TUGAS MATA KULIAH

OPTIMISASI

MAX-FLOW PROBLEM USING JULIA

Dosen Pengampuh:

Ir. Novalio Daratha, S.T., M.Sc., Ph.D.

Disusun Oleh

Ismi Hafizdah Furqana (G1D021014)

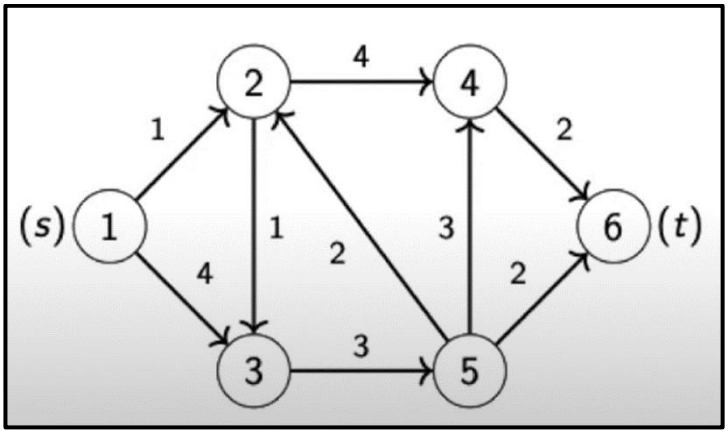
PROGRAM STUDI TEKNIK ELEKTRO

FAKULTAS TEKNIK

UNIVERSITAS BENGKULU

2024

1. Carilah aliran arus maksimal pada Aliran Jaringan (Network Flow) berikut!



2. Modelkan bentuk Aliran Jaringan di atas ke dalam bentuk matriks 6x6, sebagai berikut.

G = [

0 1 4 0 0 0

0 0 1 4 0 0

0 0 0 0 3 0

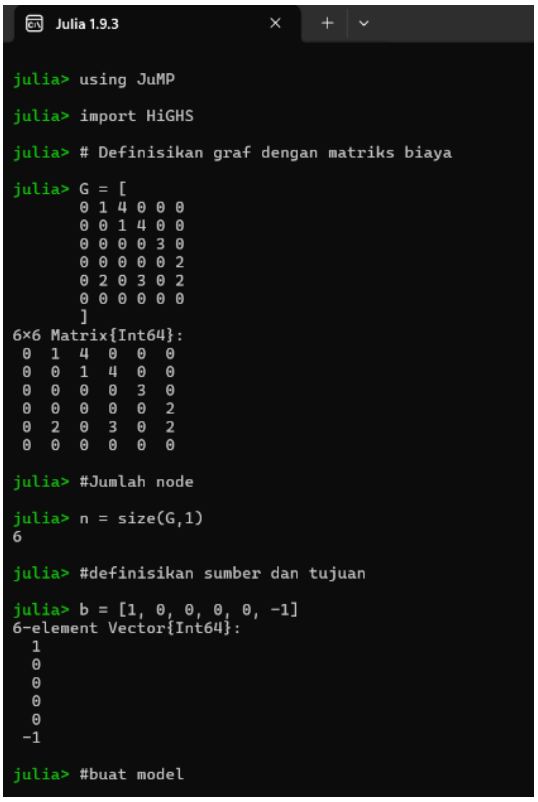
0 0 0 0 0 2

0 2 0 3 0 2

0 0 0 0 0 0

]

3. Buatlah program Julia untuk menyelesaikannya berdasarkan matriks yang telah dibua



4. Tahap pertama dalam membuat program ini adalah mengimpor paket

```
julia> using JuMP

julia> import HiGHS
```

5. Kemudian mendefinisikan graf dengan matriks

```
julia> G = [
            0 1 4 0 0 0
            0 0 1 4 0 0
            0 0 0 0 3 0
            0 0 0 0 0 2
            0 2 0 3 0 2
            0 0 0 0 0 0
            ]
6×6 Matrix{Int64}:
 0  1  4  0  0  0
 0  0  1  4  0  0
 0  0  0  0  3  0
 0  0  0  0  0  2
 0  2  0  3  0  2
 0  0  0  0  0  0
```

Matriks G mendefinisikan graf dengan biaya aliran antara node. Elemen G[i, j] menunjukkan biaya aliran dari node i ke node j. Jika biayanya 0, berarti tidak ada aliran langsung antara node tersebut.

6. Membuat jumlah node, Bagian ini menghitung jumlah node dalam graf G dengan mengambil ukuran dimensi pertama dari matriks G. Dalam contoh ini, n akan bernilai 6 karena matriks G berukuran 6x6.

```
julia> n = size(G, 1)
6
```

7. Setelah membuat jumlah node, definisikan sumber dan tujuan. Vektor b mendefinisikan sumber dan tujuan aliran. Nilai 1 menunjukkan sumber (node 1), dan nilai -1 menunjukkan tujuan (node 6). Nilai 0 menunjukkan node lainnya yang tidak memiliki aliran masuk atau keluar.
8. Kemudian kita dapat membuat model

```
julia> model = Model(HiGHS.Optimizer)
A JuMP Model
├ solver: HiGHS
├ objective_sense: FEASIBILITY_SENSE
├ num_variables: 0
├ num_constraints: 0
└ Names registered in the model: none
```

9. mendefinisikan variabel biner x untuk setiap arc dalam graf. Variabel ini akan bernilai 1 jika arc tersebut termasuk dalam jalur, dan 0 jika tidak.

```
julia> @variable(model, x[1:n, 1:n], Bin)
6×6 Matrix{VariableRef}:
 x[1,1]  x[1,2]  x[1,3]  x[1,4]  x[1,5]  x[1,6]
 x[2,1]  x[2,2]  x[2,3]  x[2,4]  x[2,5]  x[2,6]
 x[3,1]  x[3,2]  x[3,3]  x[3,4]  x[3,5]  x[3,6]
 x[4,1]  x[4,2]  x[4,3]  x[4,4]  x[4,5]  x[4,6]
 x[5,1]  x[5,2]  x[5,3]  x[5,4]  x[5,5]  x[5,6]
 x[6,1]  x[6,2]  x[6,3]  x[6,4]  x[6,5]  x[6,6]
```

10. Constraint : Arc dengan biaya 0. Constraint ini memastikan bahwa arc dengan biaya nol tidak termasuk dalam jalur. Jika biaya antara node i dan j adalah 0, maka x[i, j] harus bernilai 0.

```
julia> @constraint(model, [i=1:n, j=1:n; G[i, j] == 0], x[i, j] == 0)
JuMP.Containers.SparseAxisArray{ConstraintRef{Model, MathOptInterface.ConstraintIndex{MathOptInterface.ScalarAffineFunct
ion{Float64}, MathOptInterface.EqualTo{Float64}}, ScalarShape}, 2, Tuple{Int64, Int64}} with 27 entries:
  [1, 1]  =  x[1,1] == 0
  [1, 4]  =  x[1,4] == 0
  [1, 5]  =  x[1,5] == 0
  [1, 6]  =  x[1,6] == 0
  [2, 1]  =  x[2,1] == 0
  [2, 2]  =  x[2,2] == 0
  [2, 5]  =  x[2,5] == 0
  [2, 6]  =  x[2,6] == 0
  [3, 1]  =  x[3,1] == 0
  [3, 2]  =  x[3,2] == 0
  [4, 4]  =  x[4,4] == 0
    ⋮
  [4, 5]  =  x[4,5] == 0
  [5, 1]  =  x[5,1] == 0
  [5, 3]  =  x[5,3] == 0
  [5, 5]  =  x[5,5] == 0
  [6, 1]  =  x[6,1] == 0
  [6, 2]  =  x[6,2] == 0
  [6, 3]  =  x[6,3] == 0
  [6, 4]  =  x[6,4] == 0
  [6, 5]  =  x[6,5] == 0
  [6, 6]  =  x[6,6] == 0
```

11. Optimasi dan Hasil. Bagian ini menjalankan optimasi model dan menampilkan hasilnya.

```
julia> optimize!(max_flow)
Running HiGHS 1.7.2 (git hash: 5ce7a2753): Copyright (c) 2024 HiGHS under MIT licence terms
Coefficient ranges:
  Matrix [1e+00, 1e+00]
  Cost   [1e+00, 1e+00]
  Bound  [0e+00, 0e+00]
  RHS    [1e+00, 4e+00]
Presolving model
4 rows, 9 cols, 14 nonzeros  0s
4 rows, 9 cols, 14 nonzeros  0s
Presolve : Reductions: rows 4(-36); columns 9(-27); elements 14(-62)
Solving the presolved LP
Using EKK dual simplex solver - serial
  Iteration        Objective     Infeasibilities num(sum)
          0     0.0000000000e+00 Ph1: 0(0) 0s
          5    -4.0000000000e+00 Pr: 0(0) 0s
Solving the original LP from the solution after postsolve
Model   status      : Optimal
Simplex   iterations: 5
Objective value     :  4.0000000000e+00
HiGHS run time      :          0.02
```

Link GitHub : https://github.com/daichuu/Tugas-Optimisasi

Link youtube : https://youtu.be/kWvHU51EzgI