

CS 577 - Basics of Algorithm Analysis

Marc Renault

Department of Computer Sciences
University of Wisconsin – Madison

Summer 2023

TopHat Section 001 Join Code: 275653



ALGORITHM ANALYSIS

Algorithm Evaluation

- Sound
- Complete
- Resource requirements:
 - Time
 - Space
 - Other...

COMPUTATIONAL TRACTABILITY

DEFINING EFFICIENCY

Definition 1¹

An algorithm is efficient if, when implemented, it runs quickly on real input instances.

Issues:

- Not concrete enough for meaning algorithm comparison.
- What is “quickly”?
- What are “real input instances”?

¹Algorithm Design, p. 30

DEFINING EFFICIENCY

Definition 2²

An algorithm is efficient if it achieves qualitatively better *worst-case* performance, at an analytical level, than *brute force* search.

²Algorithm Design, p. 32

QUANTIFYING AN ALGORITHM'S PERFORMANCE

Brute-force

- 1 Enumerate all possible solutions.
- 2 Check all possible solutions and keep the best one.

Worst-case

Considering all possible inputs, what is worst possible performance of the algorithm?

- Absolute guarantee on performance.
- Only needs one data point.

DEFINING EFFICIENCY

Definition 2²

An algorithm is efficient if it achieves qualitatively better *worst-case* performance, at an analytical level, than *brute force* search.

Issues:

- Still too vague for a good measure.
- What exactly is “qualitative”?

²Algorithm Design, p. 32

STABLE MARRIAGE PROBLEM (SMP) (1962)¹²³

Problem Definition

Given a set of n men, M , and an opposite set of n women, W . Each person has a preference ranking of the opposite set. Compute a stable matching between M and W . A matching is stable if it is (i) perfect, and (ii) there are no pairs (m, w) and (m', w') in the matching where m prefers w' and w' prefers m .

- A.k.a Stable Matching Problem.
- There are more complicated variations of the model.
- Used in the real world (e.g. matching doctors to hospitals).
- Nobel Prize in Economics in 2012 (Shapley and Roth).

¹Algorithm Design, Ch 1.

²Algorithms, Ch 4.5

³<http://mathsite.math.berkeley.edu/smp/smp.html>

ANALYSIS OF SMP

Algorithm: Gale-Shapley Algorithm (1962)

Initially all $m \in M$ and $w \in W$ are free

while *there is a man m who is free and hasn't proposed to every woman* **do**

 Choose such a man m

 Let w be the highest-ranked woman in m 's preference list to whom m has not yet proposed

if w is free **then**

(m, w) become engaged

else w is currently engaged to m'

if w prefers m' to m **then**

m remains free

else w prefers m to m'

(m, w) become engaged

m' becomes free

end

end

end

return *the set S of engaged pairs*

TopHat 1

How many brute-force possibilities when there are n men and n women?

DEFINING EFFICIENCY

Definition 3³

An algorithm is efficient if it has a polynomial running time with respect to the input size.

Polynomial: $f(n) = c_d \cdot n^d + c_{d-1} \cdot n^{d-1} + \dots + c_1 \cdot n + c_0$, where d and c_i are constants.

Well defined notion:

- Natural follow-up: what is the most efficient algorithm possible?
- Not perfect: n^{100} is polynomial, but $n^{1+0.02(\log n)}$ is not.

³Algorithm Design, p. 32

ANALYSIS OF SMP

Algorithm: Gale-Shapley Algorithm (1962)

Initially all $m \in M$ and $w \in W$ are free

while *there is a man m who is free and hasn't proposed to every woman* **do**

 Choose such a man m

 Let w be the highest-ranked woman in m 's preference list to whom m has not yet proposed

if w is free **then**

(m, w) become engaged

else w is currently engaged to m'

if w prefers m' to m **then**

m remains free

else w prefers m to m'

(m, w) become engaged

m' becomes free

end

end

end

return *the set S of engaged pairs*

TopHat 2

In an implementation of SMP, what would be the input size when there are n men and n women?

ANALYSIS OF SMP

Algorithm: Gale-Shapley Algorithm (1962)

Initially all $m \in M$ and $w \in W$ are free

while *there is a man m who is free and hasn't proposed to every woman* **do**

 Choose such a man m

 Let w be the highest-ranked woman in m 's preference list to whom m has not yet proposed

if w is free **then**

(m, w) become engaged

else w is currently engaged to m'

if w prefers m' to m **then**

m remains free

else w prefers m to m'

(m, w) become engaged

m' becomes free

end

end

end

return *the set S of engaged pairs*

TopHat 3

In the Gale-Shapely algorithm, what is the maximum number of iterations when there are n men and n women?

QUANTIFYING AN ALGORITHM'S PERFORMANCE

Brute-force

- 1 Enumerate all possible solutions.
- 2 Check all possible solutions and keep the best one.

Worst-case

Considering all possible inputs, what is worst possible performance of the algorithm?

- Absolute guarantee on performance.
- Only needs one data point.

QUANTIFYING AN ALGORITHM'S PERFORMANCE

Worst-case

Considering all possible inputs, what is worst possible performance of the algorithm?

- Absolute guarantee on performance.
- Only needs one data point.

Average-case

Given a distribution over the possible inputs, what is the expected performance of the algorithm?

- Without mention of distribution, uniform is assumed.
- Analysis typically more complicated.

QUANTIFYING AN ALGORITHM'S PERFORMANCE

Worst-case

Considering all possible inputs, what is worst possible performance of the algorithm?

- Absolute guarantee on performance.
- Only needs one data point.

Best-case

Considering all possible inputs, what is best possible performance of the algorithm?

- Tends to be meaningless
- Could used when choosing between 2 otherwise equivalent algorithms.

INSERTION SORT ANALYSIS

(INTRODUCTION TO ALGORITHMS, P.26)

INSERTION-SORT(A)

1 **for** $j = 2$ **to** $A.length$

2 $key = A[j]$

3 // Insert $A[j]$ into the sorted
 sequence $A[1..j-1]$.

4 $i = j - 1$

5 **while** $i > 0$ and $A[i] > key$

6 $A[i+1] = A[i]$

7 $i = i - 1$

8 $A[i+1] = key$

cost

times

c_1

n

c_2

$n - 1$

0

$n - 1$

c_4

$n - 1$

c_5

$\sum_{j=2}^n t_j$

c_6

$\sum_{j=2}^n (t_j - 1)$

c_7

$\sum_{j=2}^n (t_j - 1)$

c_8

$n - 1$

Overall:

$$T(n) = c_1 n + c_2(n - 1) + c_4(n - 1) + c_5 \sum_{j=2}^n t_j + c_6 \sum_{j=2}^n (t_j - 1) + c_7 \sum_{j=2}^n (t_j - 1) + c_8(n - 1)$$

$$\leq c_1 n + c_2(n - 1) + c_4(n - 1) + c_5 \sum_{j=2}^n j + c_6 \sum_{j=2}^n (j - 1) + c_7 \sum_{j=2}^n (j - 1) + c_8(n - 1)$$

INSERTION SORT ANALYSIS

(INTRODUCTION TO ALGORITHMS, p.26)

INSERTION-SORT(<i>A</i>)	<i>cost</i>	<i>times</i>
1 for <i>j</i> = 2 to <i>A.length</i>	c_1	n
2 $key = A[j]$	c_2	$n - 1$
3 // Insert $A[j]$ into the sorted sequence $A[1 \dots j - 1]$.	0	$n - 1$
4 $i = j - 1$	c_4	$n - 1$
5 while $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^n t_j$
6 $A[i + 1] = A[i]$	c_6	$\sum_{j=2}^n (t_j - 1)$
7 $i = i - 1$	c_7	$\sum_{j=2}^n (t_j - 1)$
8 $A[i + 1] = key$	c_8	$n - 1$

Overall:

$$\begin{aligned}
 T(n) &\leq c_1 n + c_2(n - 1) + c_4(n - 1) + c_5 \sum_{j=2}^n j + c_6 \sum_{j=2}^n (j - 1) + c_7 \sum_{j=2}^n (j - 1) + c_8(n - 1) \\
 &= an^2 + bn - d
 \end{aligned}$$

ASYMPTOTIC ORDER OF GROWTH

ASYMPTOTIC ORDER OF GROWTH

Bounding $f(n)$ as n grows

- Bound $f(n)$ from above.
- Bound $f(n)$ from below.

Bachmann–Landau notation (Asymptotic notation)

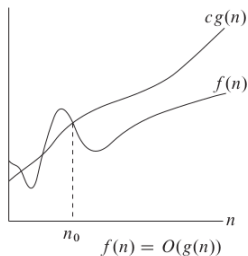
- | | |
|------------------------------------|-------------------------------|
| • Big-Oh: $O (\leq)$ | • Little-oh: $o (<<)$ |
| • Big-Omega: $\Omega (\geq)$ | • Little-omega: $\omega (>>)$ |
| • Big-Theta: Θ (equivalent) | |

BIG-OH

ASYMPTOTIC UPPER BOUND

Formal Definition¹

$$O(g(n)) = \{f(n) : \exists c, n_0 > 0 \mid \\ 0 \leq f(n) \leq cg(n) \forall n \geq n_0\}$$



Insertion sort:

$$T(n) = an^2 + bn - d \in O(n^2)$$

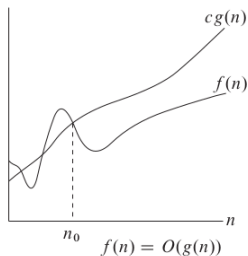
¹Introduction to Algorithms, Ch 3.1

BIG-OH

ASYMPTOTIC UPPER BOUND

Formal Definition¹

$$O(g(n)) = \{f(n) : \exists c, n_0 > 0 \mid \\ 0 \leq f(n) \leq cg(n) \forall n \geq n_0\}$$



Insertion sort:

$$T(n) = an^2 + bn - d = O(n^2)$$

Often used, but technically an abuse of notation

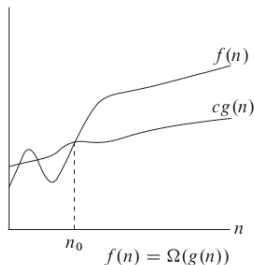
¹Introduction to Algorithms, Ch 3.1

BIG-OMEGA

ASYMPTOTIC LOWER BOUND

Formal Definition¹

$$\Omega(g(n)) = \{f(n) : \exists c, n_0 > 0 \mid \\ 0 \leq cg(n) \leq f(n) \forall n \geq n_0\}$$



Insertion sort:

$$T(n) = an^2 + bn - d \in \Omega(n^2)$$

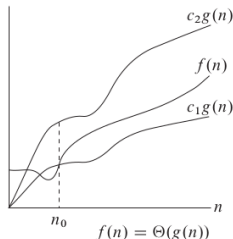
¹Introduction to Algorithms, Ch 3.1

BIG-THETA

ASYMPTOTIC TIGHT BOUND

Formal Definition¹

$$\Theta(g(n)) = \{f(n) : \exists c_1, c_2, n_0 > 0 \mid \\ 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \forall n \geq n_0\}$$



Insertion sort:

$$T(n) = an^2 + bn - d \in \Theta(n^2)$$

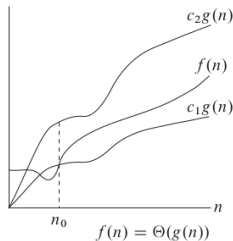
¹Introduction to Algorithms, Ch 3.1

BIG-THETA

ASYMPTOTIC TIGHT BOUND

Formal Definition¹

$$\Theta(g(n)) = \{f(n) : \exists c_1, c_2, n_0 > 0 \mid \\ 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \forall n \geq n_0\}$$



Key Property

For any two functions $f(n)$ and $g(n)$, we have $f(n) = \Theta(g(n))$ if and only if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

¹Introduction to Algorithms, Ch 3.1

LITTLE-OH

Formal Definition¹

$$o(g(n)) = \{f(n) : \forall c > 0 \exists n_0 > 0 \mid \\ 0 \leq f(n) < cg(n) \forall n \geq n_0\}$$

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$$

Insertion sort:

$$\begin{aligned} T(n) = an^2 + bn - d &\in o(n^3) \\ &\in O(n^3) \\ &\in O(n^2) \\ &\notin o(n^2) \end{aligned}$$

¹Introduction to Algorithms, Ch 3.1

LITTLE-OMEGA

Formal Definition¹

$$\omega(g(n)) = \{f(n) : \forall c > 0 \exists n_0 > 0 \mid \\ 0 \leq cg(n) < f(n) \forall n \geq n_0\}$$

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \infty$$

Insertion sort:

$$\begin{aligned} T(n) &= an^2 + bn - d \in \omega(n) \\ &\in \Omega(n) \\ &\in \Omega(n^2) \\ &\notin \omega(n^2) \end{aligned}$$

¹Introduction to Algorithms, Ch 3.1

USEFUL ASYMPTOTIC PROPERTIES

Polynomial Bound

For $c_d > 0$, $f(n) = c_d \cdot n^d + c_{d-1} \cdot n^{d-1} + \dots + c_1 \cdot n + c_0 = O(n^d)$

Logarithms

- $\log_b n = \frac{\log_a n}{\log_a b} = \Theta(\log n)$
- $(\log n)^a = o(n^b)$ for any $a, b > 0$

Exponential

- For every $r > 1$ and every $d > 0$, $n^d = o(r^n)$
- $r^n = o(s^n)$ for $r < s$

ANALYSIS OF SMP

Algorithm: Gale-Shapley Algorithm (1962)

Initially all $m \in M$ and $w \in W$ are free

while *there is a man m who is free and hasn't proposed to every woman* **do**

 Choose such a man m

 Let w be the highest-ranked woman in m 's preference list to whom m has not yet proposed

if w is free **then**

(m, w) become engaged

else w is currently engaged to m'

if w prefers m' to m **then**

m remains free

else w prefers m to m'

(m, w) become engaged

m' becomes free

end

end

end

return *the set S of engaged pairs*

Exercise

How would you implement this algorithm so that it has a running time of $O(n^2)$?

COMMON RUNTIMES

