi. Jika banyak anggota kejadian K = n(K) dan banyak anggota ruang sampelnya = n(S) maka peluang terjadinya kejadian K adalah P(K)= n(K)/n(S).

Karena kejadian merupakan himpunan bagian dari ruang sampel maka banyaknya anggota kejadian lebih kecil atau sama dengan banyak anggota ruang sampel. $0 \leq n(K) \leq n(S)$. karena peluang kejadian K adalah $P(K)= n(K)/n(S)$ maka $0 \leq P(K) \leq 1$.

## 2.6 Pengenalan Bahasa Pemrograman Java

Bahasa pemrograman Java lahir karena ketidakpuasan seorang insinyur di SUN Micro System bernama James Gosling. Ia tidak puas dengan kompiler C++ karena dinilai terlalu banyak menghasilkan *bug*, berbiaya besar, sangat bergantung pada *platform,* gosling merasa perlu membuat kompiler baru sebagai solusi terhadap sejumlah kelemahan pada C++ tersebut.

Kompiler baru tersebut diberi nama dengan Oak. Kompiler ini mirip dengan C++ tetapi dengan sejumlah pengurangan fitur yang dianggap kurang menguntungkan dalam pengembangan, seperti *multiple inheritance*, konversi tipe secara otomatis, penggunaan *pointer* dan managemen memori.

Pada tahun 1994, Oak diubah namanya menjadi Java. Pada era ini, Java divisikan sebagai bahasa yang memiliki dukungan baik terhadap web.

Java adalah bahasa pemrograman berorientasi objek dan bebas *platform* yang dikembangkan oleh SUN Micro System dengan sejumlah keunggulan.

Keunggulan-keunggulan dari Java adalah:

- Berbasis GUI

  Dengan Java, kita bisa membuat tampilan berbasis grafik untuk memudahkan pemakai berinteraksi dengan program.

- Berorientasi objek

  Java merupakan salah satu bahasa yang memiliki dukungan penuh terhadap konsep pemrograman berorientasi objek.

- Aplikasi web

  Java merupakan bahasa pemrograman yang memiliki dukungan sangat baik terhadap aplikasi web. Hal ini wajar sebab memang pada awalnya Java dilahirkan sebagai solusi untuk menjawab kebutuhan bahasa pengembangan yang mendukung aplikasi berbasis jaringan. Beberapa teknologi Java yang mendukung aplikasi web adalah Applet, JSP, CORBA, dan lain-lain.

- *Multiplatform*

  Dengan Java, sekali program dibuat maka program tersebut bisa dijalankan pada komputer dengan *platform* yang berbeda, asalkan JVM (*Java Virtual Machine*) telah ter*install* untuk *platform* itu. Jika membuat program Java di atas Windows, maka ia bisa dijalankan pada Linux, Unix, Macintosh.

- Keamanan

Fitur keamanan yang terdapat pada Java adalah *signature* (untuk menandatangani dokumen), *message digest*, pembangkitan kunci, autentikasi, enkripsi, dan bilangan besar.

- *Distributed Networking Application*

  Aplikasi yang berjalan pada jaringan terdistribusi melibatkan sejumlah komputer yang berkomunikasi secara transparan, yaitu seolah sejumlah komputer itu merupakan sebuah komputer yang bersatu untuk menjalankan sebuah perintah. Dengan fitur ini anda bisa membuat program untuk menyelesaikan kasus tertentu dengan melibatkan beberapa komputer. Ini akan membuat penyelesaian masalah lebih cepat.

- Mendukung *Software Mission-Critical*

  Sebagai dukungan terhadap *software* ini, Java menghilangkan sejumlah fitur C++ yang berpotensi menghasilkan *error* yang fatal, seperti *pointer,* konversi tipe tanpa dicek, dan lain-lain.

- *Multithread*

  Fitur *multithread* digunakan untuk menjalankan sejumlah proses secara bersamaan. Dengan menggunakan fasilitas ini kita bisa membuat program Java yang menjalankan beberapa perintah sekaligus, sehingga tidak perlu menunggu sebuah perintah selesai dikerjakan untuk mengerjakan perintah lain.

Bahasa Java saat ini sedang berada dalam masa pertumbuhannya. Dalam pertumbuhannya saat ini, Java telah merambah berbagai kebutuhan IT. Hampir semua yang kita butuhkan untuk membuat *software* canggih telah difasilitasi oleh Java.

Java merupakan bahasa yang bisa dipakai dengan gratis. Walaupun gratis, kualitas program yang dihasilkan bisa dikatakan belum ada tandingannya saat ini. Seperti yang telah dijelaskan di atas, dalam segala aspek, Java memiliki keunggulan yang memungkinkannya untuk digunakan pada pengembangan *software- software* enterprise.