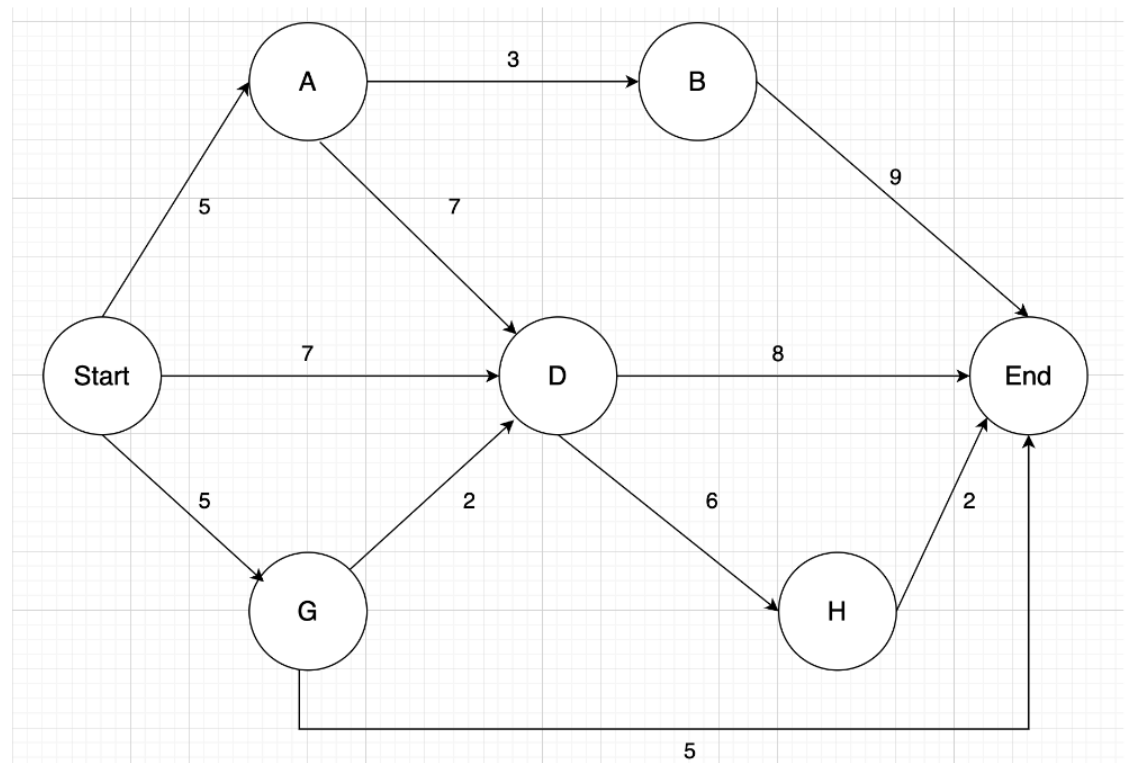
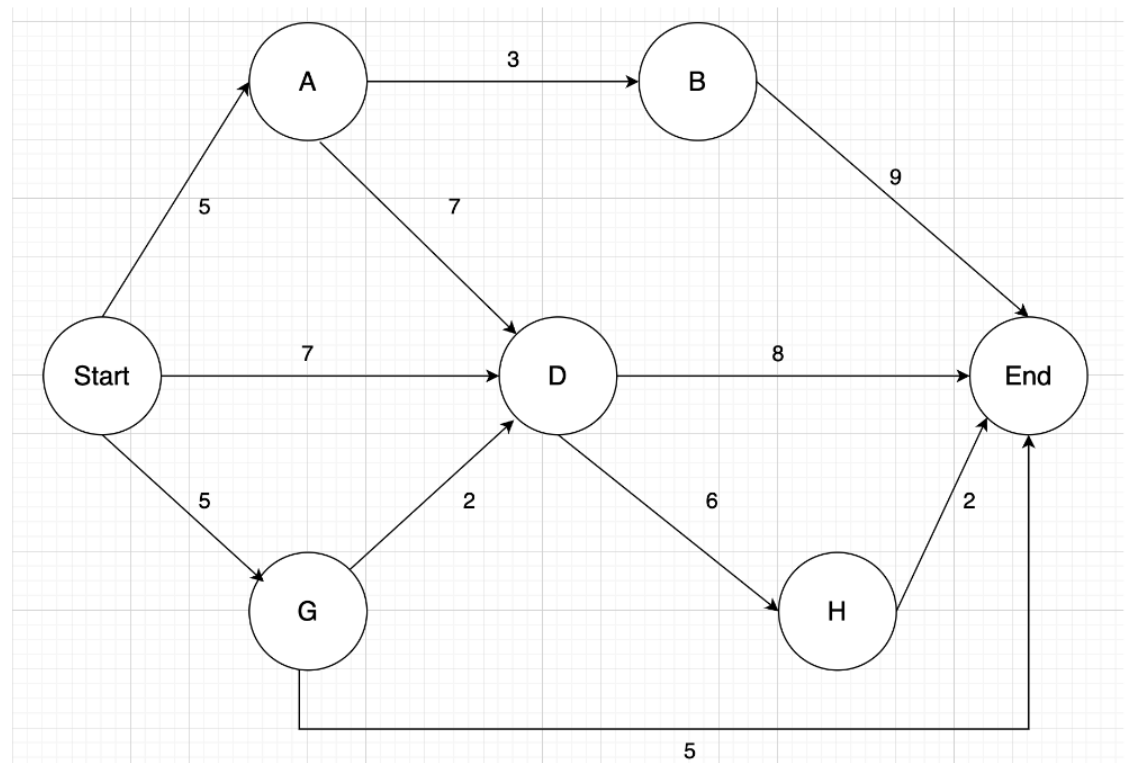
**DJIKSTRA**

|  |  |  |
| --- | --- | --- |
| Vertex | Shortest distance from Start | Previous Vertex |
| Start | - | - |
| A |  |  |
| B |  |  |
| D |  |  |
| G |  |  |
| H |  |  |
| End |  |  |



Step 1:  
- Pertama kita tahu bahwa program akan mampu menghitung jarak terpendek dari node start ke node end, karena masi pada node start sehingga beban jarak pada node start masih 0 dan tidak ada informasi mengenai previous vertex.

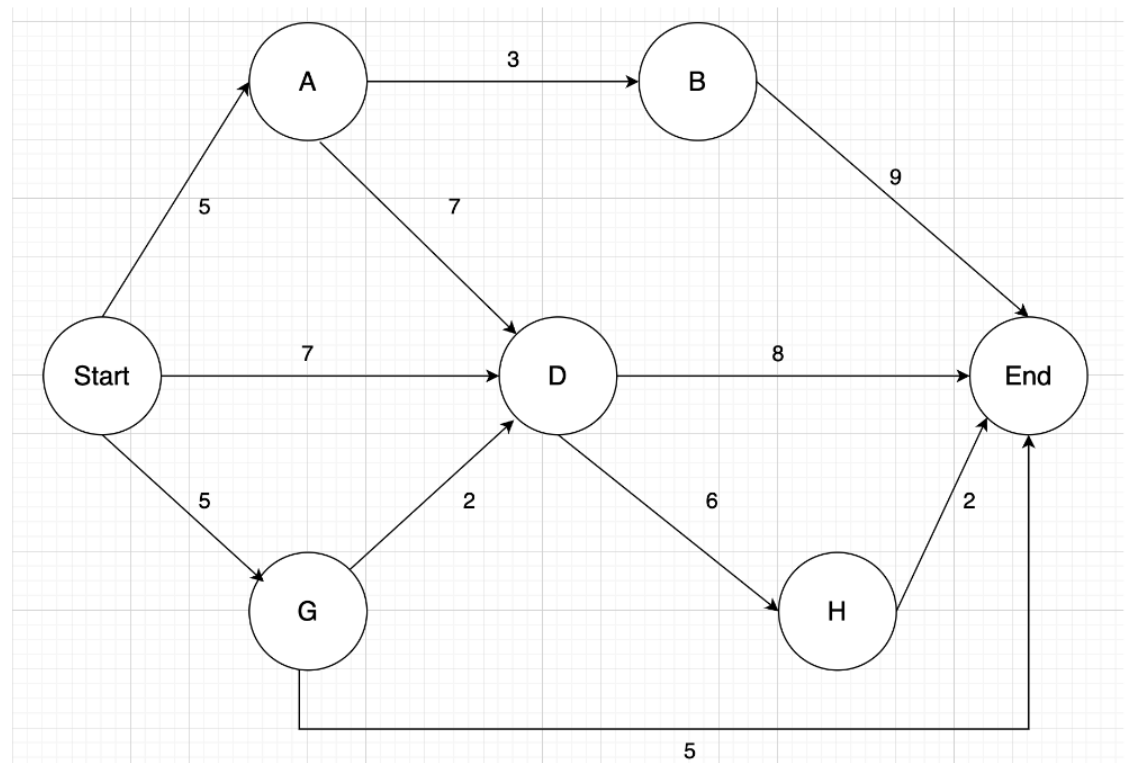
|  |  |  |
| --- | --- | --- |
| Vertex | Shortest distance from Start | Previous Vertex |
| Start | - | - |
| A | 5 | Start |
| B |  |  |
| D | 7 | Start |
| G | 5 | Start |
| H |  |  |
| End |  |  |





Step 2:  
- Selanjutnya setelah mengunjungi dirinya sendiri, node akan mengunjungi semua tetangga yang berbagi edges yaitu node A, node D dan node G. Saat mengunjungi node A, node D, dan node G data beban jarak pada titik tersebut akan langsung di update karena tidak ada data sebelumnya yang tersimpan pada node A,D, dan G, beserta informasi mengenai node sebelumnya yaitu node start. Pada node A beban jarak akan menjadi 5, node D menjadi 7, dan node G menjadi 5. Karena node A,D, dan G merupakan tetangga dari node start maka informasi previous vertex pun akan di update menjadi titik start. Setelah titik start mengunjungi semua tetangganya maka node A akan dihilangkan.

|  |  |  |
| --- | --- | --- |
| Vertex | Shortest distance from Start | Previous Vertex |
| Start | - | - |
| A | 5 | Start |
| B | 8 | A |
| D | 7 | Start |
| G | 5 | Start |
| H |  |  |
| End |  |  |

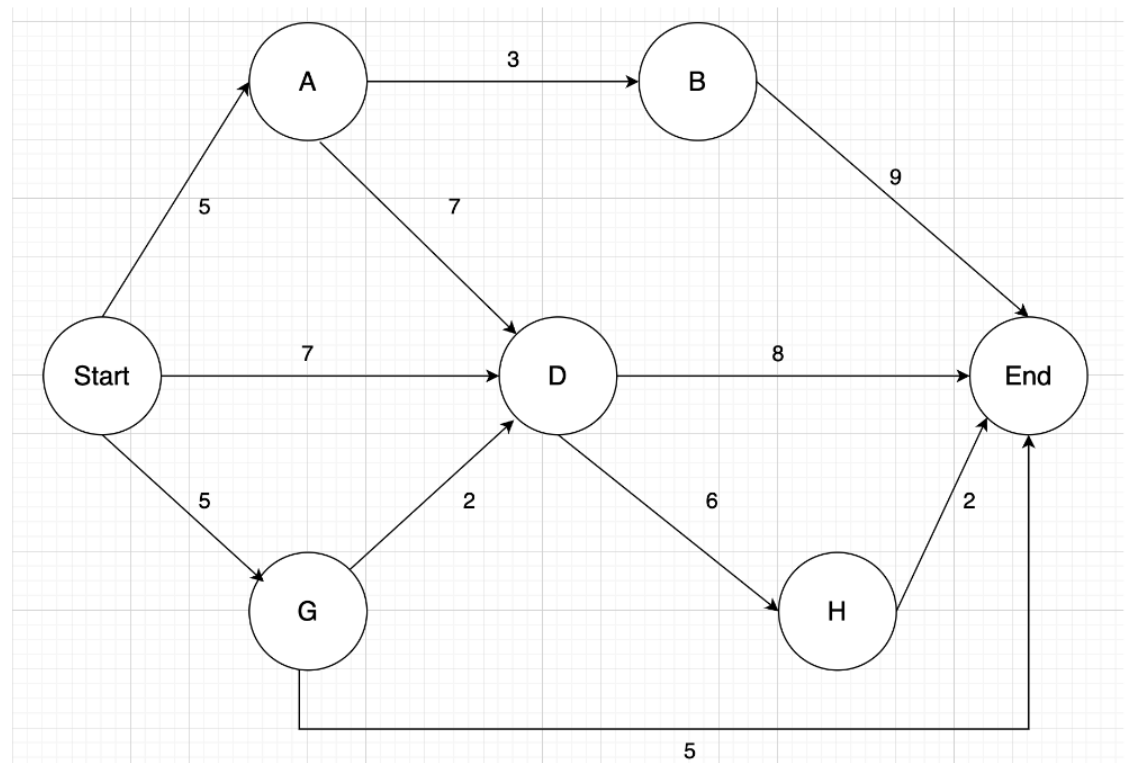




Step 3:

Setelah itu, node A akan mengunjungi seluruh tetangganya yaitu node D dan node B, karena node A memiliki beban jarak 5 maka data beban pada node B akan diupdate menjadi 5+3 = 8 dan informasi mengenai previous vertex akan diganti menjadi A ,sedangkan saat node A mengunjungi node D memerlukan jarak 5+7=12 karena data lebih besar daripada data sebelumnya maka data tidak diupdate apapun pada node D. Node A pun akan dihilangkan karena telah mengunjungi semua anggotanya.

|  |  |  |
| --- | --- | --- |
| Vertex | Shortest distance from Start | Previous Vertex |
| Start | - | - |
| A | 5 | Start |
| B | 8 | A |
| D | 7 | Start |
| G | 5 | Start |
| H | 13 | D |
| End | 15 | D |

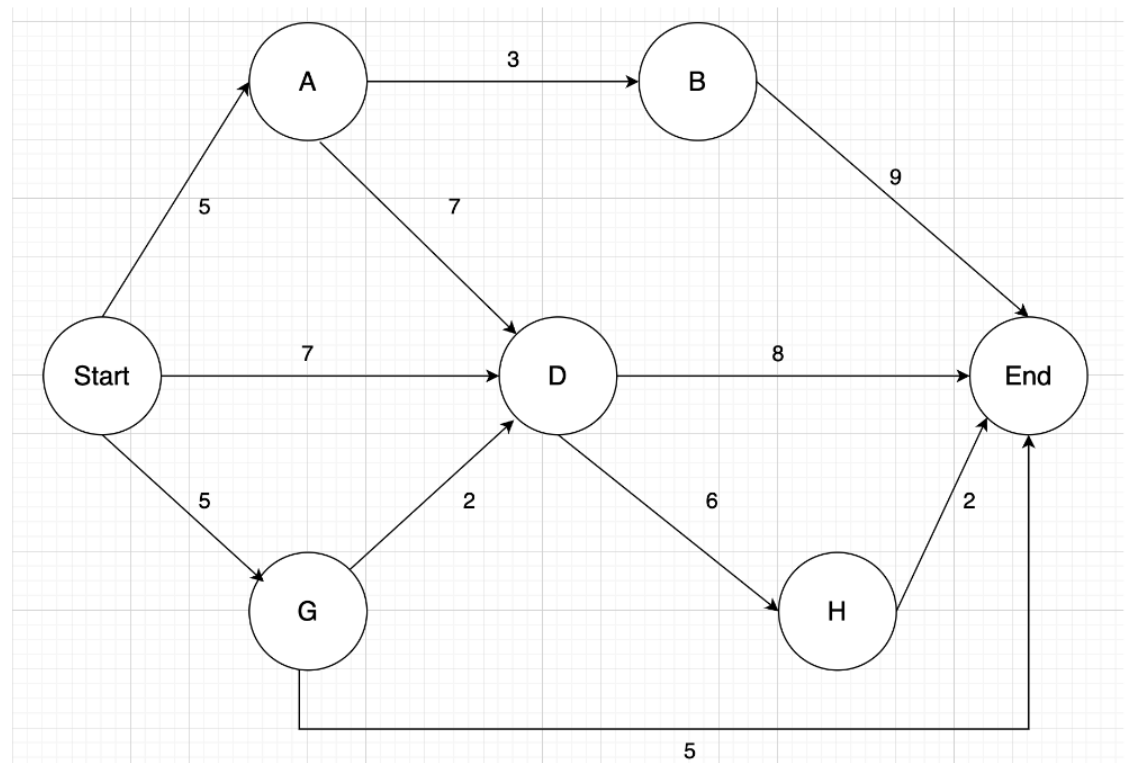




Step 4:

Setelah node A mengunjungi seluruh tetangganya, sekarang node D yang akan mengunjungi seluruh tetangganya yaitu node H dan node End. Data pada node H pun akan langsung diupdate karena tidak ada data sebelumnya menjadi 7+6=13 dan previous vertex akan menjadi D, sedangkan data di node end akan menjadi 7+8=15 dan informasi previous vertex akan diupdate menjadi node D. Node D pun akan dihilangkan karena telah mengunjungi seluruh tetangganya.

|  |  |  |
| --- | --- | --- |
| Vertex | Shortest distance from Start | Previous Vertex |
| Start | - | - |
| A | 5 | Start |
| B | 8 | A |
| D | 7 | Start |
| G | 5 | Start |
| H | 13 | D |
| End | 10 | G |

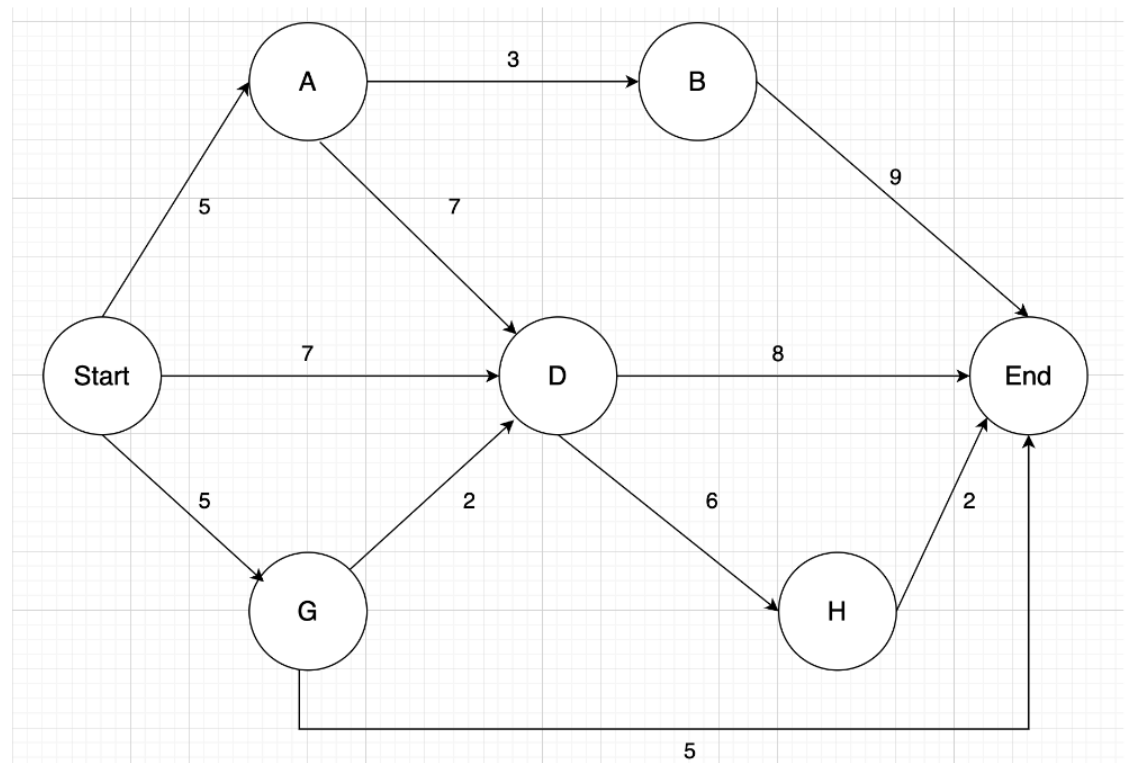




Step 5:

Setelah node D mengunjungi seluruh tetangganya, selanjutnya adalah node G akan mengunjungi seluruh tetangganya yaitu node D dan node End, jarak yang diperlukan dari node G ke D adalah 5+2=7 karena terdapat data pada node D sebelumnya, jarak terpendek pun akan dibandingkan, karena dari node G ke D memiliki jarak yang sama dengan node Start ke node D maka data tidak diupdate. Setelah itu jarak dari node G ke node End adalah 5+5=10 karena ada data informasi jarak pada node End maka jarak terpendek akan dibandingkan dan karena node G ke End hanya memerlukan beban jarak 10 sedangkan node D ke End 15 maka data beban jarak akan diupdate menjadi 10 dan informasi mengenai previous vertex akan diupdate menjadi G. Karena node G telah mengunjungi semua tetangganya maka node G akan dihilangkan.

|  |  |  |
| --- | --- | --- |
| Vertex | Shortest distance from Start | Previous Vertex |
| Start | - | - |
| A | 5 | Start |
| B | 8 | A |
| D | 7 | Start |
| G | 5 | Start |
| H | 13 | D |
| End | 10 | G |





Step 6:

Selanjutnya titik B akan mengunjungi satu-satunya tetangganya yaitu node End, dari node B ke End memerlukan beban jarak 8+9=17 karena beban jaraknya lebih besar daripada node G ke End maka data tidak diupdate, setelah itu node H akan mengunjungi tetangganya yaitu node End, beban jarak yang di perlukan untuk mengunjungi tetangganya adalah 13+2=15 dan setelah dibandingkan dengan data yang terdapat pada end sebelumnya yaitu 10 maka data tidak diupdate karena jarak dari node G ke node End lebih dekat. Node B dan node H pun akan dihilangkan karena tidak memiliki tetangga lagi, begitu pula dengan node End, karena tidak memiliki tetangga maka node End tidak akan mengunjungi node manapun. Sehingga setelah program telah selesai program telah menyimpan data data yang diperlukan dari Node A ke node End. Maka jalur terpendek dapat dilihat dari informasi previous vertex node end yaitu G, previous vertex node G adalah start, maka lintasan terpendek dari node start ke end adalah Start – G – End.