APLIKASI PENYELESAIAN MASALAH SHORTEST PATH DENGAN ALGORITMA DIJKSTRA

untuk memenuhi tugas besar matakuliah desain analisis dan algoritma

dosen pengampu

Dr. ZK Abdurachman Baizal M,kom



disusun oleh

Muhammad Fajar Al Hanief 1302160191

Putri Enita 1302162016

Iyon Priyono 1302160036

Reyhan Alkadri 1302164071

Kelas CS-40-05

PROGRAM STUDI ILMU KOMPUTASI FAKULTAS INFORMATIKA

TELKOM UNIVERSITY

TAHUN AJARAN 2018/2019

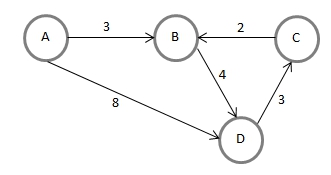
1. **PENJELASAN SINGKAT APLIKASI**

Program yang kelompok kami buat ialah program untuk menentukan lintasan dengan total jarak terpendek dari sebuah graph berarah. Bahasa yang kami gunakan ialah bahasa pemograman python.

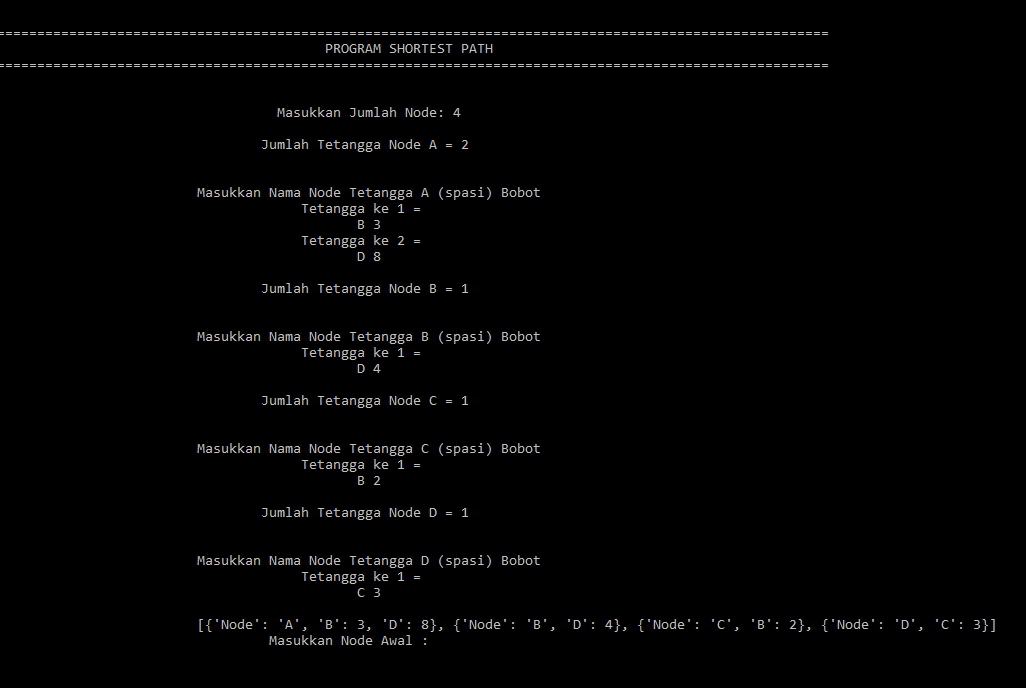
Dalam program ini user dapat menginputkan berapa jumlah node, berapa jumlah ketetanggaannya, serta berapa jarak antar node pada graph tersebut. Lalu user menginputkan node pertama yang menjadi titik awal dan menginputkan node terakhir yang menjadi titik tujuan yang akan dikunjungi. Setelah program dieksekusi, program akan menghasilkan output berupa lintasan node yang dilalui dengan total jarak terpendek.

1. **FUNGSIONALITAS APLIKASI**

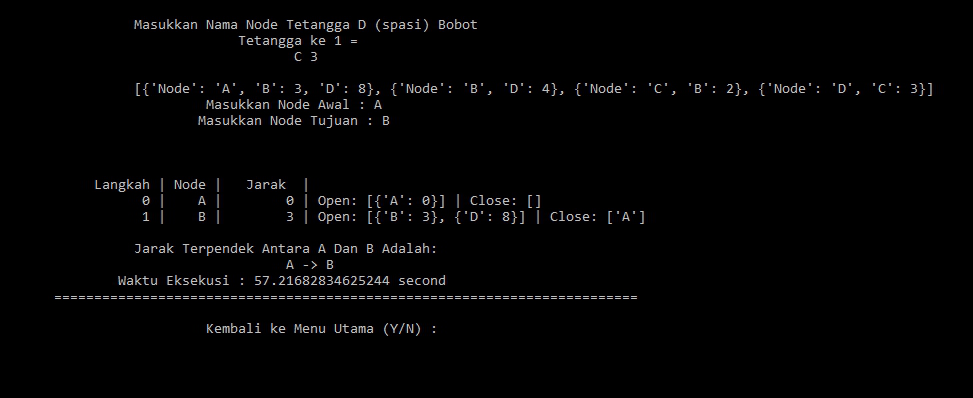
Fungsilitas aplikasi yang kami buat ialah untuk menentukan lintasan dengan total bobot lintasan terpendek. Dimana dalam pembuatan graphnya dapat diinputkan oleh user serta user dapat memilih node awal dan node tujuan akhirnya. Seperti pada graph berikut.



Pada graph tersebut dimisalkan user memilih node awal A dan node akhir B sehingga tampilan pada program seperti berikut.

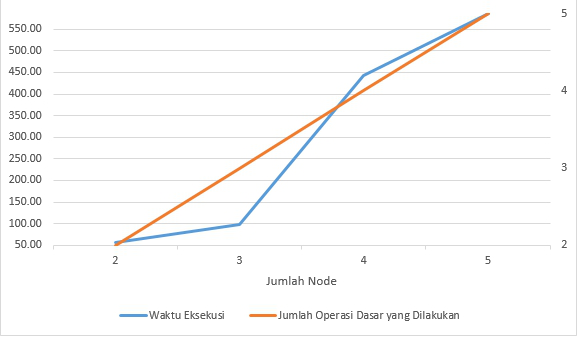


Sehingga akan menghasilkan output sebagai berikut.



1. **ANALISA KOMPLEKSITAS**
2. **Grafik Trend**

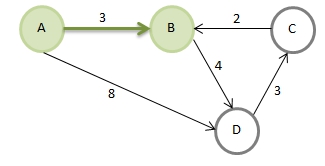
Grafik trend waktu eksekusi yang kami peroleh ialah seperti pada gambar berikut.



Pada vertikal sebelah kiri menunjukan angka waktu eksekusi dalam range 50 hingga 550 detik, sedangkan vertikal sebelah kanan menunjukan jumlah operasi dalam range 2 sampai 5. Garis biru menunjukan grafik waktu eksekusi terhadap jumlah node, sedangkan garis orange menunjukan grafik jumlah operasi terhadap jumlah node.

1. **Analisa Best Case**

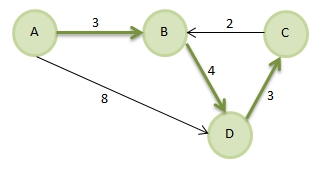
Best case pada pencarian lintasan dengan total bobot terpendek ialah jika suatu node hanya memerlukan satu langkah untuk mencapai node tujuan yang mendapatkan hasil optimum. Seperti pada sebuah graph berarah berikut.



Apabila dipilih A sebagai node awal dan node B menjadi node tujuan, maka lintasan dengan total bobot terpendeknya adalah A-B = 3 karena berdasarkan gambar graph diatas, total bobot lintasan tersebut merupakan solusi yang paling optimum.

1. **Analisa Worst Case**

Worst case pada pencarian lintasan dengan total bobot terpendek ialah jika suatu node harus melewati semua node untuk mencapai node tujuan untuk mendapatkan hasil yang optimum.



Pada gambar graph diatas node A menjadi node awal dan node C menjadi node tujuan. Dimana semua node dilalui dan mencapai hasil yang optimum dengan total bobot lintasan A-B-D-C = 10. Karena jika tidak semua node dikunjungi maka lintasan menjadi A-D-C = 11 yang menyatakan total bobot lintasan tersebut lebih besar dari sebelumnya, sehingga lintasan A-D-C bukan hasil yang optimum.

1. **Kompleksitas Waktu**

Kita melakukan eksperimen pada program untuk mengetahui kompleksitas waktu, dengan menggunakan rumus T(n) dimana n diasumsikan sebagai vertex atau node. Sehingga didapat rumus sebagai berikut.

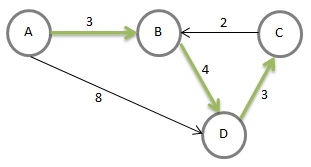
Batas atas = Jumlah node pada graph

Batas bawah = Jumlah node yang dilalui

Step = Jumlah operasi dasar

1. Eksperimen pertama

Mencoba menghitung kompleksitas waktu pada graph berikut.



Kami mencoba A sebagai node awal dan C sebagai node tujuan. Sehingga didapat hasil analisa algortimta djisktra seperti pada tabel berikut.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Node Terkunjungi | A | | B | C | D | Operasi dasar dilakukan |
|  | ∞ | | ∞ | ∞ | ∞ | 0 |
| {A} | | 0 | 3 | ∞ | 8 | 1 |
| {A,B} | | 0 | 3 | ∞ | 7 | 2 |
| {A,B,D} | | 0 | 3 | 10 | 7 | 3 |
| {A,B,D,C} | | 0 | 3 | 10 | 7 | 4 |

Berdasarkan tabel diatas maka didapat

untuk membuktikan T(n) € O( maka perlu dibuktikan T(n) ≤ cf(n). Maka f(n) = , misal c = 3 sehingga cf(n) = 3 artinya kita buktikan T(n) ≤ 3

|  |  |  |
| --- | --- | --- |
| n | T(n) | 3 |
| 4 |  | 48 |

Dari hasil tabel diatas terbukti T(n) ≤ cf(n) sehingga terbukti juga T(n) € O(.

1. Eksperimen kedua

Dengan langkah pengerjaan yang sama seperti eksperimen sebelumnya pada graph berikut. Dimana node D menjadi node awal dan node B menjadi node tujuan akhir sehingga didapat hasil seperti berikut.

2

4

3

2

B

C

D

A

8

1

3

1

E

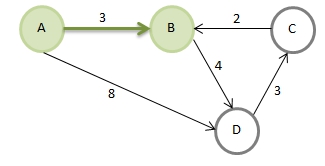
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Node Terkunjungi | A | | B | C | D | E | Operasi dasar dilakukan |
|  | ∞ | | ∞ | ∞ | ∞ | ∞ | 0 |
| {D} | | ∞ | ∞ | 3 | 0 | 1 | 1 |
| {D,E} | | 1 | ∞ | 3 | 0 | 1 | 2 |
| {D,E,A} | | 2 | 2 | 3 | 0 | 1 | 3 |
| {D,E,A,B} | | 2 | 4 | 3 | 0 | 1 | 4 |

untuk membuktikan T(n) € O( maka perlu dibuktikan T(n) ≤ cf(n). Maka f(n) = , misal c = 3 sehingga cf(n) = 3 artinya kita buktikan T(n) ≤ 3

|  |  |  |
| --- | --- | --- |
| n | T(n) | 3 |
| 5 |  | 125 |

Dari hasil tabel diatas terbukti T(n) ≤ cf(n) sehingga terbukti juga T(n) € O(.

1. Eksperimen ketiga



Seperti graph di atas dimana node A menjadi node awal, dan node B menjadi node tujuan akhir sehingga kompleksitas waktu yang didapat

1. **PENJELASAN ALGORITMA**

Pada program ini kami menerapkan algoritma dijkstra, algoritma ini menggunakan prinsip greedy dalam mencari solusi optimum pada setiap langkah yang dilalui. Input graph berarah dan berbobot (V,E) dimana V merupakan vertex atau node, dan E merupakan edge atau sisi yang harus berbobot positif. Jumlah bobot sisi-sisi pada sebuah lintasan adalah total bobot lintasan tersebut.

Ilustrasi kerja program yang kami buat ialah dengan mencoba menginputkan dengan jumlah node 4 yaitu A, B, C, D dimana node A memiliki jumlah tetangga 2 yaitu B dan D. Bobot lintasan dari A ke B yaitu 3, sedangkan dari A ke D yaitu 8. Node B memiliki jumlah tetangga 1 node yaitu D dengan bobot 4. Lalu node D memiliki jumlah tetangga yaitu 1 node C dengan bobot 3, terakhir node C bertetangga

8

2

3

C

4

D

3

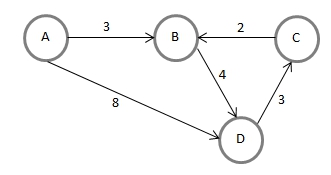
B

A

dengan node B saja dan memiliki bobot 2. Apabila digambarkan menjadi sebuah graph, akan tampak pada berikut ini.

Pada uji coba ini kami menginputkan node awal yaitu A dan node akhir yaitu C menjadi node tujuan yang akan dikunjungi. Sehingga bila dieksekusi dengan algoritma dijkstra menjadi seperti berikut ini.

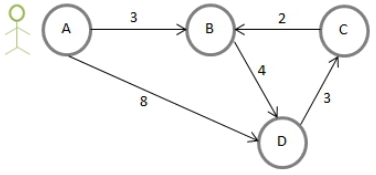
Langkah pertama



Dimana semua node belum ada yang dikunjungi, maka diberikan tanda tak hingga pada setiap node dalam tabel berikut.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Node Terkunjungi | A | B | C | D |
|  | ∞ | ∞ | ∞ | ∞ |

Langkah kedua

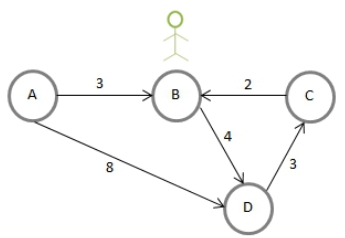


Kami memilih node A sebagai node awal dimana node A bertetangga dengan node B dan D yang memiliki masing-masing bobot lintasan 3 dan 8, sehingga terdapat update pada tabel seperti berikut.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Node Terkunjungi | A | | B | C | D |
|  | ∞ | | ∞ | ∞ | ∞ |
| {A} | | 0 | 3 | ∞ | 8 |

Langkah ketiga

Karena akan mencari lintasan dengan total jarak terpendek, maka dipilihlah node B yang memiliki lintasan bobot paling kecil seperti pada gambar berikut.

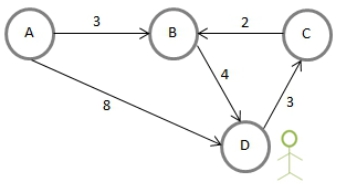


Node B hanya bertetangga dengan node D yang memiliki bobot lintasan 4, sehingga terdapat update pada tabel menjadi seperti berikut.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Node Terkunjungi | A | | B | C | D |
|  | ∞ | | ∞ | ∞ | ∞ |
| {A} | | 0 | 3 | ∞ | 8 |
| {A,B} | | 0 | 3 | ∞ | 7 |

Langkah ketiga

Karena node B hanya bertetangga dengan node D sehingga node D yang sekarang dikunjungi.

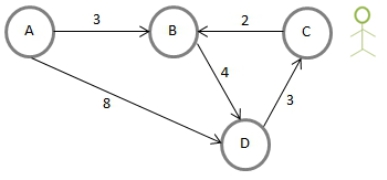


Node D bertetangga dengan node C, sehingga terdapat update pada tabel seperti berikut.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Node Terkunjungi | A | | B | C | D |
|  | ∞ | | ∞ | ∞ | ∞ |
| {A} | | 0 | 3 | ∞ | 8 |
| {A,B} | | 0 | 3 | ∞ | 7 |
| {A,B,D} | | 0 | 3 | 10 | 7 |

Langkah keempat

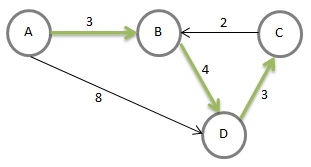
Setelah mengetahui bahwa bobot lintasan menuju node C dari node D ialah 3 maka dikunjungilah node C.



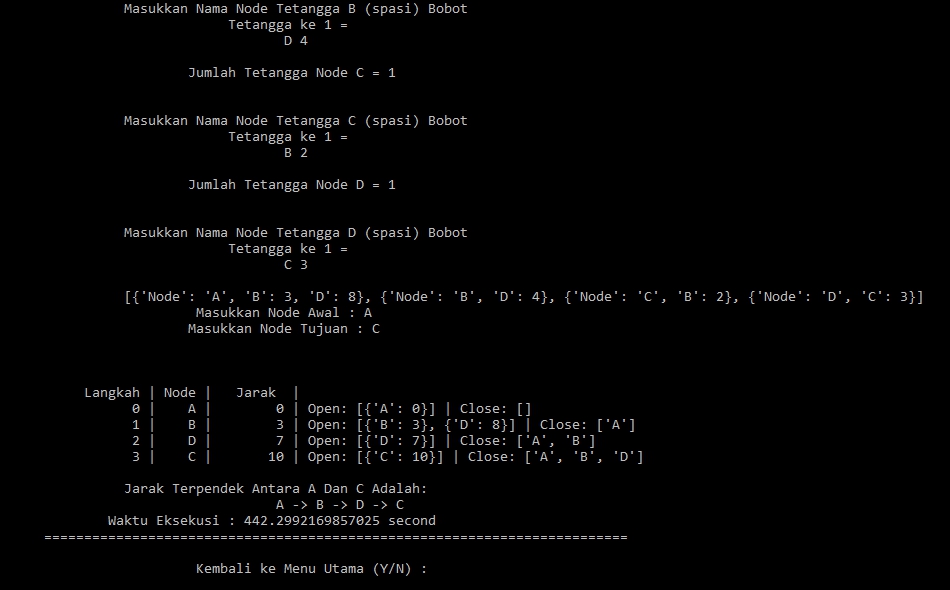
Sehingga update tabel menjadi seperti berikut.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Node Terkunjungi | A | | B | C | D |
|  | ∞ | | ∞ | ∞ | ∞ |
| {A} | | 0 | 3 | ∞ | 8 |
| {A,B} | | 0 | 3 | ∞ | 7 |
| {A,B,D} | | 0 | 3 | 10 | 7 |
| {A,B,D,C} | | 0 | 3 | 10 | 7 |

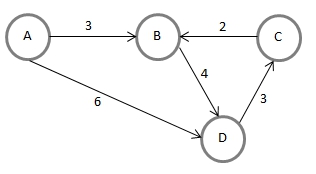
Maka ditemukanlah lintasan dari node A menuju node C dengan total bobot terpendek ialah 10 dengan lintasan A-B-D-C.



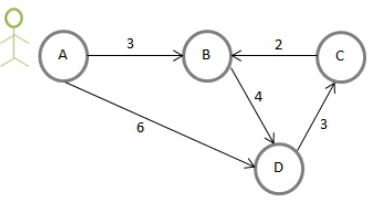
Maka sesuai dengan hasil pada program yang kami buat seperti berikut ini.



Lalu kami juga melakukan uji coba kedua yang bertujuan untuk mengetahui apakah program kami berhasil terdapat strategi backtracking untuk mencari total bobot lintasan bila lintasan bobot node A ke node D pada graph di atas dirubah menjadi 6 seperti pada gambar berikut.

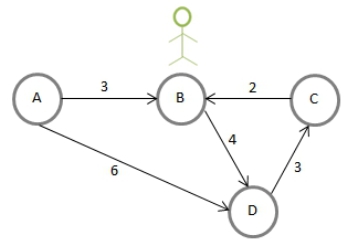


Node A menjadi node awal dan node D menjadi node tujuan, sehingga seperti berikut.



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Node Terkunjungi | A | | B | C | D |
|  | ∞ | | ∞ | ∞ | ∞ |
| {A} | | 0 | 3 | ∞ | 6 |

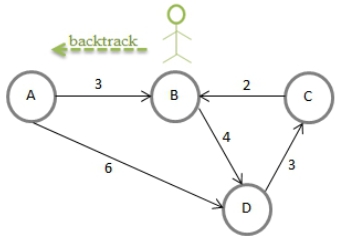
Langkah selanjutnya, dipilih node B karena memiliki bobot lintasan paling minimum yaitu 3.



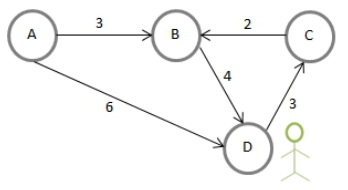
Untuk menuju node D dari node B maka total bobot lintasan menjadi 7 sehingga terdapat update tabel menjadi seperti berikut.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Node Terkunjungi | A | | B | C | D |
|  | ∞ | | ∞ | ∞ | ∞ |
| {A} | | 0 | 3 | ∞ | 6 |
| {A,B} | | 0 | 3 | ∞ | 7 |

Karena total bobot lintasan A-B-D = 7 maka dilakukan backtracking, dari posisi sekarang di node B kembali ke node A, dan langsung menuju node tujuan yaitu node D.

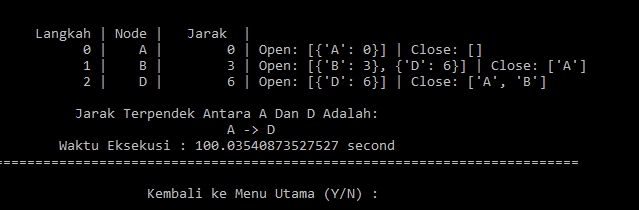


|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Node Terkunjungi | A | | B | C | D |
|  | ∞ | | ∞ | ∞ | ∞ |
| {A} | | 0 | 3 | ∞ | 6 |
| {A,B} | | 0 | 3 | ∞ | 7 |
| {A} | | 0 | 3 | ∞ | 6 |



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Node Terkunjungi | A | | B | C | D |
|  | ∞ | | ∞ | ∞ | ∞ |
| {A} | | 0 | 3 | ∞ | 6 |
| {A,B} | | 0 | 3 | ∞ | 7 |
| {A} | | 0 | 3 | ∞ | 6 |
| {A,D} | | 0 | 3 | ∞ | 6 |

Sehingga didapatkan total bobot lintasan A-D = 6 yang merupakan hasil paling optimum. Hasil akhir tersebut ternyata sesuai dengan uji coba yang kami lakukan pada program seperti berikut.



Simpulan dari setiap uji coba yang kami lakukan pada program terdapat hasil yang sama dengan hasil uji coba secara analisis.