UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

J. RAPIDEZ

Laboratório 1

Porto Alegre, agosto de 2013

1.1 Tarefa

Implementar o algoritmo de X, que permite calcular um Y em tempo $O(T(n))$. Validar experimentalmente que o desempenho é $O(T(n))$. Determinar a constante de proporcionalidade c, tal que o tempo é $c \log n$.

1.2 Solução

Implementei o algoritmo de X usando uma estrutura da dados Y. Um problema particular na solução foi fazer Z, que resolvemos fazendo W.

Não existem restrições para a forma da solução: a escolha de estruturas de dado, por exemplo, e livre. Qualquer linguagem de programação que possui uma implementação (compilador, interpretador) em software livre, disponível para um Ubuntu Linux é admissível.

1.3 Ambiente de teste

Os resultados foram obtidos numa Cray X-MP, com 24 processadores de 200 MHz, e 256 GB de RAM. Testamos com dados gerados randomicamente de tamanho $n = 2, 4, 8, \ldots, 2^{12}$. Cada teste foi repetido 20 vezes.

1.4 Resultados

Dicas para avaliar o tempo:

- Sempre se for possível, não medir o tempo de uma execução de uma única operação ou execução, mas de várias, e relatar o tempo médio. Uma diretiva simples é não medir tempo menor que 1s.
- Pode ser difícil comparar visualmente um gráfico de medidas com o tempo teórico esperado. Uma maneira para facilitar a comparação é dividir o tempo observado $T_o(n)$ pelo tempo esperado $T(n)$. Para $n \to \infty$ a curve deve se aproximar a uma constante.
- Uma outra abordagem com um modelo genérico: A complexidade de tempo de um algoritmo prático com alta probabilidade possui a forma

  $$T(n) \sim ab^n n^c \log^d n$$

  (ver p.ex. Sedgewick e Wayne [3, cáp. 1.4] e Sedgewick [2]). Frequentemente podemos focar em dois casos simples. Para uma série de medidas (n, T) podemos avaliar

  uma hipótese exponencial Com $T(n) \sim ab^n$, obtemos $\log T \sim \log a + n \log b$. Logo podemos determinar um modelo por regressão linear entre $\log T$ e n;

  uma hipótese polinomial Com $T(n) \sim an^b$ obtemos $\log T \sim \log a + b \log n$. Logo podemos determinar um modelo por regressão linear entre $\log T$ e $\log n$.

n =	2	4	8	16	32	64
T_o [μs]	0.77	0.87	1.19	2.36	7.57	27.91
n =	128	256	512	1024	2048	4096
T_o [μs]	92.52	336.13	1434.72	6024.10	24390.20	100000.00

Tabela 1.1: Tempo de execução T_o do algoritmo de Dijkstra para um grafo com $n = 2^i$, $i = 2, \ldots, 12$ vértices é $\approx 0.6n^2$ arestas.

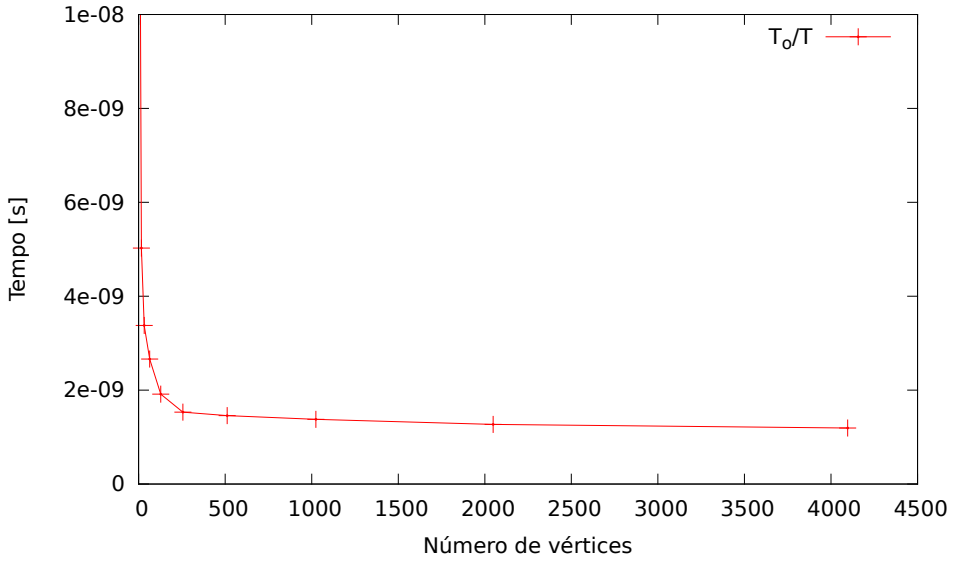

Figura 1.1: Tempo de execução normalizado $T_o/(0.6n^2+n)\log n$ do algoritmo de Dijkstra para um grafo com $n = 2^i$, $i = 2, \ldots, 12$ vértices é $\approx 0.6n^2$ arestas.

A Tabela 1.1 mostra o tempo de execução do algoritmo de Dijkstra para um grafo com $n = 2^i$, $i = 2, \ldots, 12$ vértices é $\approx 0.6n^2$ arestas. Usando uma hipótese linear obtemos $a = 54\text{ns}$ e $b \approx 1.65$. A complexidade pessimista teórica é $(n + m)\log n = (0.6n^2 + n)\log n$.

Dicas:

- Alinhar colunas numéricas sempre à direita.
- Informar quantidades com um número de dígitos razoável e arredondar corretamente.

O tempo de execução dividido pelo tempo esperado é mostrado na Fig. 1.1.

Dicas para figuras:

- Rotular os eixos.

1.5 Conclusão

O algoritmo X comporta-se como esperado nos intervalos A e B e um melhor desempenho para valores C e D. Isso pode ser explicado por Y.

Muito mais dicas: Johnson [1].

BIBLIOGRAFIA

[1] David S. Johnson. “A theoretician’s guide to the experimental analysis of algorithms”. Em: *Proceedings of the 5th and 6th DIMACS Implementation Challenges*. 2002.

[2] Robert Sedgewick. *Algorithms for the Masses*. ANALCO’11, San Francisco. Jan. de 2011.

[3] Robert Sedgewick e Kevin Wayne. *Algorithms*. 4th. Addison-Wesley, 2011.