CS 120: Intro to Algorithms and their Limitations

Lecture 2: Measuring Efficiency Thursday, September 7, 2023

Pset Due: September 13, 2023 **Denny Cao**

---

§1 Announcements

- SRE starts Tuesday
- Participation surveys (see course website)

§2 Review

§2.1 Exhaustive Search Sort

Input: $B = ((k_0, v_0), ..., (k_{n-1}, v_{n-1}))$ for each perm π :
if $k_{\pi(0)} \leq k_{\pi(1) \leq ... \leq k_{\pi n-1}}$:
return $((k_{\pi(0)}, v_{\pi(0)}), ..., (k_{\pi(n-1)}, v_{\pi(n-1)})$

§2.2 Insertion Sort

Input: $B = ((k_0, v_0), ..., (k_{n-1}, v_{n-1}))$
for each $i = 0, 1, ..., n-1$:
Insert $B(i)$ at correct place in $(B(0), ..., B(i-1))$
return B

§2.3 Merge Sort

Input: $B = ((k_0, v_0), ..., (k_{n-1}, v_{n-1}))$
if $n \leq 1$, return B
else if $n = 2$ and $k_0 \leq k_1$, return B
else if $n = 2$ and $k_0 > k_1$, return $((k_1, v_1), (k_0, v_0))$
else: $i = \lceil n/2 \rceil$ $B_i = \text{MergeSort}((k_0, v_0), ..., (k_{i-1}, v_{i-1}))$ $B_2 = ...((k_i, v_i, ...(k_{n-1}, v_{n-1}))$
return $\text{Merge}(B_1, B_2)$

§2.4 Computational Problem

Definition 2.1 (Computational Problem)**.** A *computational problem* is a triple $\Pi = (\mathcal{I}, \mathcal{O}, f)$, where:

- $\mathcal{I}$ is the set of inputs/instances (typically infinity)
- $\mathcal{O}$ is the set of outputs
- $f(x)$ is the set of solutions
- $\forall x \in \mathcal{I}(f(x) \subseteq \mathcal{O})$
  - Some inputs have multiple correct outputs, and thus $f(x) \subseteq \mathcal{O}$ rather than just being an element. Example: If same key, different values. Then there are multiple answers to sorting.

Definition 2.2 (Algorithm)**.** An *algorithm* A solves Π if:

- $\forall x \in \mathcal{I}$ st $f(x) \neq \emptyset$ then $A(x) \in f(x)$
- $\forall x \in \mathcal{I}$ if $f(x) = \emptyset$, $A(x) = \perp$

> **Remark 2.3.** The second part must be added because there is a possibility that there is no solution and the algorithm must return something.

§3 Measuring Efficiency

Definition 3.1 (Run Time)**.** For an algorithm A, given a function size(x) the runtime is:

$$T(n) = \max(x : \text{size}(x) \leq n)$$

where $T : \mathbb{N} \to \mathbb{R}^+$.

- n : Length of input, constraint. Consider all input, x, with size less than or equal to n.
- $T(n)$: Maximize # of basic operations performing A on x
- size(x): # of (key, value) pairs
- Basic Operations: Arithematic operations, manipulating pointers, executing lines of code, assigning variables
- Non-basic Operations: Sorting
- We take the max of x because we are interested in the worst case scenario.
- size$(x) \leq n$ to make sure that $T(n)$ is increasing and valid on real inputs

Notation 3.2 (Big-Oh Notation) Let $f, g : \mathbb{N} \to \mathbb{R}^+$. We say:

- $f = O(g) : \exists c > 0, \forall n \geq k \in \mathbb{N} \mid f(n) \leq cg(n)$
- $f = \Omega(g) : \exists c > 0, \forall n \geq k \in \mathbb{N} \mid f(n) \geq cg(n) \leftrightarrow g = O(f)$
- $f = \Theta(g) :\ f = O(g) \wedge f = \Omega(g)$
- $f = o(g) : \lim\limits_{n\to\infty} \dfrac{f(n)}{g(n)} = 0$
- $f = \omega(g) : \lim\limits_{n\to\infty} \dfrac{f(n)}{g(n)} = \infty \leftrightarrow g = o(f)$

Example 3.3

Find the running time of ESS.

Solution. There are $n!$ permutations of input of size n. For each permutation, execute $n-1$ steps. Therefore, $T_{ESS}(n) = \Theta(n!(n-1)) = O(n!(n-1)) = \Omega(n!(n-1)) = \Theta(n!n)$. □

Example 3.4

Find the running time of Insertion Sort.

Solution. $T_{\text{ins sort}}(n) = O\left(\sum_{i=0}^{n} i\right) = O(n^2) = \Omega(n^2) = \Theta(n^2).$ $\square$

Example 3.5

Find the running time of Merge Sort.

Solution. $T_{\text{merge sort}}(n) \leq T_{\text{merge sort}}\left(\left\lceil \frac{n}{2} \right\rceil\right) + T_{\text{merge sort}}\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + \Theta(n) = O(n \log n)$ $\square$