Uma análise comportamental do algoritmo Dijkstra com *heap* 3-nário

Ramayane Bonacin Braga[2], **Renato Gomes Borges Júnior**[1]

[1]Instituto de Informática – Universidade Federal de Goiás (UFG)
Caixa Postal 131 – 74.001-970 – Goiânia – GO – Brazil

ramayanebraga@inf.ufg.br, renatoborges@inf.ufg.br

Abstract. *The goal of this work is to analyze the behavior of an implementation of the Dijkstra's algorithm, in the C++ language, using a 3-nary heap as a data structure in a road network as data base. It was analyzed source nodes of different degrees and measured execution time and the behavior of the heap in relation to the amount of nodes and height per iteration.*

Resumo. *Este trabalho tem como objetivo analisar o comportamento da implementação do algoritmo de Dijkstra, na linguagem C++, utilizando a estrutura de dados de um heap 3-nário em uma rede viária como base de dados. Foi analisado nós fontes de diferentes graus e medido tempo de execução e o comportamento do heap em relação a quantidade de nós e altura por iteração.*

1. Introdução

O impacto da crescente evolução urbana, tem sido objeto de estudo para diversas pesquisas e discussões em temas como: mobilidade e a modelagem de redes viárias. Uma rede viária pode ser considerada um conjunto de caminhos onde cada caminho possui um valor ou peso específico. Uma forma eficiente de representar uma modelagem é a implementação de um grafo.

Podemos utilizar um grafo para representar diferentes problemas de uma forma clara e precisa, como por exemplo, um mapa político, definir um número mínimo de cores, levando em consideração os países vizinhos [Dasgupta et al. 2009].

Neste contexto, um dos problemas mais conhecidos de representação por grafos é o caminho mínimo ou menor caminho e tem como principal objetivo relacionar um percurso mínimo entre os vértices de um grafo. Formalmente, um grafo é definido por $G = (V, E)$, onde V representa o conjunto de vértices (também chamados de nós) e E representa arestas entre pares de vértices, como ilustrado na figura 1 [Dasgupta et al. 2009].

A solução mais conhecida para a implementação do problema de caminho mínimo em grafos é o algoritmo de Dijkstra. O algoritmo parte de um vértice inicial e a cada iteração busca o caminho de menor custo em relação aos vértices adjacentes a ele, repetindo o processo até que todos os vértices tenham sido visitados [Dijkstra 1959].

2. Metodologia

O algoritmo foi implementado na linguagem C++ de forma a permitir a análise do seu comportamento. Foi usado um *heap* para otimizar a eficiência pela busca do caminho mínimo com o Dijkstra.

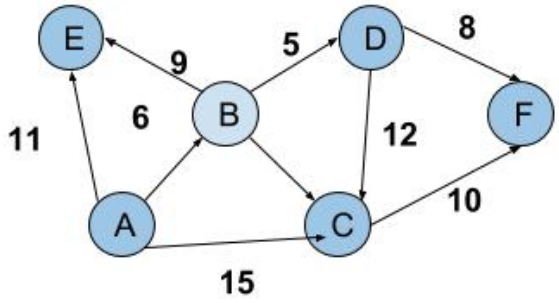

Figure 1. Representação de um grafo com pesos nas arestas.

Um *heap* pode ser considerado uma estrutura de dados eficiente para suportar as operações constrói, insere, retira, substitui e altera (ZIVIANI, 2002). O *heap* escolhido para esta análise foi o *heap* d-nário pois é de fácil implementação, com $d = 3$ por possuir um melhor desempenho.

Como base de dados para a análise, foi utilizado a rede viária da cidade de Nova Iorque disponibilizado na página web DIMACS [dim]. O grafo gerado possui 264.346 vértices e 733.846 arestas.

Foram escolhidos k vértices de grau mínimo, máximo e médio como parâmetros iniciais para executar o algoritmo de Dijkstra. Para determinar o valor de k, buscou-se a menor quantidade de vértices que teria a mesma quantidade de graus em comum (mínimo ou máximo). Assim, foram encontrados 41.050 vértices com grau mínimo de 1 e 2 vértices com grau máximo de 8, portanto foi atribuido o valor 2 para k.

Para os testes, foi usado uma máquina com as seguintes configurações: CPU Intel Core i5 1.7 GHz, RAM 8 GB e Sistema Operacional Ubuntu 14.04 64-bit.

3. Resultados

Após a execução do algoritmo os resultados foram obtidos analisando as seguintes métricas: altura e quantidade de elementos no *heap* a cada iteração e tempo de execução total.

Considerando os parâmetros mencionados anteriormente, a tabela 1 mostra os tempos de execução médio para cada grau escolhido. Nota-se que não há muita diferença entre os tempos de execução para os vértices de diferentes graus.

Grau	Tempo de execução (segundos)
Mínimo	0.134683
Médio	0.135482
Máximo	0.139509

Table 1. Tempo de execução médio do algoritmo de Dijkstra para vértices de grau mínimo, máximo e médio.

O tamanho do *heap* a cada iteração do algoritmo para os graus mínimo, máximo e médio estão apresentados graficamente nas figuras 2, 3 e 4 respectivamente.

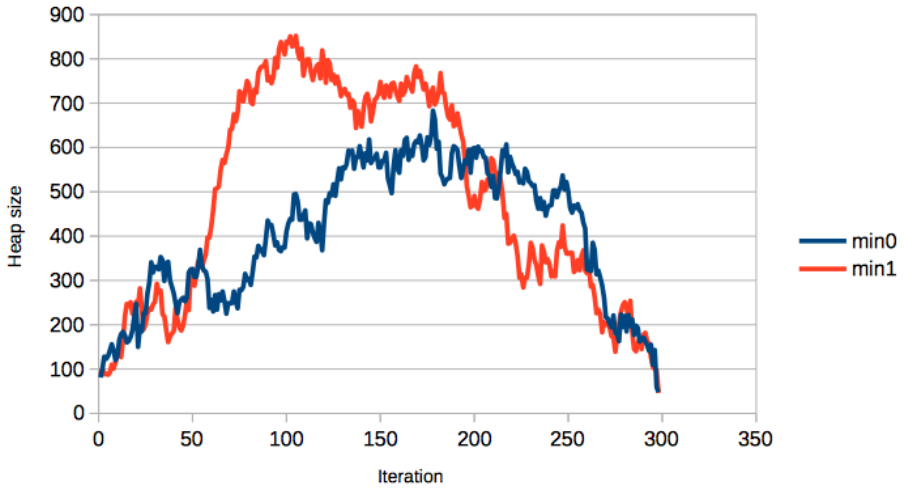

Figure 2. Número de nós no *heap* em cada iteração do algoritmo para k vértices de grau mínimo.

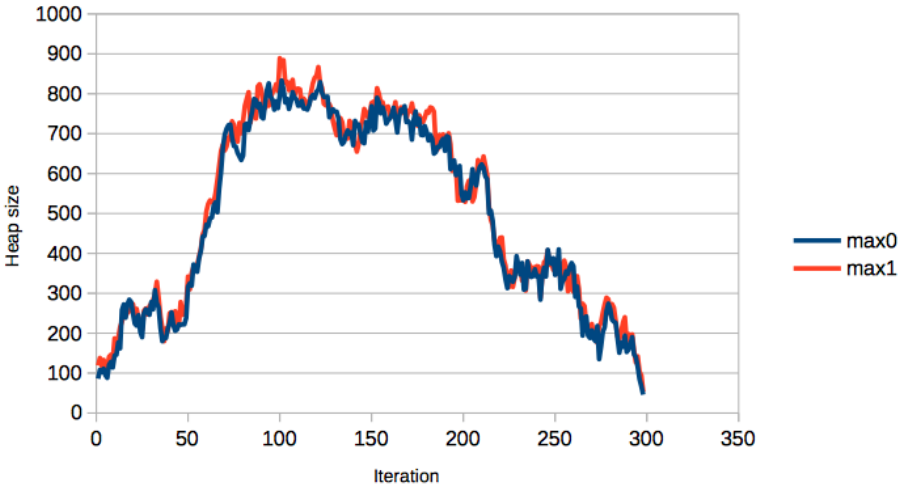

Figure 3. Número de nós no *heap* em cada iteração do algoritmo para k vértices de grau máximo.

Observa-se na figura 2 que para o segundo vértice de grau mínimo o número de elementos no *heap* cresce mais rapidamente nas primeiras iterações.

O comportamento do *heap* dos dois vértices em ambos os graus máximo e médio tiverem crescimento semelhante em relação a quantidade de elementos, como ilustrado nas figuras 3 e 4.

Após a análise dos logs, verificou-se que a altura máxima obtida no *heap* 3-nário para todos os vértices foi 6.

4. Conclusão

Levando em consideração a perspectiva da análise da execução do algoritmo de Dijkstra implementado com um *heap* 3-nário para a base de dados da rede viária de Nova Iorque, nota-se que o *heap* não apresenta muita variação em seu tamanho e altura para os vértices de diferentes graus.

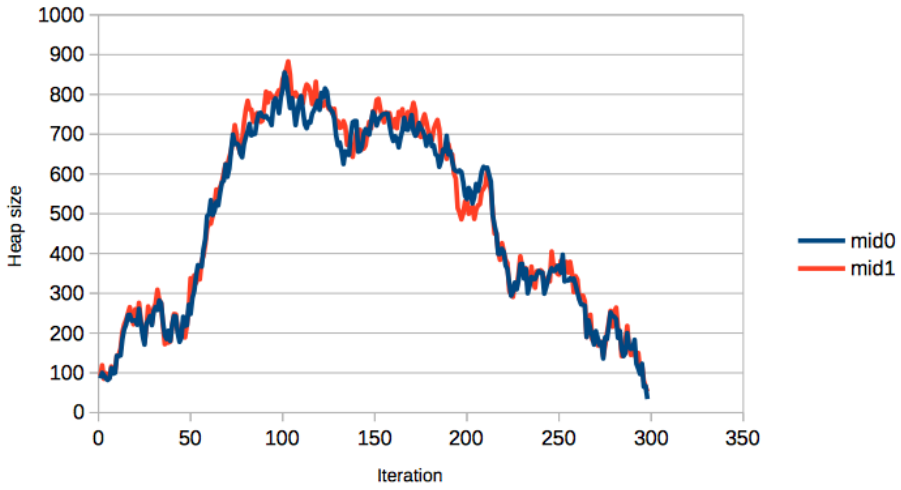

Figure 4. Número de nós no *heap* em cada iteração do algoritmo para k vértices de grau médio.

References

9th dimacs implementation challenge. `http://www.dis.uniroma1.it/challenge9/download.shtml`. Accessed: 2016-07-29.

Dasgupta, S., Papadimitriou, C., and Vazirani, U. (2009). *Projeto de algoritmos: com implementação em Pascal e C*. McGraw-Hill, São Paulo.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *NUMERISCHE MATHEMATIK*, 1(1):269–271.