

SK5002 ALGORITHM AND SOFTWARE DESIGN

Catatan Kuliah
Sebuah Usaha untuk Mencatat

Mohammad Rizka Fadhli (Ikang)
20921004@mahasiswa.itb.ac.id

07 October 2021

Contents

1	SILABUS	5
1.1	Keterangan Umum	5
1.1.1	Silabus Ringkas	5
1.1.2	Silabus Lengkap	5
1.1.3	<i>Outcomes</i>	5
1.1.4	Panduan Penilaian	6
1.2	Satuan Acara Perkuliahan	7
1.3	Info Lainnya	7
2	PERTEMUAN MINGGU I	8
2.1	<i>Computational Thinking</i>	8
2.2	<i>Algorithm</i>	9
	Contoh	9
3	PERTEMUAN MINGGU II	14
3.1	<i>Algorithm Design</i>	14
3.2	Algoritma Sederhana	14
3.2.1	<i>Case Study I</i>	14
3.3	Quiz	16
	Soal 1	16
	Soal 2	17
	Soal 3	18
4	PERTEMUAN MINGGU III	22
4.1	<i>Numerical Method</i>	22
4.1.1	Mencari Akar Persamaan	22
4.1.2	Luas Area di Bawah Kurva	25
4.1.3	Solusi Persamaan Diferensial	27

5 PERTEMUAN MINGGU IV	32
5.1 <i>Algorithm Design: Randomness</i>	32
5.1.1 Simulasi Monte Carlo	32
5.1.2 <i>Flowchart</i> Simulasi Monte Carlo	32
5.1.3 Modifikasi Simulasi Monte Carlo	33
5.1.4 <i>Flowchart</i> Modifikasi Simulasi Monte Carlo	33
6 PERTEMUAN MINGGU V	34
6.1 <i>Shortest Path Problems</i>	34
6.1.1 <i>Breadth-First Search</i>	34
6.1.2 <i>Bellman-Ford</i>	34
6.1.3 Djikstra	38
6.1.4 <i>A-star</i> A* Algorithm	38
7 PERTEMUAN MINGGU VI	39
7.1 <i>Genetic Algorithm</i>	39
7.1.1 <i>Advantages vs Disadvantages</i>	39
7.1.2 <i>Basic Terminology</i>	39
7.1.3 <i>Basic Structure</i>	40
7.1.4 Contoh Soal	40
EPILOG	43

List of Figures

1	Bobot Penilaian	6
2	Satuan Acara Perkuliahan	7
3	Computational Thinking	8
4	Flowchart	9
5	Contoh Pseudo Code	10
6	Contoh Flowchart	10
7	Flowchart Faktorial	19
8	Flowchart Brute Force	32
9	Flowchart Modifikasi Monte Carlo	33
10	Graf Soal I	34
11	Graf Soal I	37
12	Basic Terminology	39
13	GA Flow	40

1 SILABUS

1.1 Keterangan Umum

1.1.1 Silabus Ringkas

Pada mata kuliah ini, mahasiswa akan mempelajari dan membahas tentang teknik-teknik pengembangan algoritma yang banyak digunakan, optimasi dan perancangan program berbasis data.

1.1.2 Silabus Lengkap

1.1.2.1 Pendahuluan Ruang lingkup perkuliahan:

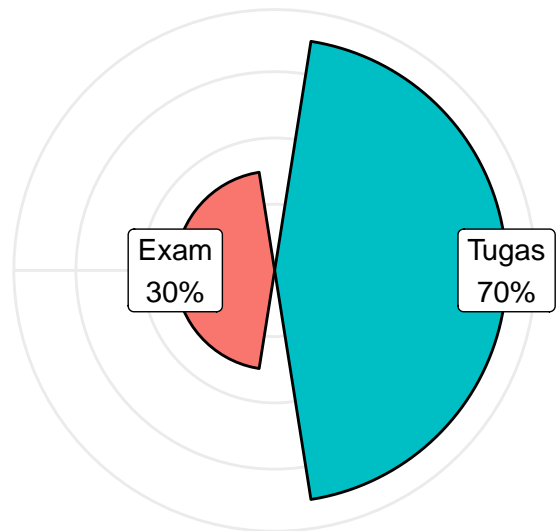
- Topik-topik terkait dengan perkuliahan, pengantar algoritma;
- Algoritma pemrograman:
 - Pengembangan pola berpikir,
 - Kinerja program,
 - Algoritma dasar,
 - *Flow-chart*,
 - Fungsi,
 - Pola pemrograman algoritma rekursif.
- Optimasi Algoritma:
 - *Bellman Algorithm*,
 - Pencarian jalan terdekat, pengukuran efisiensi, Dijkstra Algorithm.
- Aplikasi *Dijkstra*, *Genetic algorithm*, *Holland theory*, aplikasi GA;
- Perancangan program terstruktur:
 - Program *design*, *water fall*, *quick*, *proto typing*, pengertian *system* dan aliran data.
- Rancangan *context diagram*
- Data *Flow* diagram, basis data, *ERD*.
- Perancangan berbasis object *Unified Model Language* (UML), aktor, diagram *use-case*, pendefinisian masalah, *activation diagram* dalam UML, *break down* problem dalam *use-case* dan *activity diagram*, *interaction diagram*, *class diagram*, *swim lane*, *collaboration diagram*.

1.1.3 Outcomes

1. Mahasiswa akan memahami dan menguasai teknik-teknik pengembangan algoritma dan perancangang program yang banyak digunakan.
2. Mahasiswa akan dapat mengembangkan dan menulis algoritma suatu karya progam dengan baik dan benar menggunakan metoda yang beragam.
3. Mahasiswa akan memiliki pengalaman perancangan suatu system dan program.

1.1.4 Panduan Penilaian

Porsi Penilaian Awal Mata Kuliah
Pemrograman dalam Sains



Dibuat dengan R
ikanx101.com

Figure 1: Bobot Penilaian

1.2 Satuan Acara Perkuliahan

Algorithm and Software Design (SK5002)			
Minggu	Topik	Jumlah tugas	Jenis tugas
1	Pendahuluan, ruang lingkup perkuliahan, topik-topik terkait dengan perkuliahan,		
2-4	Algoritma pemrograman	2-3	Individu
5-7	Optimasi algoritma	2-3	Individu/kelompok
8	UTS (Wajib)		
9-10	Perancangan program tersuktur	1	Kelompok
11-14	Program berbasis objek	1	Kelompok

Assessment <ol style="list-style-type: none"> 1. Tugas individu (homeworks) 2. Problem based assessment 3. Final project 	Referensi <ol style="list-style-type: none"> 1. William Kocay, Donald L. Kreher, Graphs, Algorithms, and Optimization, 2, CRC Press, 2016 2. Ronald J. Leach, Introduction to Software Engineering, 2, CRC Press, 2016
--	---

Figure 2: Satuan Acara Perkuliahan

1.3 Info Lainnya

UTS akan dilaksanakan kira-kira pada minggu ke-8. Setelah itu baru masuk ke topik perancangan perangkat lunak (baru di sini akan ada kerja kelompok). *Ending*-nya presentasi per kelompok.

Semua dikumpulkan via *Ms. Teams*.

2 PERTEMUAN MINGGU I

26 Agustus 2021

2.1 *Computational Thinking*

Ada empat pilar:

1. Dekomposisi.
 - Memecah masalah besar ke masalah-masalah yang lebih kecil sehingga lebih bisa *di-manage*.
2. *Pattern recognition*.
 - Menganalisa dan melihat apakah ada pola atau pengulangan.
3. *Algorithm design*
 - Menuliskan langkah-langkah dalam bentuk formal.
4. *Abstraction*
 - Memisahkan mana yang *important*, mana yang *less important*.

What is computational thinking?

Computational thinking is the thought processes involved in formulating a problem and expressing its solution(s) in such a way that a computer —human or machine— can effectively carry out.

Fahdzi Muttaiqien, S.Si., M.Si., M.Sc.,

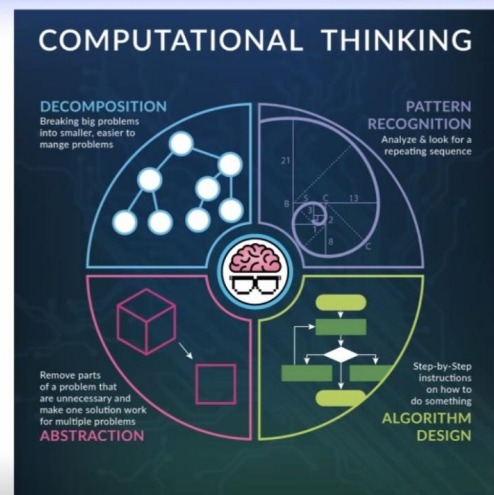


Figure 3: Computational Thinking

2.2 Algorithm

A set of procedure (step by step) to solve a (sub)problem.

Bentuknya bisa:

- *Pseudocode*
- *Flowchart*

















Algorithm Design → Flowchart		 Flow Direction symbol Yaitu simbol yang digunakan untuk menghubungkan antara simbol yang satu dengan simbol yang lain. Simbol ini disebut juga connecting line.	 Simbol Manual Input Simbol untuk pemasukan data secara manual on-line keyboard
Simbol-simbol dalam penulisan flowchart		 Terminator Symbol Yaitu simbol untuk permulaan (start) atau akhir (stop) dari suatu kegiatan	 Simbol Preparation Simbol untuk mempersiapkan penyimpanan yang akan digunakan sebagai tempat pengolahan di dalam storage.
		 Connector Symbol Yaitu simbol untuk keluar - masuk atau penyambungan proses dalam lembar / halaman yang sama.	 Simbol Predefine Proses Simbol untuk pelaksanaan suatu bagian (sub-program)/prosedure
		 Connector Symbol Yaitu simbol untuk keluar - masuk atau penyambungan proses pada lembar / halaman yang berbeda.	 Simbol Display Simbol yang menyatakan peralatan output yang digunakan yaitu layar, plotter, printer dan sebagainya.
		 Processing Symbol Simbol yang menunjukkan pengolahan yang dilakukan oleh komputer	 Simbol disk and On-line Storage Simbol yang menyatakan input yang berasal dari disk atau disimpan ke disk.
		 Simbol Manual Operation Simbol yang menunjukkan pengolahan yang tidak dilakukan oleh komputer	 Simbol magnetik tape Unit Simbol yang menyatakan input berasal dari pita magnetik atau output disimpan ke pita magnetik.
		 Simbol Decision Simbol pemilihan proses berdasarkan kondisi yang ada.	 Simbol Punch Card Simbol yang menyatakan bahwa input berasal dari kartu atau output ditulis ke kartu
		 Simbol Input-Output Simbol yang menyatakan proses input dan output tanpa tergantung	 Simbol Dokumen Simbol yang menyatakan input berasal dari dokumen dalam bentuk kertas atau output

Figure 4: Flowchart

Algoritma biasanya memiliki minimal 3 *control structure*, yakni:

1. *Sequence*
2. *Conditional*
3. *Repetition / loop*

Contoh

Bagaimana cara mengurutkan bilangan berikut ini:

5, 8, 6, 1, 3

Pseudocode Jawaban standarnya seperti ini:

```

Step 1: Start
Step 2: Read the array of given items from the user.
Step 3: Take the first element(index = 0), compare the
current element with the next element.
Step 4: If the current element is greater than the next
element, swap them.
Step 5: Else, If the current element is less than the next
element, then move to the next element.
Step 6: Repeat Step 3 to Step 5 until all elements are
sorted.
Step 7: Stop

```

Figure 5: Contoh Pseudo Code

Flowchart Berikut adalah bentuk *flowchart*-nya:

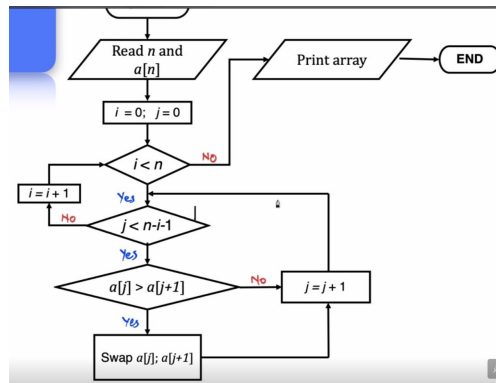


Figure 6: Contoh Flowchart

Jawaban Sendiri Versi 1¹ Kalau dengan jawaban sendiri, *pseudocode*-nya adalah sebagai berikut:

- **Step 1:** *INPUT vector* berupa numerik yang tidak berurut, misal dinotasikan sebagai a dengan n buah elemen.
- **Step 2:** Akan saya buat *vector* b dengan panjang n elemen.
- **Step 3:** b_1 akan saya isi dengan angka terendah dari a . Setelah dipindahkan, a kini memiliki elemen sebanyak $n - 1$.
- **Step 4:** Ulang **step 3** hingga semua elemen a dipindahkan ke b .

```
rm(list=ls())

soal = c(5,8,6,1,3,1,2,-5,2,2,-4)

urut_donk = function(input){
  # set initial condition
  a = input
  n = length(input)

  # siapkan vector kosong
  b = c()

  # repetition dengan syarat a masih ada
  # dilihat dari panjang n yang masih > 0
  while(n > 0){
    # mengambil elemen terkecil dari a
    min = min(input)
    n_min = length(which(input == min))
    # memindahkan elemen tersebut ke dalam b
    b = c(b,
          rep(min,n_min)
          )
    # menghapus elemen a yang sudah dipindahkan
    input = input[!input %in% b]
    n = length(input)
  }

  output = list(`Data asli` = a,
                `Hasil terurut` = b)
  return(output)
}

urut_donk(soal)
```

¹<https://ikanx101.com/blog/sort-number/>

```
## `$Data asli`
## [1] 5 8 6 1 3 1 2 -5 2 2 -4
##
## `$Hasil terurut`
## [1] -5 -4 1 1 2 2 2 3 5 6 8
```

Jawaban Sendiri Versi 2² Bagaimana jika kita tidak boleh menggunakan fungsi lain selain *looping*, *conditional*, dan `length()`?

Caranya hampir mirip dengan *flowchart* di atas, yakni:

1. *Input vector* yang hendak diurutkan, misal notasi a .
 - Hitung ada berapa *elemen* yang ada, misal notasi n .
2. Bandingkan a_1 dengan elemen di sebelahnya.
 - Jika $a_1 > a_2$ maka tukar posisi.
 - Lanjutkan perbandingan a_1 dengan elemen selanjutnya hingga elemen ke- n .
3. Lanjutkan perbandingan untuk elemen kedua dan seterusnya.

```
rm(list=ls())

urut_lagi_donk = function(a){
  # set initial condition
  input = a
  n = length(a)

  # looping
  for(i in 1:(n-1)){
    for(j in (i+1):n){
      temp1 = a[i] # temporary
      temp2 = a[j] # temporary
      if(temp1 > temp2){
        # proses penukaran
        a[i] = temp2
        a[j] = temp1
      }
    }
  }

  # menulis hasilnya
  hasil = list(
    `Data asli` = input,
```

²<https://ikanx101.com/blog/sort-number/>

```
    `Hasil terurut` = a
  )
  # print output
  return(hasil)
}

soal = c(9,8,6,1,3,1,-10)

urut_lagi_donk(soal)
```

```
## $`Data asli`
## [1]  9  8  6  1  3  1 -10
##
## $`Hasil terurut`
## [1] -10  1  1  3  6  8  9
```

3 PERTEMUAN MINGGU II

2 September 2021

3.1 *Algorithm Design*

Algoritma disusun setidaknya memiliki tiga *control structure*, yakni:

1. *Sequence*,
2. *Conditional*,
3. *Repetition* atau *loop*.

Algoritma dibuat sesuai dengan kreativitas masing-masing. Bisa jadi jalan seseorang berbeda dengan orang lainnya. Untuk membandingkan algoritma mana yang terbaik, kita bisa cek dua parameter ini:

1. *Running time* terkecil.
2. *Memory used (RAM)* terkecil.

3.2 Algoritma Sederhana

3.2.1 *Case Study I*

Buatlah algoritma yang bisa mengecek suatu bilangan adalah bilangan prima atau bukan!

```
rm(list=ls())

is_prime = function(n){
  hasil = NA
  temp_1 = 0
  if(n > 1){
    for(i in 2:n){
      temp_2 = n %% i == 0
      temp_2 = temp_1 + temp_2
      hasil = ifelse(temp_2 == 0,
                     "Prime",
                     "Not Prime")
    }
  } else{
    hasil = "Bilangan < 1"
  }
  return(hasil)
}
```

```
}  
  
is_prime(4)
```

```
## [1] "Not Prime"
```

3.3 Quiz

Soal 1

Construct Pascal's triangle!

```
rm(list=ls())

# ide dasarnya adalah hanya dengan menggeser baris atas ke bawah
# lalu geser ke kiri
# lalu geser ke kanan

pascal = function(n){
  # initial
  x = 1
  print(x)
  for(i in 2:n){
    x = c(0,x) + c(x,0)
    print(x)
  }
}

# kita uji coba ya
pascal(15)
```

```
## [1] 1
## [1] 1 1
## [1] 1 2 1
## [1] 1 3 3 1
## [1] 1 4 6 4 1
## [1] 1 5 10 10 5 1
## [1] 1 6 15 20 15 6 1
## [1] 1 7 21 35 35 21 7 1
## [1] 1 8 28 56 70 56 28 8 1
## [1] 1 9 36 84 126 126 84 36 9 1
## [1] 1 10 45 120 210 252 210 120 45 10 1
## [1] 1 11 55 165 330 462 462 330 165 55 11 1
## [1] 1 12 66 220 495 792 924 792 495 220 66 12 1
## [1] 1 13 78 286 715 1287 1716 1716 1287 715 286 78 13 1
## [1] 1 14 91 364 1001 2002 3003 3432 3003 2002 1001 364 91 14 1
```


Soal 2

Find n -th Fibonacci number: $F_n = F_{n-1} + F_{n-2}$, $F_0 = 0$, $F_1 = 1$

```
rm(list=ls())

# set initial condition
f = c(0,1)

# karena basis di R dimulai dari 1 sedangkan Fibonacci dimulai dari 0
# maka nanti indexing akan dibuat kurang 1
# proses deduksi
n = 10
for(i in 2:n){
  f[i+1] = f[i] + f[i-1]
}

# berikut adalah hasilnya:
f
```

```
## [1] 0 1 1 2 3 5 8 13 21 34 55
```

Atas dasar ini kita akan buat *function*-nya sebagai berikut:

```
fibo = function(n){
  f = c(0,1)
  for(i in 2:n){
    f[i+1] = f[i] + f[i-1]
  }
  return(list(`Fibonacci Seq` = f,
              `nth Fibo Number` = max(f))
  )
}

# saatnya uji coba
fibo(10)
```

```
## $`Fibonacci Seq`
## [1] 0 1 1 2 3 5 8 13 21 34 55
##
## $`nth Fibo Number`
## [1] 55
```

Soal 3

Bikin algoritma faktorial!

Untuk memudahkan pembuatan algoritma, kita akan melihat kembali definisi dari faktorial³ sebagai berikut:

$$n! = (n)(n-1)(n-2)..(1)$$

Dengan syarat $n \geq 0$ dan n berupa *integer*. Namun perlu diperhatikan bahwa $0! = 1$.

Oleh karena itu, kita bisa menggunakan prinsip rekursif dengan algoritma dalam *pseudocode* berikut ini:

Algoritma dalam *Pseudocode*

```
INPUT n

IF n NOT INTEGER OR n < 0 STOP

IF n = 0 OR n = 1 RETURN 1

ELSE

  DEFINE a = 1

  FOR i 2:n
    a = a*i
  RETURN a
```

Bentuk *flowchart* dari *pseudocode* di atas adalah sebagai berikut:

³<https://id.wikipedia.org/wiki/Faktorial>

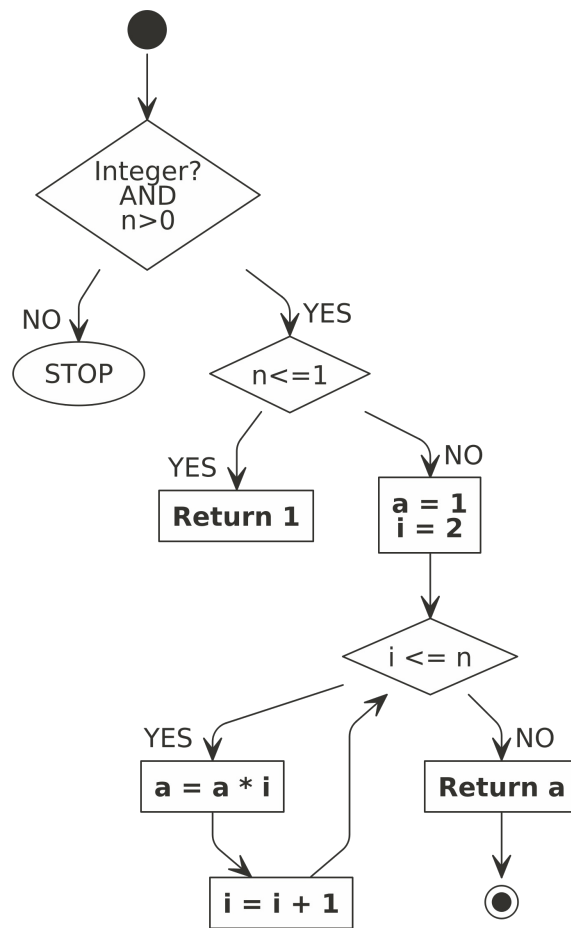


Figure 7: Flowchart Faktorial

Algoritma dalam *Flowchart*

R function Sekarang algoritma di atas jika dibuat **R function**-nya adalah sebagai berikut:

```
f_torial = function(n){
  # initial definition
  hasil = 1

  # conditional
  if(n < 0){hasil = "n yang dimasukkan < 0"}
  else if(n %in% c(0,1)){hasil = 1}
  else{
    for(i in 2:n){
      hasil = hasil*i
    }
  }

  # return output perhitungan
  output = list(
    `Input angka` = n,
    `n!` = hasil
  )

  # print output
  return(output)
}
```

Mari kita cek hasilnya dalam berbagai kondisi berikut:

```
f_torial(-2)
```

```
## $`Input angka`
## [1] -2
##
## $`n!`
## [1] "n yang dimasukkan < 0"
```

```
f_torial(0)
```

```
## $`Input angka`
## [1] 0
##
## $`n!`
## [1] 1
```

```
f_torial(1)
```

```
## $`Input angka`  
## [1] 1  
##  
## $`n!`  
## [1] 1
```

```
f_torial(4)
```

```
## $`Input angka`  
## [1] 4  
##  
## $`n!`  
## [1] 24
```

```
f_torial(7)
```

```
## $`Input angka`  
## [1] 7  
##  
## $`n!`  
## [1] 5040
```

```
f_torial(10)
```

```
## $`Input angka`  
## [1] 10  
##  
## $`n!`  
## [1] 3628800
```

4 PERTEMUAN MINGGU III

9 September 2021

4.1 *Numerical Method*

4.1.1 Mencari Akar Persamaan

Bisection Method Ini mirip dengan mata kuliah **Analisis Numerik Lanjut SK5001**. Dari mata kuliah sebelumnya, saya modifikasi penentuan *error* menjadi $(b - a) < tol_{max}$ sehingga *function*-nya menjadi:

```
rm(list=ls())

bagi_dua = function(a,b,f,iter_max,tol_max){
  # fungsi untuk grafik
  # set nilai x
  baris = seq(a,b,by = .05)
  # nilai y
  y = f(baris)
  # bikin plotnya
  plot =
    data.frame(x = baris,y) %>%
    ggplot(aes(x,y)) +
    geom_line(group = 1,
              color = "darkblue") +
    coord_equal() +
    geom_hline(yintercept = 0)

  # fungsi hitung bisection
  # initial condition
  i = 1
  hasil = data.frame(n_iter = NA,
                     a = NA,
                     b = NA,
                     c = NA)

  while(i<= iter_max && (b-a) > tol_max){
    # cari titik tengahnya
    p = a + ((b-a)/2)
    # hitung fungsi di titik tengah
    FP = f(p)
    # hitung fungsi di titik awal
    FA = f(a)
```

```
# hitung fungsi di titik akhir
FB = f(b)
# tulis hasil dalam data frame
hasil[i,] = list(i,a,b,p)

# tukar nilai a atau b dengan nilai p
if(FA*FP < 0){b = p} else{a = p}
# untuk iterasi berikutnya
i = i + 1

# mencatat akar persamaan
akar = p
# tambahkan dulu checking apakah f(a), f(b), atau f(c) ada yang nol?
if(FP == 0){
  akar = p
  break} else if(FA == 0){
  akar = a
  break} else if(FB == 0){
  akar = b
  break}

}

# mencatat iterasi terbesar
iterasi = i-1 # dikurang satu karena pada i+1
               # sebenarnya tidak ada proses jika pada while TRUE

# membuat output
hasil = list(
  `plot f(x) di selang [a,b]` = plot,
  `iterasi max` = iterasi,
  `akar persamaan` = akar,
  `hasil perhitungan` = hasil
)

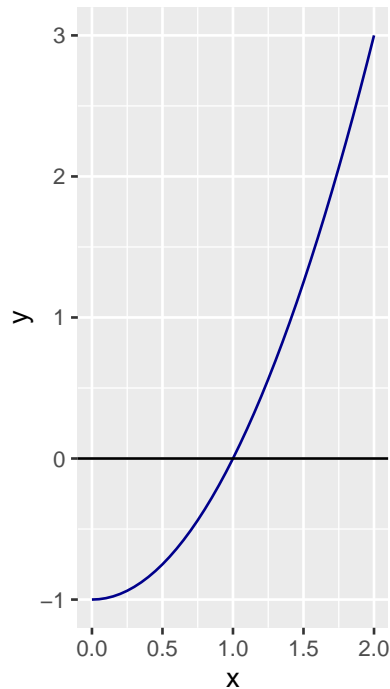
# print output
return(hasil)
}
```

Pada metode ini, pemilihan selang menjadi krusial. Contohnya adalah $f(x) = x^2 - 1$ di selang $[0.5, 3]$.

```
a = 0
b = 2
f = function(x){x^2-1}
iter_max = 50
tol_max = 10^(-7)

bagi_dua(a,b,f,iter_max,tol_max)
```

```
## $`plot f(x) di selang [a,b]`
```



```
##
## $`iterasi max`
## [1] 1
##
## $`akar persamaan`
## [1] 1
##
## $`hasil perhitungan`
##   n_iter a b c
## 1      1 0 2 1
```


Contoh Lainnya $f(x) = x^2 - 3x + 2$ di selang $[1.8, 2.2]$.

4.1.2 Luas Area di Bawah Kurva

Ide dasarnya adalah $L = \text{alas} \cdot \text{tinggi}$, dimana:

1. *alas* bisa didefinisikan sebagai $\Delta x = x_2 - x_1$.
2. Sedangkan definisi *tinggi* tergantung bidang yang dipilih.
 - Jika kita menggunakan *square* di titik tengah Δx , maka $\text{tinggi} = f(\frac{x_1+x_2}{2})$.
 - Jika kita menggunakan *trapezoid*, maka $\text{tinggi} = \frac{f(x_1)+f(x_2)}{2}$.

Algoritmanya:

```
rm(list=ls())

persegi = function(x0,xn,n){
  h = (xn - x0) / n
  integration = f(x0)
  for(i in 1:n){
    k = x0 + i*h
    integration = integration + f(k)
  }
  integration = integration * h
  return(integration)
}

# bikin sendiri
trapezoid = function(x0,xn,n){
  h = (xn - x0) / n
  f0 = f(x0)
  # selang pertama
  i = 1
  k = x0 + i*h
  fn = f(k)
  integration = (f0+fn)/2
  for(i in 2:n){
    f0 = fn
    k = x0 + i*h
    fn = f(k)
    temp = (f0+fn)/2
    integration = integration + temp
  }
  integration = integration * h
}
```

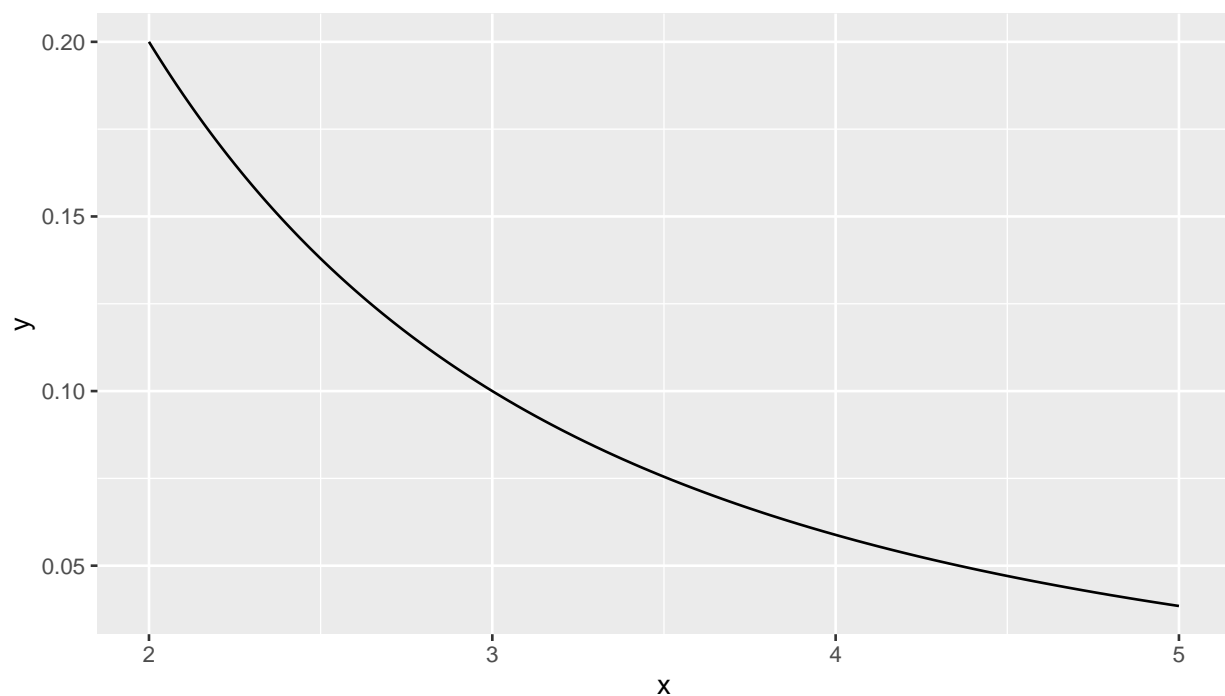
```
    return(integration)
}

# dari dosen
trapezoidal = function(x0,xn,n){
  h = (xn - x0) / n
  integration = f(x0) + f(xn)

  for(i in 1:n){
    k = x0 + i*h
    integration = integration + 2*f(k)
  }
  integration = integration * h/2
  return(integration)
}

f = function(x){1/(1 + x^2)}
a = 2
b = 5

data.frame(x = seq(a,b,by = .01)) %>%
  mutate(y = f(x)) %>%
  ggplot(aes(x,y)) +
  geom_line()
```



```
persegi(2,5,90000)
```

```
## [1] 0.266256
```

```
trapezoid(2,5,90000)
```

```
## [1] 0.266252
```

```
trapezoidal(2,5,90000)
```

```
## [1] 0.2662533
```

4.1.3 Solusi Persamaan Diferensial

$$\frac{dy}{dt} = f(t, y)$$

Harus diketahui minimal 1 *initial condition*, misal: $y(t_0) = y_0$.

Kita akan selesaikan dengan metode **Runge-Kutta** order 4. Bentuk umumnya adalah sebagai berikut:

$$y_{n+1} = y_n + h \sum_{i=1}^n b_i k_i$$

dimana k_i adalah konstanta yang harus dicari.

```
rk_4order = function(f, x0, y0, h, n){
  # initial condition
  x = x0
  y = y0
  # proses iterasi
  for(i in 1:n){
    k1 = f(x0,y0)
    k2 = f(x0 + 0.5*h,y0 + 0.5*k1*h)
    k3 = f(x0 + 0.5*h,y0 + 0.5*k2*h)
    k4 = f(x0 + h,y0 + k3*h)
    y0 = y0 + (1/6)*(k1 + 2*k2 + 2*k3 + k4) * h
    x0 = x0 + h
    x = c(x, x0)
    y = c(y, y0)
  }
  # output
```

```

output = data.frame(x = x,
                    y = y)
return(output)
}

```

4.1.3.1 Contoh Soal *Based on the following differential equation:*

$$\frac{d}{dt}y(t) = t\sqrt{y(t)}; y(0) = 1$$

Define $y(t)$ for $t = 0, 1, 2, \dots, 10$ by using 4th order Runge-Kutta! Use $h=0.1$.

Compare the result with the following exact solution:

$$y(t) = \frac{1}{16}(t^2 + 4)^2$$

Sekarang kita akan hitung bagaimana hasilnya:

```

dydt = function(t,y){t * sqrt(y)}
y0 = 1
t0 = 0
h = 0.1
n = 100

hampiran =
  rk_4order(f = dydt,
            x0 = t0,
            y0 = y0,
            h = h,
            n = n)

```

Lalu akan saya bandingkan hasilnya dari perhitungan *exact* dan menghitung $\Delta = \text{numerik} - \text{exact}$ berikut:

```

y = function(t){(1/16)*(t^2 + 4)^2}
exact =
  data.frame(t = seq(0,10,by = h)) %>%
  mutate(y = y(t))

```

Table 1: Hasil Perhitungan Exact vs Numerik (RK4)

t	y_numerik	y_exact	delta
0.0	1.000000	1.000000	0.00e+00
0.1	1.005006	1.005006	0.00e+00
0.2	1.020100	1.020100	0.00e+00
0.3	1.045506	1.045506	0.00e+00
0.4	1.081600	1.081600	0.00e+00
0.5	1.128906	1.128906	0.00e+00
0.6	1.188100	1.188100	0.00e+00
0.7	1.260006	1.260006	-1.00e-07
0.8	1.345600	1.345600	-1.00e-07
0.9	1.446006	1.446006	-1.00e-07
1.0	1.562500	1.562500	-1.00e-07
1.1	1.696506	1.696506	-2.00e-07
1.2	1.849600	1.849600	-2.00e-07
1.3	2.023506	2.023506	-3.00e-07
1.4	2.220100	2.220100	-3.00e-07
1.5	2.441406	2.441406	-4.00e-07
1.6	2.689599	2.689600	-5.00e-07
1.7	2.967006	2.967006	-6.00e-07
1.8	3.276099	3.276100	-7.00e-07
1.9	3.619505	3.619506	-8.00e-07
2.0	3.999999	4.000000	-9.00e-07
2.1	4.420505	4.420506	-1.10e-06
2.2	4.884099	4.884100	-1.20e-06
2.3	5.394005	5.394006	-1.40e-06
2.4	5.953598	5.953600	-1.60e-06
2.5	6.566404	6.566406	-1.70e-06
2.6	7.236098	7.236100	-2.00e-06
2.7	7.966504	7.966506	-2.20e-06
2.8	8.761598	8.761600	-2.40e-06
2.9	9.625504	9.625506	-2.60e-06
3.0	10.562497	10.562500	-2.90e-06
3.1	11.577003	11.577006	-3.20e-06
3.2	12.673597	12.673600	-3.50e-06
3.3	13.857003	13.857006	-3.80e-06
3.4	15.132096	15.132100	-4.10e-06
3.5	16.503902	16.503906	-4.40e-06
3.6	17.977595	17.977600	-4.70e-06
3.7	19.558501	19.558506	-5.10e-06
3.8	21.252094	21.252100	-5.50e-06
3.9	23.064000	23.064006	-5.80e-06
4.0	24.999994	25.000000	-6.20e-06

t	y_numerik	y_exact	delta
4.1	27.066000	27.066006	-6.60e-06
4.2	29.268093	29.268100	-7.10e-06
4.3	31.612499	31.612506	-7.50e-06
4.4	34.105592	34.105600	-7.90e-06
4.5	36.753898	36.753906	-8.40e-06
4.6	39.564091	39.564100	-8.80e-06
4.7	42.542997	42.543006	-9.30e-06
4.8	45.697590	45.697600	-9.80e-06
4.9	49.034996	49.035006	-1.03e-05
5.0	52.562489	52.562500	-1.08e-05
5.1	56.287495	56.287506	-1.13e-05
5.2	60.217588	60.217600	-1.19e-05
5.3	64.360494	64.360506	-1.24e-05
5.4	68.724087	68.724100	-1.30e-05
5.5	73.316393	73.316406	-1.36e-05
5.6	78.145586	78.145600	-1.41e-05
5.7	83.219992	83.220006	-1.47e-05
5.8	88.548085	88.548100	-1.53e-05
5.9	94.138490	94.138506	-1.60e-05
6.0	99.999983	100.000000	-1.66e-05
6.1	106.141489	106.141506	-1.72e-05
6.2	112.572082	112.572100	-1.79e-05
6.3	119.300988	119.301006	-1.86e-05
6.4	126.337581	126.337600	-1.92e-05
6.5	133.691386	133.691406	-1.99e-05
6.6	141.372079	141.372100	-2.06e-05
6.7	149.389485	149.389506	-2.13e-05
6.8	157.753578	157.753600	-2.20e-05
6.9	166.474483	166.474506	-2.28e-05
7.0	175.562477	175.562500	-2.35e-05
7.1	185.027982	185.028006	-2.43e-05
7.2	194.881575	194.881600	-2.50e-05
7.3	205.133980	205.134006	-2.58e-05
7.4	215.796073	215.796100	-2.66e-05
7.5	226.878879	226.878906	-2.74e-05
7.6	238.393572	238.393600	-2.82e-05
7.7	250.351477	250.351506	-2.90e-05
7.8	262.764070	262.764100	-2.99e-05
7.9	275.642975	275.643006	-3.07e-05
8.0	288.999968	289.000000	-3.16e-05
8.1	302.846974	302.847006	-3.24e-05
8.2	317.196067	317.196100	-3.33e-05
8.3	332.059472	332.059506	-3.42e-05

t	y_numerik	y_exact	delta
8.4	347.449565	347.449600	-3.51e-05
8.5	363.378870	363.378906	-3.60e-05
8.6	379.860063	379.860100	-3.69e-05
8.7	396.905968	396.906006	-3.79e-05
8.8	414.529561	414.529600	-3.88e-05
8.9	432.743966	432.744006	-3.98e-05
9.0	451.562459	451.562500	-4.07e-05
9.1	470.998465	470.998506	-4.17e-05
9.2	491.065557	491.065600	-4.27e-05
9.3	511.777463	511.777506	-4.37e-05
9.4	533.148055	533.148100	-4.47e-05
9.5	555.191360	555.191406	-4.57e-05
9.6	577.921553	577.921600	-4.67e-05
9.7	601.352958	601.353006	-4.78e-05
9.8	625.500051	625.500100	-4.88e-05
9.9	650.377456	650.377506	-4.99e-05
10.0	675.999949	676.000000	-5.10e-05

5 PERTEMUAN MINGGU IV

16 September 2021

5.1 *Algorithm Design: Randomness*

5.1.1 Simulasi Monte Carlo

Menggunakan Simulasi *Monte Carlo* untuk menghitung luas integral. Cara kerjanya simpel, analoginya adalah melempar *darts*. Luas area di bawah kurva dihitung dengan cara:

$$L = \frac{\text{darts}_{\text{on target}}}{\text{darts}_{\text{All}}}$$

Metode ini merupakan metode *brute force*.

5.1.2 *Flowchart* Simulasi Monte Carlo

Berikut adalah flowchartnya:

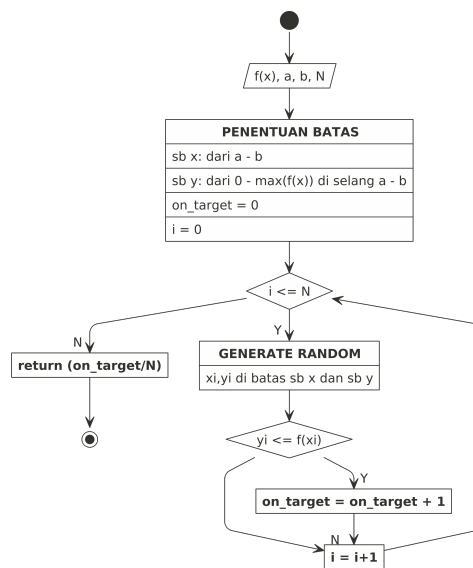


Figure 8: Flowchart Brute Force

5.1.3 Modifikasi Simulasi Monte Carlo

Kalau pakai simulasi Monte Carlo yang sebelumnya, kita terlalu *brute force* sehingga secara komputasi akan butuh waktu lebih lama dan *generate random* titik lebih lama. Oleh karena itu kita bisa memodifikasi menjadi sebagai berikut:

Idenya:

$$I = \int_a^b f(x)dx$$

dihitung sebagai:

$$\langle F^N \rangle = \frac{b-a}{N+1} \sum_{i=0}^N f(a + (b-a)\xi_i)$$

dengan

ξ_i adalah random number antara 0 dan 1

5.1.4 Flowchart Modifikasi Simulasi Monte Carlo

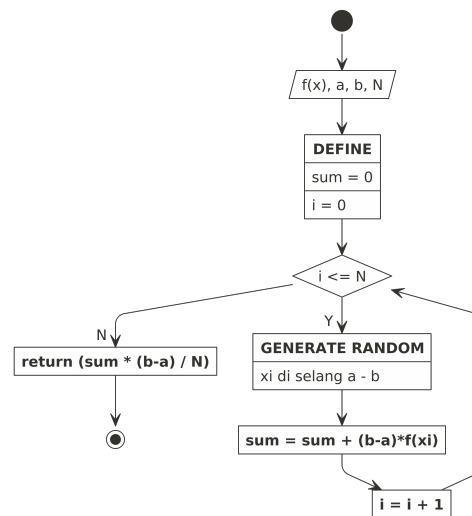


Figure 9: Flowchart Modifikasi Monte Carlo

6 PERTEMUAN MINGGU V

6.1 Shortest Path Problems

Menggunakan energi seminimum mungkin untuk menyusui *path* tertentu.

6.1.1 Breadth-First Search

Algoritma sederhana untuk mencari rute di graf. Misalkan:

Graf $G = (V, E)$

Dengan V adalah *vertex*.

Cara kerjanya adalah **systematically explores** semua kemungkinan *vertexes* yang bisa dilalui lalu mencari *smallest number* yang ada. Rute terpendek kemudian akan dicari.

Nanti setiap *vertexes* yang terlewati akan diberikan bobot. Rute terpendek akan ditemukan dari sana.

6.1.2 Bellman-Ford

Perbedaan algoritma ini dengan algoritma sebelumnya adalah **dimungkinkan ada bobot yang negatif**. Namun tidak boleh ada **negative cycle**.

Soal Contoh Jalankan *Bellman-Ford Algorithm* dengan titik z sebagai titik awal!

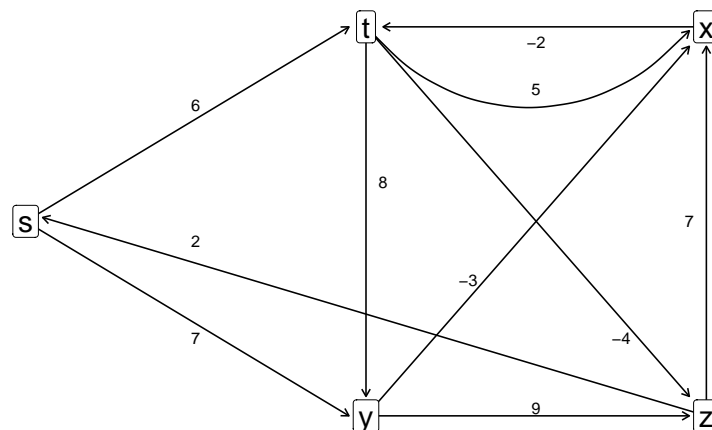


Figure 10: Graf Soal I

Jawab Kita akan bikin dulu algoritmanya. *Feeding*-nya kita buat sebagai `data.frame()` saja. Prinsipnya sederhana, yakni melakukan *relaxation* semua kemungkinan *edges* dari graf yang ada.

```
##      from to bobot
## 1      s  y      7
## 2      s  t      6
## 3      t  x      5
## 4      t  y      8
## 5      t  z     -4
## 6      y  x     -3
## 7      y  z      9
## 8      x  t     -2
## 9      z  x      7
## 10     z  s      2
```

```
##      titik bobot
## 1      z      0
## 2      t    Inf
## 3      y    Inf
## 4      x    Inf
## 5      s    Inf
```

Berikut adalah algoritmanya⁴:

```
if d[u] + c(u,v) < d[v]
  d[v] = d[u] + c(u,v)
```

Sederhana kan?

Berikut dalam **R**-nya:

```
# fungsi yang dibutuhkan
# ambil bobot edges dari u ke v
c = function(u,v){
  temp =
    graf_df %>%
    filter(from == u) %>%
    filter(to == v)
  temp$bobot
}
```

⁴<https://www.youtube.com/watch?v=FtN3BYH2Zes>

```

# nilai node u
d = function(u){
  temp =
    d_titik %>%
    filter(titik == u)
  temp$bobot
}

# update node v
upd = function(v,value){
  d_titik$bobot[d_titik$titik == v] <- value
}

# proses iterasi
for(ikanx in 1:10){ # diulang berkali-kali
  for(init in d_titik$titik){
    temp = graf_df %>% filter(from == init)
    for(u in temp$from){
      for(v in temp$to){
        if(d(u) + c(u,v) < d(v)){
          val = d(u) + c(u,v)
          upd(v,val)
        }
      }
    }
  }
}

d_titik

```

```

##      titik bobot
## 1      z      0
## 2      t      4
## 3      y      9
## 4      x      6
## 5      s      2

```

Lihat kembali grafnya:

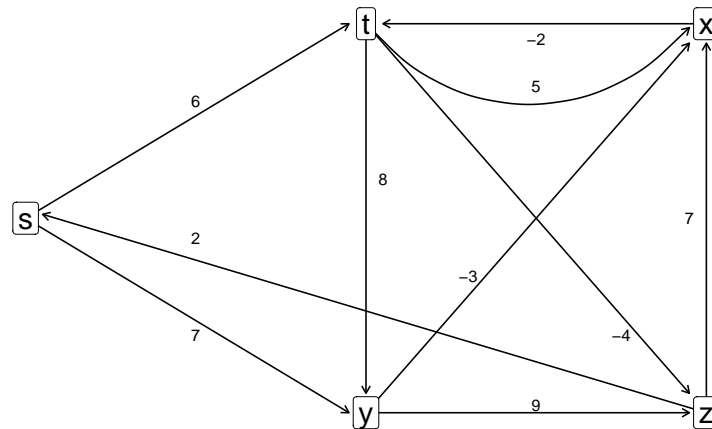


Figure 11: Graf Soal I

Kesimpulan Rutenya adalah **z-s-y-x-t**.

6.1.3 Dijkstra

Kalau **Dijkstra** mirip dengan **Bellman-Ford** tapi nilai bobotnya selalu **positif**.

6.1.4 *A-star* A^* Algorithm

Menggunakan *heuristic function* untuk menentukan *path* mana yang harus dipilih.

Heuristic functions yang bisa dipilih:

1. **Euclidean distance:** $h = \sqrt{(x - x')^2 + (y - y')^2}$.
2. **Manhattan distance:** $h = |x - x'| + |y - y'|$.
3. **Diagonal distance:** $h = \max \{|x - x'|, |y - y'|\}$.

7 PERTEMUAN MINGGU VI

7.1 *Genetic Algorithm*

Algoritma yang didasarkan dari *natural selection*. Karena ini *heuristic*, maka tidak ada jaminan bahwa solusi yang ada sudah paling optimal karena bergantung dari pemilihan *parent solution*. Perhatikan istilahnya bernama *parent*, karena ini mengikuti biologi. Dari *parent* bermutasi sampai jadi solusi. Kalau di analisa numerik kan disebutnya *initial solution*.

7.1.1 *Advantages vs Disadvantages*

7.1.2 *Basic Terminology*

- **Population** adalah semua kemungkinan solusi yang ada dari permasalahan.
- **Chromosomes** adalah **satu solusi** dari permasalahan kita.
- **Gene** adalah **satu elemen** dari suatu solusi yang ada.
- **Allele** adalah **isi** dari suatu elemen solusi.
- **Genotype** adalah populasi dari *computation space*.
- **Phenotype** adalah populasi dari solusi real.
- **Decoding** adalah proses dari *Genotype* ke *Phenotype*.
- **Encoding** adalah proses dari *Phenotype* ke *Genotype*.
- **Fitness Function** adalah fungsi yang mengukur seberapa **cocok** suatu solusi.
- **Genetic Operators** adalah proses alterasi dari kromosom. Bisa berupa *crossover*, mutasi, seleksi, dan lainnya.

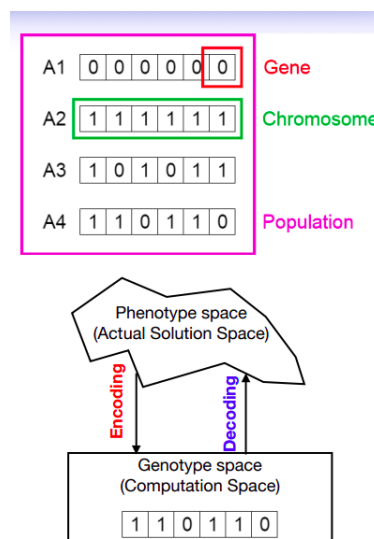


Figure 12: Basic Terminology

7.1.3 Basic Structure

Berikut adalah *flow* dari *genetic algorithm*:

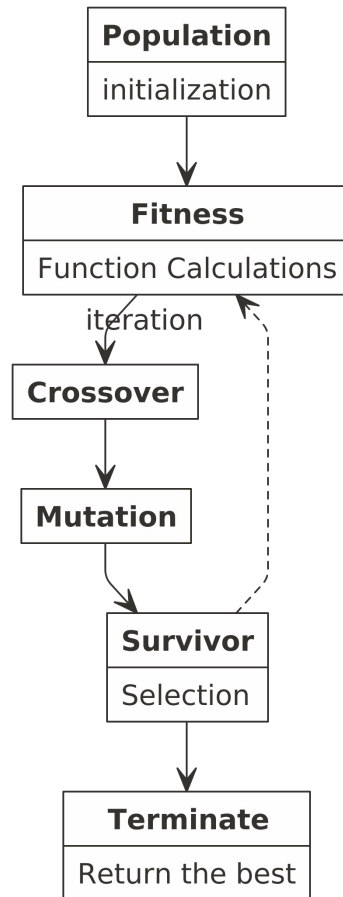


Figure 13: GA Flow

Salah satu tantangan dalam *GA* adalah menentukan populasi awalnya.

7.1.4 Contoh Soal⁵

Diberikan $2a^2 + b = 57$. Tentukan nilai a, b dengan *GA* yang meminimumkan:

$$f(a, b) = 2a^2 + b - 57$$

Jawab STEP I kita akan buat suatu populasi. Kita bisa *set* misalkan ada 6 kromosom secara *random*.

⁵<https://towardsdatascience.com/genetic-algorithm-explained-step-by-step-65358abe2bf>

##	[,1]	[,2]
## Cromosome1	1	6
## Cromosome2	2	9
## Cromosome3	2	3
## Cromosome4	3	8
## Cromosome5	9	3
## Cromosome6	8	8

STEP II proses *selection*, yakni memilih *fittest* kromosom. Caranya adalah dengan menghitung *fitness value*. Kromosom yang menghasilkan *low values* akan dipilih untuk generasi berikutnya.

Kita akan gunakan metode *roulette wheel*, yakni dengan menghitung:

$$FP = \frac{F_i}{\sum_{i=1}^n F_i}$$

Di mana FP menandakan *fitness probability* dan F_i menandakan *fitness value*.

Untuk memilih *fittest* kromosom, kita akan generate **6 random probabilities**.

##	[,1]	[,2]
## Cromosome1	3	8
## Cromosome2	2	9
## Cromosome3	2	3
## Cromosome4	2	9
## Cromosome5	2	3
## Cromosome6	2	9

Kita akan memilih kromosom yang jatuh di titik `ran_prob` sebagai *parents* di generasi berikutnya.

STEP III Berikutnya kita akan ubah nilai a dan b menjadi *binary* lalu menggabungkannya. Saya akan gunakan sistem 4 *bits*, sehingga akan didapatkan 8 *bits*.

Kemudian kita akan lakukan *crossover* atau perkawinan silang. Kita *set* terlebih dahulu *crossover rate*. Misalkan 0.5, artinya dari 6 kromosom, hanya 3 yang boleh **kawin**.

##	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]
## Cromosome1	1	1	NA	NA	1	0	0	0
## Cromosome2	1	0	NA	NA	1	0	0	1
## Cromosome3	1	0	NA	NA	1	1	NA	NA
## Cromosome4	1	0	NA	NA	1	0	0	1
## Cromosome5	1	0	NA	NA	1	1	NA	NA
## Cromosome6	1	0	NA	NA	1	0	0	1

STEP IV Kita akan lakukan *mutation*. *Mutation rate* juga harus kita definisikan terlebih dahulu.

##	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]
## Cromosome1	1	1	NA	NA	1	0	0	0
## Cromosome2	1	0	NA	NA	1	0	0	1
## Cromosome3	1	0	NA	NA	0	0	NA	NA
## Cromosome4	1	1	NA	NA	1	1	0	1
## Cromosome5	1	0	NA	NA	1	1	NA	NA
## Cromosome6	1	0	NA	NA	1	0	0	1

##	[,1]	[,2]
## Cromosome1	3	8
## Cromosome2	2	9
## Cromosome3	2	0
## Cromosome4	3	13
## Cromosome5	2	3
## Cromosome6	2	9

Proses di atas kita lakukan berulang-ulang hingga mendapatkan hasil paling optimal.

EPILOG

Alhamdulillah.