

UNIP – UNIVERSIDADE PAULISTA

Sistemas de Informação

**TRABALHO DE GRAFO**

**ALGORITMO DE DIJSTRAK**

Mauricio Galendi - C6036A-8

Kauby Silva – C7409B-5

Abril /2017

* **Implementação do Dijstrak em Java**

1. import java. util. ArrayList;  
2. import java. util. Collections;  
3. import java. util. List;  
4.  
5. public class Dijkstra {  
6.  
7.  
8.  
9. // Numero de vertices do gráfico  
10. static final int V = 9;  
11.  
12.  
13. // funcao de utilidade para encontrar o vertice com a distância mínima,  
14. // a partir do conjunto de vertices ainda não incluidas no  
15. // caminho mais curto  
16.  
17. private static int minDistance(int[ ] dist, boolean[ ] verticeProcesado)  
18. {  
19. // Inicializar min valor  
20. int min = Integer. MAX\_VALUE; int min\_index=0;  
21.  
22. for (int v = 0; v < V; v++)  
23. if (verticeProcesado[ v] == false && dist[ v] <= min) {  
24. min = dist[ v] ;  
25. min\_index = v;  
26. }  
27.  
28. return min\_index;  
29. }  
30.  
31. // funcao de utilidade para imprimir as distancias matriz calculados  
32. private static void printSolution(int[ ] dist, int n)  
33. {  
34. System. out. println("" distancia vertice da origem \ n "") ;  
35. for (int i = 0; i < V; i++)  
36. System. out. println(i + " \t\t " + dist[ i] ) ;  
37. }  
38.  
39. private static void dijkstra(int[ ] [ ] grafo, int src)  
40. {  
41. int[ ] dist = new int[ V] ;  
42. // dist [ i] armazena a distancia mais curta entre src para o vertice i  
43.  
44. boolean[ ] verticeProcesado = new boolean[ V] ;  
45. //Este arranjo e verdade se o vértice i já foi processado  
46.  
47. // Inicializar todas as distâncias tão infinito e stpSet [ ] como falsa  
48. for (int i = 0; i < V; i++) {  
49. dist[ i] = Integer. MAX\_VALUE;  
50. verticeProcesado[ i] = false;  
51. }  
52. // A distancia desde a origem ate o mesmo vertice e sempre 0  
53. dist[ src] = 0;  
54.  
55. //Encontrar o caminho mais curto para todos os vértices  
56. for (int count = 0; count < V‐1; count++)  
57. {  
58.  
59. // Pegue o vertice com a distancia mínima entre o vértice cojunto ainda não  
processados  
60. // src na primeira iteração sempre voltava  
06/04/2017 Dijstrak.java  
2/2  
61. int u = minDistance(dist, verticeProcesado) ;  
62.  
63. // ele é marcado como já processados  
64. verticeProcesado[ u] = true;  
65.  
66. // Atualizacao do valor dist dos vertices adjacentes do vertice escolhido.  
67. for (int v = 0; v < V; v++)  
68.  
69. // a dist é atualizado [ v] só se for verticeProcesado, não é um  
70. // arco de uavy o peso total da estrada de src para percorrer ou é  
71. // menor do que o valor atual de dist [ v]  
72. if (! verticeProcesado[ v] && grafo[ u] [ v] > 0 && dist[ u] ! = Integer. MAX\_VALUE  
73. && dist[ u] +grafo[ u] [ v] < dist[ v] )  
74. dist[ v] = dist[ u] + grafo[ u] [ v] ;  
75. }  
76.  
77. // a matriz é impresso com distâncias  
78. printSolution(dist, V) ;  
79. }  
80.  
81. // programa para testar acima função  
82. public static void main(String[ ] args)  
83. {  
84. /\* Vamos criar o gráfico exemplo discutido acima \*/  
85. int[ ] [ ] graph = {{0, 4, 0, 0, 0, 0, 0, 8, 0},  
86. {4, 0, 8, 0, 0, 0, 0, 11, 0},  
87. {0, 8, 0, 7, 0, 4, 0, 0, 2},  
88. {0, 0, 7, 0, 9, 14, 0, 0, 0},  
89. {0, 0, 0, 9, 0, 10, 0, 0, 0},  
90. {0, 0, 4, 0, 10, 0, 2, 0, 0},  
91. {0, 0, 0, 14, 0, 2, 0, 1, 6},  
92. {8, 11, 0, 0, 0, 0, 1, 0, 7},  
93. {0, 0, 2, 0, 0, 0, 6, 7, 0}  
94. };  
95.  
96. dijkstra(graph, 0) ;  
97. }  
98. }

* **Estrutura e Analise do algoritmo.**

Algoritmo de Dijkstra

1. Seja G(V,A) um grafo orientado e s um vértice de G:
2. Atribua valor zero à estimativa do custo mínimo do vértice s (a raiz da busca) e infinito às demais estimativas;
3. Atribua um valor qualquer aos precedentes (o precedente de um vértice t é o vértice que precede t no caminho de custo mínimo de s para t);
4. Enquanto houver vértice aberto:
   * 1. Seja k um vértice ainda aberto cuja estimativa seja a menor dentre todos os vértices abertos;
     2. Feche o vértice k
     3. Para todo vértice j ainda aberto que seja sucessor de k faça:
        + 1. Some a estimativa do vértice k com o custo do arco que une k a j;
          2. Caso esta soma seja melhor que a estimativa anterior para o vértice j, substitua-a e anote k como precedente de j.

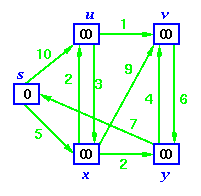
Exemplo 1

Passo 1

* Inicialmente todos os nodos tem um custo infinito, exceto

s (a raiz da busca) que tem valor 0:

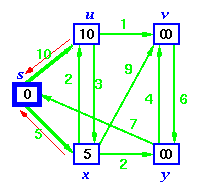
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Vértices** | **s** | **u** | **v** | **x** | **y** |
| **Estimativas** | **0** | **∞** | **∞** | **∞** | **∞** |
| **Precedentes** | **s** | **-** | **-** | **-** | **-** |



Passo 2

* Selecione s (vértice aberto de estimativa mínima)
* Feche s
* Recalcule as estimativas de u e x

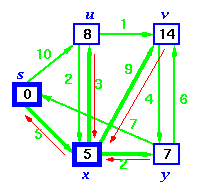
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Vértices** | **s** | **u** | **v** | **x** | **y** |
| **Estimativas** | **0** | **10** | **∞** | **5** | **∞** |
| **Precedentes** | **s** | **s** | **-** | **s** | **-** |



Passo 3

* Selecione x (vértice aberto de estimativa mínima)
* Feche x
* Recalcule as estimativas de u, v e y

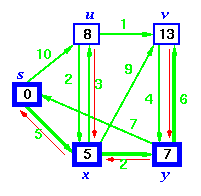
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Vértices** | **s** | **u** | **v** | **x** | **y** |
| **Estimativas** | **0** | **8** | **14** | **5** | **7** |
| **Precedentes** | **s** | **x** | **x** | **s** | **x** |



Passo 4

* Selecione y (vértice aberto de estimativa mínima)
* Feche y
* Recalcule as estimativas de v

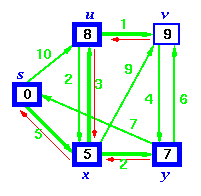
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Vértices** | **s** | **u** | **v** | **x** | **y** |
| **Estimativas** | **0** | **8** | **13** | **5** | **7** |
| **Precedentes** | **s** | **x** | **y** | **s** | **x** |



Passo 5

* Selecione u (vértice aberto de estimativa mínima)
* Feche u
* Recalcule as estimativas de v

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Vértices** | **s** | **u** | **v** | **x** | **y** |
| **Estimativas** | **0** | **8** | **9** | **5** | **7** |
| **Precedentes** | **S** | **X** | **U** | **S** | **X** |



Passo 6

* Selecione v (vértice aberto de estimativa mínima)
* Feche v

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Vértices** | **s** | **u** | **v** | **x** | **y** |
| **Estimativas** | **0** | **8** | **9** | **5** | **7** |
| **Precedentes** | **S** | **X** | **U** | **S** | **X** |

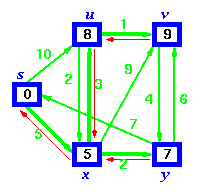


Tabela 1.0 – Sequência de diagramas – Algoritmo de Dijkstra (Parte fina l da tabela - pág. 2 8, 29, 30).

Fonte: MARIANI – (Livro eletrônico) Site http://www. inf.ufsc.br/grafos/(2012)

* Quando todos os vértices tiverem sido fechados, os valores obtidos serão os custos mínimos dos caminhos que partem do vértice tomado como raiz da busca até os demais vértices do grafos
* O caminho propriamente dito é obtido a partir dos vértices chamados acima de precedentes

Para exemplificar, considere o caminho de custo mínimo que vai de s até v, cujo custo mínimo é 9. O vértice precedente de v na última das tabelas acima é u. Sendo assim, o caminho é:

S 🡪... 🡪 u 🡪 v

Por sua vez, o precedente de u é x. Portanto, o caminho é:

s 🡪 ... 🡪 x 🡪 u 🡪 v

Por último, o precedente de x é o próprio vértice s. Logo, o caminho de custo mínimo é:

s 🡪 x 🡪 u 🡪 v

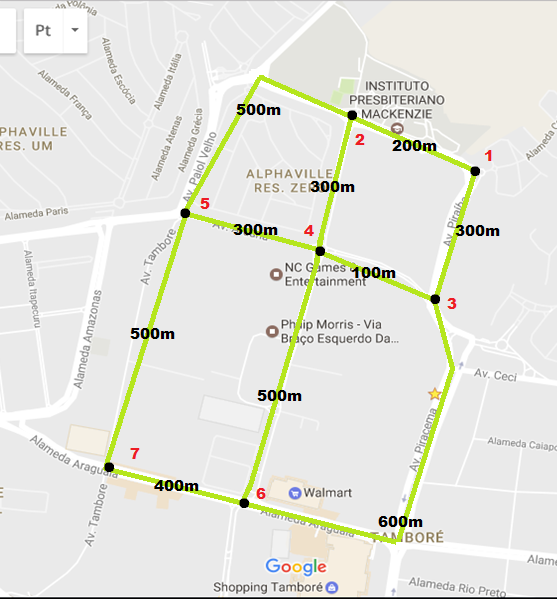
Então o Dijkstra encontra o caminho mínimo de origem única que tem como características; O funciona para grafos ponderados, apenas para arestas com peso positivo e os grafos podem conter ciclos.

* **Problema do mundo real para criação do grafo e denominar para arestas e vértices**

- Calculo de Rotas, esse é um problema que muitas pessoas enfrentam, como empresas de Taxis, Ubers, pessoas normais quando viajam etc. Todos querem o caminho mais curto, tanto para chegar mais rapido quanto para economizar em combustivel, mas como encontrar o caminho mais curto?

Resposta: Ultilização do Algoritmo de Dijkstra em um grafo, como demonstrado a baixo, usando o algoritmo demonstrado na Estrutura e Analise do algoritmo. O caminho mais curto do ponto 1 ao 7 seria 1-2-5-7 com um custo de 1.2km.

* Grafo.



• vértices

Arestas

Bibliografia:

<http://www.inf.ufsc.br/grafos/temas/custo-minimo/dijkstra.html>

<http://homepages.dcc.ufmg.br/~rainerpc/cursos/grafos/aulas/a05.pdf>